PERT: A New Power-Efficient Real-Time Packet Delivery Scheme for Sensor Networks

Shanzhong Zhu*
Department of Computer Science and Engineering,
University of California, Riverside
Riverside, CA 92521
E-mail: szhu@cs.ucr.edu
*Corresponding author

Wei Wang
Department of Computer Science and Engineering,
University of California, Riverside
Riverside, CA 92521
E-mail: wangw@cs.ucr.edu

Chinya V. Ravishankar
Department of Computer Science and Engineering,
University of California, Riverside
Riverside, CA 92521
E-mail: ravi@cs.ucr.edu

Abstract: We present *PERT*, a power-efficient scheme to deliver real-time data packets in sensor networks. Time-sensitive sensor data is common in applications such as hazard monitoring systems, traffic control systems and battlefield command 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 of wireless network behavior using the `ns-2` simulation package.

Keywords: sensor networks, real-time, load-balanced routing, power conservation, packet aggregation.

Biographical notes: Shanzhong Zhu received his BS degree from the Honors Program at Zhejiang University, Hangzhou, China. He received his PhD degree in computer science from University of California at Riverside in 2006. He was awarded Dean's Fellowship from 1999 to 2005 at UCR. His research interests are sensor networks, data stream processing, and query optimization. Wei Wang received her B.Eng. and M.Eng. degrees in computer science and engineering from Northeastern University, China. She is currently working toward the Ph.D. degree in Computer Science at University of California, Riverside. Chinya V. Ravishankar has been with the University of California at Riverside since Fall 1999, where he is currently Professor of Computer Science and Engineering, and Associate Dean in the Bourns College of Engineering. Between 1986 - 1999, he was on the Faculty of the Electrical Engineering and Computer Science Department at the University of Michigan - Ann Arbor. Prof. Ravishankar's research is currently in the areas of Databases, Networking, and Security, and broadly in the area of software systems.

1 Introduction

There has been considerable interest in large networks of wireless sensors in recent years. Such sensor networks are widely used in many applications, such as microclimate monitoring of redwood trees (Redwood), freeway traffic monitoring (Madden and Franklin, 2002), animal habit tracking (Mainwaring et al., 2002), and the monitoring of the physiological status of soldiers (Warfighter). 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 (Lu et al., 2002; He et al., 2003; Liu et al., 2003), 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 (Heinzelman et al., 2000). 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.

1.1 Power-Efficient Real-Time Delivery

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.

1.1.1 Our Approach in PERT

We achieve both real-time delivery and power efficiency in PERT through two new 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 (Madden and Franklin, 2002). 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 (Lu et al., 2002) and SPEED (He et al., 2003) 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. The actual relay rate is estimated at each node, and packets are dropped only when no outgoing link can sustain the desired rate. 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. It guarantees that each path is the shortest from the source to the BS, and the traffic loads are very well balanced among the nodes at the same level (see Section 4). In our simulation (see Section 6), we compare PERT with RAP and show that PERT achieves much lower deadline miss ratio and saves more power.

1.2 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 is very important for delivering real-time data packets in sensor networks, since unbalanced traffic causes congestion and hot spots. Packets are then more likely to miss their deadlines under heavy loads, or even get dropped at highly congested nodes. Besides, nodes that relay high packet volumes will exhaust their power quickly, possibly causing part of the network unreachable. 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. Sensor readings may be delayed at relaying nodes for a total time not exceeding their slack time, but we must distribute this slack time across relaying nodes so that the overall number of transmissions is minimized. 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. Our simulation results suggest that both schemes contribute to extending the lifetime of the sensor network and lowering end-to-end delays (see Section 6).

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. We also compare our approach to allocate hold times with other approaches, and find that much fewer transmissions are generated in our approach.

The rest of this paper is organized as follows: We review some related work in Section 2. Our system model is introduced in Section 3. In Section 4, we propose our load balanced routing scheme, which aims to balance the traffic over the relaying nodes. In Section 5, we discuss our packet grouping mechanism that allows relaying nodes to combine packets together. Simulation results are shown in Section 6 to verify our approach. Section 7 concludes our work.

2 Related Work

The literature largely addresses, in isolation, the issue of real-time packet delivery in the demanding sensor networks environment. For example, SWAN (Ahn et al., 2002) proposes a stateless network model to deliver service differentiation in wireless ad-hoc networks. To address the delay requirements of the real-time UDP traffic, rate control of the best-effort TCP and UDP traffic is performed at each node in the network. 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 form larger packets.

A novel real-time routing protocol, SPEED, was introduced by He et al. (2003), 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, however, minimizes congestion via load-balanced routing and packet grouping, so packets can meet deadlines with high confidence.

Several new MAC layer protocols have been designed to accommodate real-time requirements. Lu et al. (2002) proposed RAP, a new real-time communication architecture for large-scale sensor networks. Its *velocity monotonic scheduling* mechanism is a key component in prioritizing real-time traffic at the MAC layer. Caccamo et al. (2002) proposed an EDF-based MAC layer protocol. The periodicity of sensor-generated traffic allows contention to be resolved implicitly, without need for exchange-control packets for channel reservation. Another contention-free TDMA-based MAC protocol was proposed by Carley et al. (2003), where high scalability can be achieved due to the low time and space complexity of the scheduler at each node. 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 (Madden et al., 2002; Sharaf et al., 2003; Intanagonwiwat et al., 2000; Manjhi et al., 2005). It is typical to compute partial results intermediate nodes and send them to the BS. Madden et al. (2002) provided a classification of commonly used aggregates (SUM, AVG, COUNT, etc.), according to their sensitivity to duplicates and state requirement in the sensor network. Approaches have also been proposed to build in-network aggregation trees to minimize overall power consumption (Madden et al., 2002; Sharaf et al., 2003; Manjhi et al., 2005). 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? [1]

[1] We use the verbs 'group' and 'aggregate' interchangeably, mean-

The idea of combining small packets into larger packets was first studied in the Internet context (Badrinath and Sudame, 2000), 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 by He et al. (2004), 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.

