OPAG: Opportunistic Data Aggregation in Wireless Sensor Networks

Zhigang Chen and Kang G. Shin
Real-Time Computing Laboratory,EECS Department
The University of Michigan
Ann Arbor, MI 48109-2121
Email: {zhigangc,kgshin}@eecs.umich.edu

Abstract

We propose **Opportunistic Data Aggregation** *(OPAG) that incurs no computation error and tolerates moderate message losses in wireless sensor networks (WSNs). OPAG performs in-network data aggregation in two layers: (1) at the data-aggregation layer, aggregation results are computed accurately; and (2) at the data-routing layer, a WSN node may send intermediate/partial results to its aggregation node using multi-path routing in order to tolerate message losses.*

By space-multiplexing messages (i.e., padding multiple partial results or sensor readings in a single message), OPAG opportunistically exploits a multi-path routing scheme, which is more energy-efficient than retransmission. This is based on a key observation that, when sending a message, the radio may consume much more energy in idle listening during the backoff period and the time to wait for its acknowledgment than transmitting the data bits. We implemented OPAG on TinyOS 2.x and TMote Sky Mote, and evaluated its performance on the Motelab Testbed at Harvard University. When a network is relatively well connected, OPAG can reduce the energy consumption by 33% at the expense of slightly higher relative errors, compared to TAG with reliable transmission. Compared to Synopsis Diffusion, OPAG can reduce the aggregation errors by 50% while consuming roughly the same amount of energy.

1 Introduction

In a large-scale wireless sensor network (WSN), data aggregation (