3 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. In a wild-fire monitoring system, sensors are deployed in a forest to monitor the temperature of the surrounding area. High temperature values must be delivered to the central server with tighter deadlines than low values, since they may indicate fires.

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 5).

A packet en route to the BS may encapsulate multiple sensor readings. Along its way, it can be disassembled, delayed, and its sensor readings grouped with sensor readings from other sources.

3.1 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

ing that we combine small data packets into a larger one.

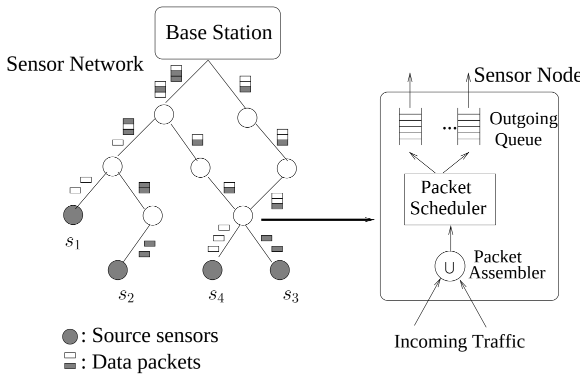


Figure 1: The System Model

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 4).

3.2 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 6). This model is more general and challenging than assuming *periodical* generation of sensor readings, such as Chipara et al. (2005).

3.3 The Power Model

Power efficiency is our primary goal. Typically, power is required for communications, computation, and data sensing (sampling). We focus on reducing the power consumed by radio communications. Since moderate computation takes place on sensors in our model, the power for computation is an order of magnitude lower than the power for communication and can be ignored (Hill et al., 2000). The data generation (sampling) rate is application-dependent (Madden et al., 2003), and is orthogonal to our scheme.

A sensor's radio communication module may operate in one of the following modes: *transmitting (T)*, *receiving (R)*, *idle (I)*, or *sleeping (S)* (Feeney and Nilsson, 2001). In the idle mode, the sensor listens to the channel, waiting for incoming packets. In the sleeping mode, it turns off its radio, so that the power consumed is negligible compared with other modes. Therefore, the total consumed power P is the sum of the power consumed in modes T, R, and I: $P = P_T + P_R + P_I$.

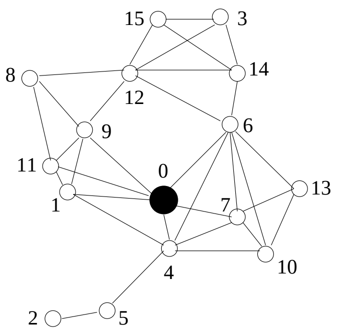
We achieve power efficiency by two means. First, we balance the power consumption at individual nodes by balancing the number of packets they transmit using our load-balanced routing scheme. Balancing power consumption extends network lifetime by avoiding bottleneck nodes that will exhaust their power 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.

To further prolong the lifetime of sensor nodes, we must regularly schedule sensors to sleep without jeopardizing the network's ability to deliver real-time packets. We may adopt schemes such as GAF (Xu et al., 2001) and SPAN (Chen et al., 2001), both of which maintain a routing backbone to ensure connectivity of the network, while allowing as many nodes as possible to sleep. At least one routing node is guaranteed to be within the transmission range of any node. Periodically, the routing nodes are rotated to ensure even power dissipation. Our approach can be seamlessly built on the top of such duty cycling schemes, since, at any time, an enough number of nodes are awake to route real-time packets.

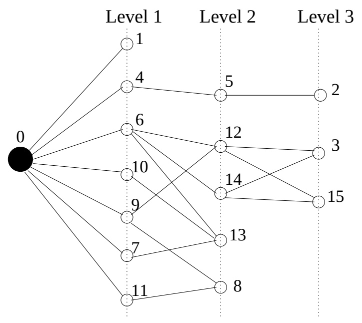
4 Load Balanced Routing

Routing in ad hoc networks has been extensively studied, and various routing schemes, such as DSDV (Perkins, 1994), DSR (Johnson and Maltz, 1996), and AODV (Perkins and Royer, 1999), 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, possibly causing part of the network to become unreachable. Moreover, a bottleneck node will cause packets to experience longer delays, possibly missing their deadlines at the BS (see Section 6.2).

Several load balanced routing schemes have been proposed for wireless sensor networks (Hsiao et al., 2001; Huang and Jan, 2004; Hong et al., 2002). Hsiao et al. (2001) constructed a *load balanced backbone tree (LBB-tree)* to balance the loads over the nodes that are one hop away from the BS. In the 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 6.2). Hong et al. (2002) propose 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. A node is at level i if the lowest hop count from it to the BS is i. As we will show in Fig. 2, our multi-path routing scheme can ditribute traffic more balanced than this simple random scheme. The notion of *layered network* was introduced by Huang and Jan (2004), based on which two load balanced routing schemes, MCP and MCP-PS, were proposed. In MCP, a shortest path with maximum available power is chosen to forward the packets. However, by letting all packets take the nodes with maximum available energy, these nodes will make packets experience much longer delays and may soon become bottlenecks for our real-time traffic.



(a) The sensor network



(b) The routing network

Figure 2: The routing network for 15 sensor nodes

Our goal is to achieve long network lifetime and low end-to-end delays by routing packets as evenly as possible inside the sensor network. First, we show how to build a *routing network* based on which packets are delivered to the BS (see Section 4.1). Then, we propose a novel load-balanced routing algorithm (see Section 4.2) to route data traffic onto multiple paths intelligently so that the traffic loads are balanced level by level throughout the entire sensor network. We discuss the time and space complexity of our algorithm in Section 4.3, and its power efficiency in Section 4.4.

4.1 The Routing Network

A *routing network* (RN) is a DAG (*directed acyclical graph*) (Cormen et al., 1997) 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 node depending on its distance to the BS, and to deliver data packets from higher-level nodes to lower-level nodes (Madden et al., 2002; Sharaf et al., 2003; Intanagonwiwat et al., 2000; Huang and Jan, 2004). 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 (Madden et al., 2002; Huang and Jan, 2004; Hong et al., 2002), the level of a node may equal the number of hops in the shortest distance to the BS, and its parents may be the neighbours on its shortest paths to the BS. In another (Sharaf et al., 2003), 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.

Fig. 2 shows the construction of an RN for a simple network of 15 sensor nodes. We use the same construction scheme as Hong et al. (2002). Node 0 is the BS. Initally, all nodes set their levels to infinity (∞). The BS broadcasts a query message with its ID and level number 0. When a node receives a query message from another node at level i, it updates its own level number to $i + 1$ only if $i + 1$ is smaller than its current assigned level, records those neighbours at level i that have signal strength higher than a threshold as its parents (we choose the nodes with good-quality links as parents), and broadcasts its own query message. This process is repeated until the query messages from all nodes flood the entire network. Clearly, any path from a source to the BS is a shortest path in the resulting RN (see Fig. 2(b)). We can also use other methods to determine levels and parents of the nodes, which may result in different RNs.

4.2 The Routing Algorithm

Given the RN, we propose a novel *load balanced routing* (LBR) scheme to balance traffic loads level by level, going 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 the next (lower) level. We map the routing problem to the *maximum flow* problem (Cormen et al., 1997) by constructing a *flow network* based on the link topology between the current level 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 an RN of 12 sensor nodes, of which 4 nodes (1, 6, 7, and 8) are sources generating data packets at certain rates. Traffic loads need to be balanced level by level, from level 3 to level 1. To determine the load distribution at level 2, we must calculate how much traffic should be assigned to each link from level 3 to level 2. We start from the construction of a flow network for nodes in level 2 and 3.

4.2.1 Constructing the Flow Network

As shown in Fig. 3(b), each level-2 or level-3 node in the RN is represented by a node in the flow network. A *virtual source* node *VS* and a *virtual sink* node *VT* are added to the flow network. 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 4.2.2).

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.

4.2.2 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*, that is, 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. Now 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.

In our example, the initial sink-edge capacity and η are set to 1.0 and 0.5, respectively (see Section 4.3 for how to set these values). Fig. 3(c) shows the resulting maximum flows. Note that all sink edges are saturated. When the capacities of the sink edges are increased to 6.5, as shown in Fig. 3(d), Nodes 4 and 5 can only take load 6.0, showing that we have reached the maximum flows over these nodes. The final loads on nodes 4 and 5 have now been obtained, and they can be removed from the flow network with their associated edges (see Fig. 3(e)). The capacities of the other sink edges are incremented to 13.0, when sink edges of nodes 2 and 3 can no longer be saturated, and can be removed from the network (see Fig. 3(f)). Now we only need to adjust the capacity of the edge (1, *VT*). The final flows on the sink edges are shown in Fig. 3(g).

After obtaining the final balanced loads (see Fig. 3(h)), 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

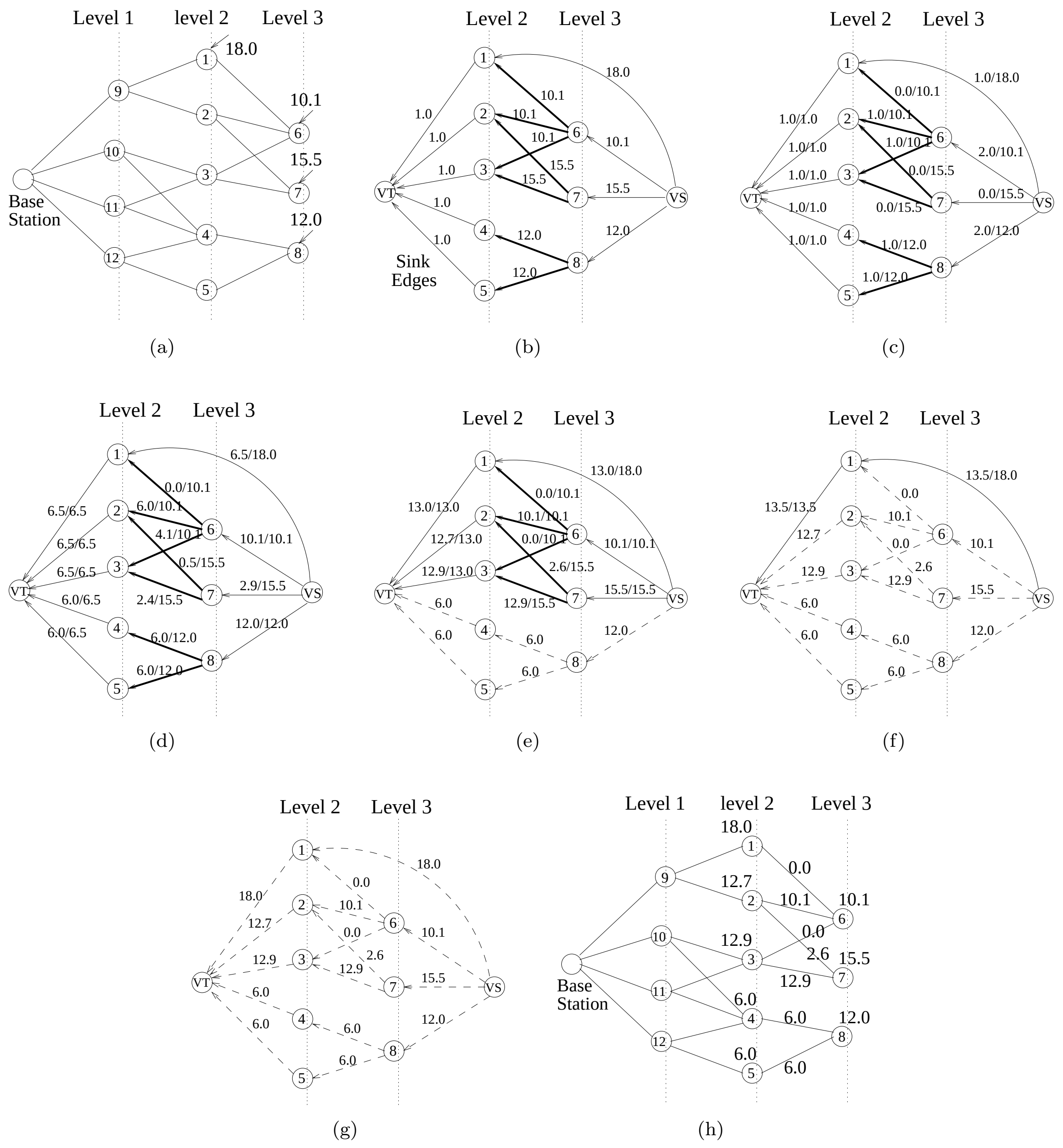


Figure 3: *(a): A RN with 4 sources. (b): A flow network is constructed to balance loads at level 2. Edges are labeled with capacities. (c): The resulting flow after solving the maximum flow for the network in (b). (d): The resulting flow after the sink-edge capacities are increased to 6.0. (e): Sink-edge capacities reach 13.0. The dashed edges indicate these edges and the corresponding nodes are removed from the network. The labels on them are the final loads. (f): Node 2, 3, 4, 5 all obtained their final loads. (g): The final flow. (h): The final traffic distribution.*

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 similarly.

4.3 Complexity Analysis

Since the time complexity for solving the maximum flow is $O(V^3)$ (Cormen et al., 1997), the time complexity for LBR is $O(\frac{L}{\eta}V^3)$, where L is the total traffic load to be balanced, and V is the total number of nodes at the two levels. The larger the η is, the faster our algorithm is. However, increasing η causes the resulting loads deviate more from the optimal, since η is the unit of increment, and bounds the deviation of the final loads from the optimal. A smaller η leads to better accuracy but also to more computation and power consumption at the sensor nodes. For example, in Fig. 3(e), the final loads on Node 2 and 3 are 12.9 and 12.7, respctively, while the *optimal* load is 12.8 each. The deviation is 0.1 (less than $\eta = 0.5$). Thus, we must choose η carefully to balance the algorithm's accuracy and efficiency. In our experiments, we chose a moderate value for η.

Our algorithm requires space $O(V^2)$ for constructing the flow network. Since V is small compared to the total number of nodes in the network, this space requirement is reasonable.

4.4 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 4.2.1).

We propagate the loads to other nodes at the same level as follows: Each node is set to work in *promiscuous* mode (802.11) so that it can overhear packets sent by its neighbours. Let the *siblings* of node s to be the nodes at the same level 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 a node is not reachable from s at levels i and $i-1$, the traffic loads of these two nodes can be scheduled independently, as with nodes 6 and 8 in Fig. 3.

For example, in Fig. 3(a), when node 6 needs to propagate an updated load value, it piggybacks the value on its regular data packets which are sent to its parents, Node 1, 2, and 3. When node 2 sends the packet a level down, node 7 can overhear it and obtain the load value.

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 (level number and parents) to its siblings through a similar mechanism. Since sensor nodes are static and it is less frequent to add or remove nodes in typical applications, the topology information only needs to be exchanged initially, and updated once in a while. As discussed in Section 4.3, the computation cost for our scheme is moderate. Thus we can ignore the power consumption incurred by computation.

5 Packet Aggregation

All packets are destined for the same 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 of grouping more sensor readings into a packet. However, a sensor reading can only be held for a time determined by its deadline. An important question for us to consider is: Given the deadline for individual sensor readings, how do we 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?

5.1 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 (see Section 3). 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. Now, 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 the 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 5.2. If $\delta_i(u_j,t) \le 0$, u_j must be forwarded to the BS 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 5.1.1, we provide an upper bound for u_j's hold time on each relaying node. In Section 5.1.2, we discuss how u_j's hold time on S_i will affact its outgoing packet rate, based on which we formulate an optimization problem in Section 5.1.3 that minimize 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.

5.1.1 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,

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

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.

5.1.2 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 the 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 (Kleinrock, 1975), $r_i h_i(u_j)$ 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 $\frac{r_i}{r_i \cdot h_i(u_j)+1}$.

5.1.3 Obtaining the Hold Times

If u_j governs the relaying nodes' outgoing packet rates, 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 (Ye et al., 2002), 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* (Bertsekas, 1996) 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 5.2). The $\{h'_i\}$ value thus obtained is conservative, but incurs little overhead. We show in Section 6.3.1 that our approach performs far better than other schemes, and is easy to implement.

5.2 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 He et al. (2003). 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 delays may vary because of wireless channel variability or changes in packet rates, but we would like the variance of the 1-hop delay at each node to be low. Our simulation results show that this is indeed the case. We randomly deployed 100 nodes in a $1500 \times 1500\, m^2$ area with the BS at the center (see other parameters in Table 1). Each node generated data at the same rate. In Fig. 4, we show the *relative standard deviation (RSD)* for 1-hop delays under various source generation rates. The deadline for each sensor reading is set to be *df* times the maximum E2E delay from the source. As can be seen, both our load balancing scheme and packet aggregation scheme contribute to lowering delay variations.

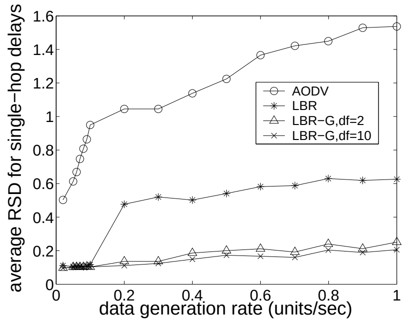


Figure 4: RSD for 1-hop delays. LBR-G denotes our scheme with both LBR and packet aggregation.

The 1-hop delay must still 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.

5.3 The Algorithm

Algorithm 1 shows how sensor readings are grouped at each node. The sensor readings are extracted from received packets, queued, and a timer for each sensor reading u_i is set to expire after the hold time $h(u_i)$. Grouping continues as long as the payload size is smaller than M. If the timer for any sensor reading expires, we send the packet out immediately. We re-evaluate each sensor reading's remaining time to the deadline $\tau(u_i)$ before delivering the packet.

Algorithm 1 Packet Aggregation Scheme at node S

```
$h(u_i)$: the permissible hold time for sensor reading $u_i$ on node $S$.
$\tau(u_i)$: the time remaining to $u_i$'s deadline on arriving on $S$.
$WQueue$: the waiting queue for the sensor readings.
loop
  if an incoming packet $pt_{in}$ is received then
    /* $pt_{in}$ contains multiple sensor readings. */
    Extract sensor readings $u_1$, $u_2$, ..., $u_n$ from $pt_{in}$;
    for $i = 1$ to $n$ do
      /* Set a timer for each sensor reading. */
      Update $\tau(u_i)$, $u_i$'s remaining time to the deadline;
      Calculate $h'(u_i)$, $h''(u_i)$ and $h'''(u_i)$;
      $h(u_i) = \min\{h'(u_i), h''(u_i), h'''(u_i)\}$;
      $u_i.timer = t_{curr} + h(u_i)$;
      Enqueue $u_i$ into $WQueue$;

  if sensor reading $u_j$'s timer expires then
    Remove $u_j$ from $WQueue$;
    Create an outgoing packet $pt_{out}$ for $u_j$;
    while $sizeof(pt_{out}) < M$ and $WQueue$ is not empty do
      Dequeue a sensor reading $u_k$ from $WQueue$;
      Re-evaluate $\tau(u_k)$ and group it into $pt_{out}$;
    Send packet $pt_{out}$ to the outgoing link according the link probability;
```

6 Experiments

We evaluated our approach through extensive simulations on the ns-2 simulator (ns2). In Section 6.1, we describe our simulation setup. In Section 6.2, we show results for our LBR scheme. The results for our packet aggregation mechanism (with LBR) are shown in Section 6.3.

6.1 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 nodes and 150 nodes, with a subset of the sensor nodes selected as *sources* that generated sensor readings periodically. The generation rates were varied to reflect different workloads. All sensor generated data were delivered to the BS.

We chose 802.11 (802.11) as our MAC layer protocol instead of 802.15.4 (802.15.4). The 802.15.4 protocol is intended for low-rate wireless personal area networks with low complexity and low power. However, we consider real-time sensor networks with high generation rates and stringent deadlines (see Section 6.2), which makes 802.15.4 unsuitable. Among other issues, 802.15.4 lacks the RTS/CTS mechanism, so that interference is a big issue as the load

Table 1: Some simulation parameters

Parameters	Default Values
Transport layer protocol	UDP
Sensor's outgoing queue size	50
Payload size for each sensor reading	8 bytes
Maximum payload size	128 bytes
MAC protocol	802.11
Radio propagation model	Shadowing
Radio communication/interference range	$250m/550m$
Transmit/receive power	$31.2mW/22.2mW$
Sensor node placement	Uniform

grows. It is known that 802.15.4 delivers a much lower fraction of packets to the receiver than 802.11 (Zheng and Lee, 2004). However, a high packet delivery ratio is crucial to guaranteeing that packets will meet deadlines. Zheng and Lee (2004) performs extensive comparisons between 802.11 and 802.15.4, and the results showed that the packet delivery ratio drops rapidly as the traffic load increases under 802.15.4, while can remain at higher levels in 802.11. The 802.11 MAC protocol was also used in other real-time sensor networks, such as RAP (Lu et al., 2002) and SPEED He et al. (2003).

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. The ns-2 simulator currently supports three propagation models, among which the *shadowing* model is more general and widely-used (Rappaport, 1996). The shadowing model consists of two parts: a path loss model, and a statistical model which reflects the variation of reception at certain distance. Nodes can only communicate under this probabilistic model when near the edge of the communication range. In our simulations, we set the value of the path loss exponent as 2.0, and the value of the 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. Table 1 shows the parameters for our simulations.

We used the AODV routing protocol as the basis for our comparisons. For each source, AODV discovers a single fixed route to the BS. Our results show that our load balanced routing scheme coupled with packet aggregation significantly outperforms this traditional single-path routing scheme.

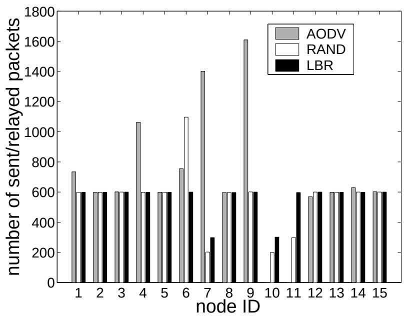
To measure power, we adopted the power parameters from the *Chipcon* CC1000 RF transceiver (Chipcon), which is used as the radio module in both MICA2 and MICA2DOT (Mica) sensor models. When operated at $433MHz$, its receiving power is $22.2mW$, and the transmitting power is $31.2mW$, with the output power of $0dBm$. In the ns-2 power model, each node was set to the same power level initially, and we measured the remaining power after the simulation ran for some time.

6.2 Performance of LBR

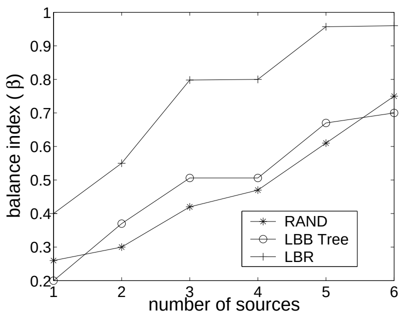
We first evaluate how well we can balance traffic using our LBR scheme (see Section 4).

6.2.1 Load Balancing

We first simulated LBR on the sensor network shown in Fig. 2 for 300 seconds. Fig. 5(a) shows the traffic load distribution for all nodes. Six *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, packets take random routes to the BS. Each node randomly picks an outgoing link in the RN with equal probability to send/relay a packet. Although RAND randomly distributes loads, it does not aim to balance the loads as evenly as possible. 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 still relays heavy traffic under RAND. Traffic loads are very balanced in LBR.



(a) Load distribution



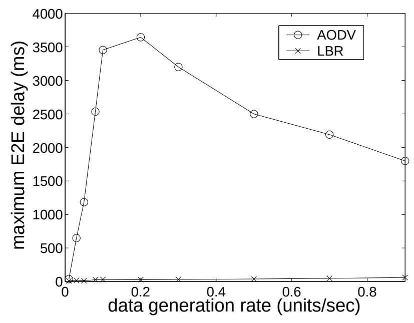
(b) β value for level-1 nodes

Figure 5: LBR performance for network in Fig. 2.

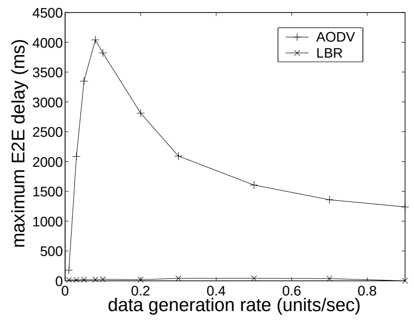
Fig. 5(b) compares LBR with RAND and LBB-Tree, another load balanced routing scheme by Hsiao et al. (2001), on the same topology. We adopted the *Balance Index* (β) metric (Hsiao et al., 2001), which measures the degree of load balancing among a given set of nodes:

$$\beta(S) = \frac{(\sum_{s_i \in S} f_i)^2}{q \sum_{s_i \in S} f_i^2}, \qquad (11)$$

where S is the given set of nodes, f_i is the load on node s_i and q is the total number of nodes in S, $0 < \beta(S) \leq 1$.



(a) 100 sensor nodes



(b) 150 sensor nodes

Figure 6: Average of maximum E2E delays (without packet aggregation)

Generally, the higher $\beta(S)$ is, the more balanced the loads are. The loads among nodes in S are perfectly balanced if $\beta(S) = 1$.

In Fig. 5(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. Generally, the β values 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.

6.2.2 E2E Delays

Fig. 6 illustrates the maximum E2E delays from packet sources to the BS for different node densities. The maximum E2E delay for a given sensor network is the longest delay experienced by packets from a farthest source or via a highly congested route, 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 Fig. 6 shows, 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. 8). 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.

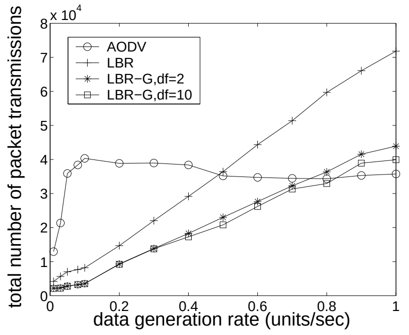


Figure 7: Number of packets compared (100 nodes, 350 secs)

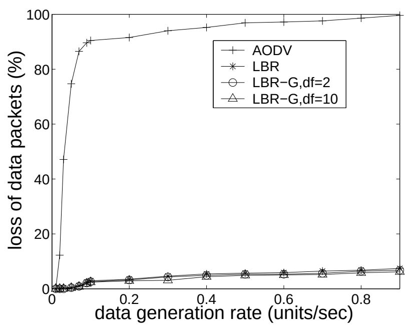


Figure 8: Packet loss compared (100 nodes, 350 secs)

6.3 Performance of Packet Aggregation

Having verified the benefits of LBR, we next evaluated how the packet aggregation mechanism further reduces transmissions. We ran the simulations for 350 seconds on a topology of 100 nodes. Fig. 7 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. 8.

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.

6.3.1 Allocating Hold Times

Our hold time allocation algorithm (see Section 5.1) distributes the slack times for each packet across the relaying nodes to minimize the overall number of transmissions. Fig. 9 compares our algorithm (LBR-G) with two other hold time allocation approaches, namely the *UNIFORM* and the *SRC* scheme, under two *df* values. 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.

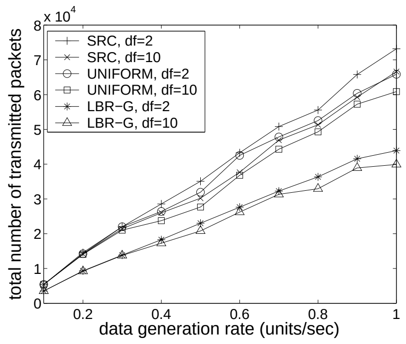


Figure 9: Hold time allocation schemes compared (100 nodes, 350 secs)

We used a topology of 100 nodes and ran the simulation for 350 seconds. Each node generated data at the same rate, shown on the x-axis. 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 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.

6.3.2 Meeting Real-time Requirements

Fig. 10 shows the fraction of packets missing their deadlines under different deadline requirements. We compare our scheme with AODV and RAP (Lu et al., 2002). RAP's *velocity monotonic (VM)* scheduling mechanism prioritizes the real-time packets at each node based on each packet's deadline and the source-BS distance. Two versions of the VM algorithm were proposed in RAP: the *static velocity monotonic (SVM)* and the *dynamic velocity monotonic (DVM)*. Since it was reported by Lu et al. (2002) that SVM performs better than DVM, we implemented SVM for RAP. Since RAP is independent of the underlying routing protocol, we simulated RAP on both AODV and LBR.

Each data point in Fig. 10 was obtained by running the simulation on a network of 100 nodes for 350 seconds, and associating each sensor reading with the same end-to-end deadline. Each node generated readings at the rate of 0.5 units/sec. When the deadlines are so stringent that the slack time is negative, most packets will miss their deadlines under all schemes, and LBR-G reduces to LBR. However, the miss ratio drops rapidly for LBR-G as the deadlines become less stringent, and LBR-G always has a lower miss ratio than both AODV and RAP. We also observe that RAP/LBR achieves much lower miss ratio than RAP/AODV does, showing that our load-balanced routing scheme contributes to lower end-to-end delays significantly and avoid bottleneck nodes.

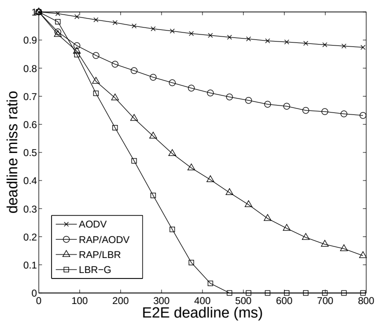


Figure 10: Deadline miss ratio (Data rate 0.5 $units/sec$)

Fig. 11 shows the deadline miss ratio under different data generation rates. Each node generated readings at the same rate, and all sensor readings were associated with the same deadline of 200ms. Under both schemes, the deadline miss ratio increases as the data generation rate is increased to 0.2 units/sec. However, LBR-G incurs lower miss ratio than RAP, especially under heavy traffic loads. This effect is better appreciated in Fig. 12, which shows the percentage of packets lost under different data generation rates. Clearly, the packet loss ratio is much lower in LBR-G especially under heavy data traffic, suggesting that more data packets can reach the BS in LBR-G, thereby lowering the deadline miss ratio.

6.3.3 Power Savings

We first measured the number of bytes transmitted at the MAC layer. MAC-level traffic is a better estimate of power consumption than the number of application-level packets, since each application packet can cause many MAC-level packets due to channel contention. Fig. 13 shows the total number of MAC bytes, including packet transmission, retransmission, and RTS/CTS/ACK messages, for a network of 100 nodes. Clearly, LBR-G incurs fewer transmissions than AODV. As in Fig. 7, the number of MAC transmissions levels off under AODV after a certain rate because the

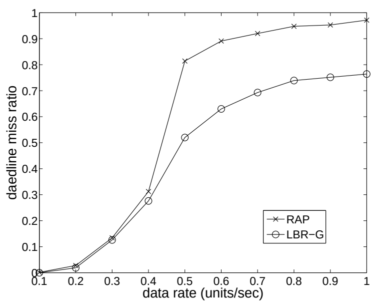


Figure 11: E2E deadline miss ratio (Deadline 200ms)

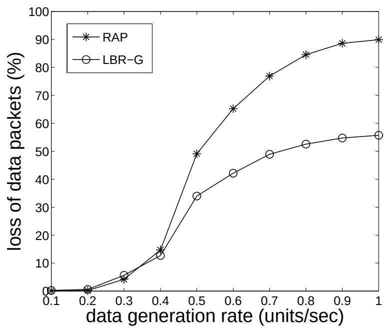


Figure 12: Packet loss ratio (Deadline 200ms)

network is saturated and starts dropping packets. Fig. 13 suggests that our scheme is more power efficient for moderate traffic rates, and achieves better throughput at high traffic rates.

We also directly measured power consumption at each node using the power model available in the ns-2 simulator. Fig. 14 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 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 (see Fig. 13). Second, many packets fail to meet their deadlines under AODV than under LBR-G (see Fig. 10).

Extending the network lifetime is an important goal in real-time sensor networks. Fig. 15 compares the fraction of surviving nodes under AODV, LBR, and LBR-G for both 100-node and 150-node networks, after a given simulation time. The initial energy level was set to 5 J. The deadline for each reading was set to twice the average end-to-end delay ($df = 2$). For the 100-node network, the survivor fraction starts to drop rapidly at 300 sec under AODV, while it remains 100% under both LBR and LBR-G. Our LBR scheme achieves 150% longer network lifetime than AODV, while our LBR-G scheme achieves network lifetime 50% longer than LBR. It is clear that both the load-balanced routing scheme and the packet aggregation scheme contribute significantly to extending network lifetime. We also observe that the 100-node network has longer lifetime than

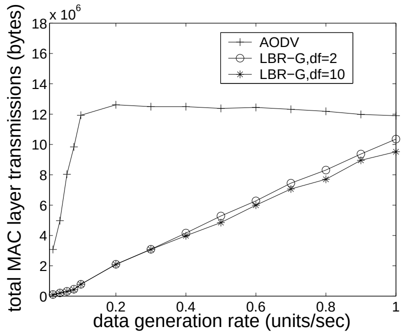


Figure 13: MAC layer data transmissions compared (100 nodes, 700 secs)

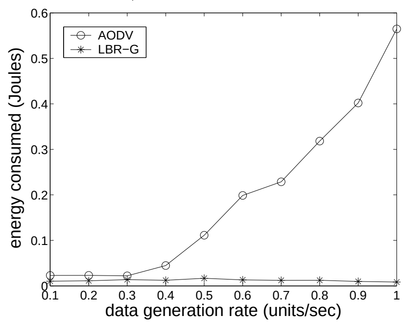


Figure 14: Energy Consumed per on-time sensor reading (100 nodes, simulation time: 350 $secs$, deadline: 500 ms)

the 150-node network under all schemes. All nodes generate data in our simulations, so the traffic load is much higher in the 150-node network.

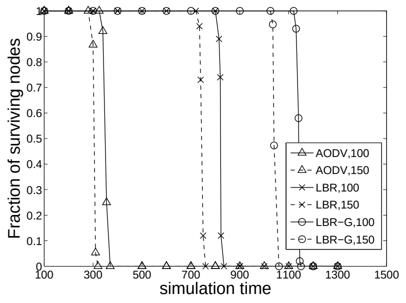


Figure 15: Fraction of surviving nodes (df=2)

7 Conclusions

We presented *PERT*, 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 transmissions. 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.

We conducted extensive experiments using ns-2 to evaluate the performance of our approach. Our 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. When the data traffic rate is high in the network, our scheme can achieve better throughput. PERT also outperforms RAP, another real-time packet delivery scheme for sensor networks.

We next plan to study how to construct power-efficient in-network data aggregation trees (Madden et al., 2002) for real-time traffic.

ACKNOWLEDGMENT

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

REFERENCES

Chipcon CC1000 RF transceiver datasheet. http://www.chipcon.com. April, 2004.

The network simulator ns-2. http://www.isi.edu/nsnam/ns/.

Redwoods now part of wireless network. http://www.cnn.com/2003/TECH/science/08/15/coolsc.redwoods.

MPR/MIB mote sensor hardware users manual. http://www.xbow.com/Support/manuals.

Warfighter physiological status monitoring. http://www.usariem.army. mil/wpsm.

Wireless medium access control (MAC) and physical layer (PHY) specifications for low-rate wireless personal area networks (LR-WPAN). In *IEEE 802.15.4 Standard, 2003 Edition.*

Wireless LAN medium access control (MAC) and physical layer (PHY) specifications. In *IEEE 802.11 Standard, 1997 Edition.*

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

Badrinath, B. R. and Sudame, P. (2000). 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.*

Bertsekas, D. P. (1996). *Constrained Optimization and Lagrange Multiplier methods.* Belmont, Mass. : Athena Scientific.

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

Carley, T. W., Ba, M. A., Barua, R., and Steward, D. B. (2003). Contention-free periodic message scheduler medium access control in wireless sensor/actuator networks. In *Proc. of the 24th IEEE Real-Time Systems Symposium*, Cancun, Mexico.

Chen, B., Jamieson, K., Balakrishnan, H., and Morris, R. (2001). Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. In *Proc. of the MobiCom Conf*, Rome, Italy.

Chipara, O., Lu, C., and Roman, G.-C. (2005). Efficient power management based on application timing semantics for wireless sensor networks (to appear). In *Proc. of the 25th ICDCS.*

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

Feeney, L. M. and Nilsson, M. (2001). Investigating the energy consumption of a wireless network interface in an ad hoc networking environment. In *Proc. of the 20th IEEE INFOCOM.*

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

He, T., Stankovic, J. A., Lu, C., and Abdelzaher, T. (2003). 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.

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

Hill, J., Szewczyk, R., Woo, A., Hollar, S., Culler, D., and Pister, K. (2000). System architecture directions for networked sensors. In *Proc. of the ASPLOS.*

Hong, X., Gerla, M., Hanbiao, W., and Clare, L. (2002). Load balanced, energy-aware communications for mars sensor networks. In *Proc. of the Aerospace Conference, vol 3.*

Hsiao, P. H., Hwang, A., H.T.Kung, and Vlah, D. (2001). Load-balancing routing for wireless access networks. In *Proc. of the 20th IEEE INFOCOM*, Anchorage, Alaska.

Huang, S. C. and Jan, R. H. (2004). Energy-aware load balanced routing schemes for sensor networks. In *Proc. of the 10th Intl Conference on Parallel and Distributed Systems*, Newport Beach, California.

Intanagonwiwat, C., Govindan, R., and Estrin, D. (2000). Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *Proc. of MobiCom.*

Johnson, D. B. and Maltz, D. A. (1996). Dynamic source routing in ad hoc wireless networks. In *Mobile Computing*, volume 353. Kluwer Academic Publishers.

Kleinrock, L. (1975). *Queueing Systems: Theory, Volume 1.* John Wiley and Sons.

Liu, X., Wang, Q., Sha, L., and He, W. (2003). Optimal qos sampling frequency assignment for real-time wireless sensor networks. In *Proc. of the 24th IEEE Real-Time Systems Symposium (RTSS)*, Cancun, Mexico.

Lu, C., Blum, B. M., Abdelzaher, T. F., John A, S., and He, T. (2002). RAP: A real-time communication architecture for large-scale wireless sensor networks. In *Proc. of the 8th IEEE Real-Time and Embedded Techonology and Application Symposium (RTAS)*, San Jose, CA.

Madden, S. and Franklin, M. J. (2002). Fjording the stream: An architecture for queries over streaming sensor data. In *Proc. of the 18th ICDE Conf*, San Jose.

Madden, S., Franklin, M. J., Hellerstein, J. M., and Hong, W. (2002). TAG: A tiny aggregation service for ad-hoc sensor networks. In *Proc. of OSDI*.

Madden, S., Franklin, M. J., Hellerstein, J. M., and Hong, W. (2003). The design of an acquisitional query processor for sensor networks. In *Proc. of the ACM SIGMOD'03*.

Mainwaring, A., Polastre, J., Szewczyk, R., Culler, D., and Anderson, J. (2002). Wireless sensor networks for habitat monitoring. In *Proc. of ACM Workshop on Wireless Sensor Networks and Applications*.

Manjhi, A., Nath, S., and Gibbons, P. B. (2005). Tributaries and deltas: Efficient and robust aggregation in sensor network streams. In *Proc. of the ACM SIGMOD*.

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

Perkins, C. E. and Royer, E. M. (1999). Ad hoc on-demand distance vector routing. In *Proc. of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, New Orleans, LA.

Rappaport, T. S. (1996). *Wireless Communications, Principles and Practice*. Prentice Hall.

Sharaf, M. A., Beaver, J., Labrinidis, A., and Chrysanthis, P. K. (2003). TiNA: A scheme for temporal coherency-aware in-network aggregation. In *Proc. of the 3rd ACM MobiDE Workshop*.

Xu, Y., Heidemann, J., and Estrin, D. (2001). Geography-informed energy conservation for ad hoc networks. In *Proc. of the MobiCom Conf*, Italy.

Ye, W., Heidemann, J., and Estrin, D. (2002). An energy-efficient mac protocol for wireless sensor networks. In *Proc. of the 21st INFOCOM*.

Zheng, J. and Lee, M. J. (2004). A comprehensive performance study of ieee 802.15.4. In *IEEE Press Book*.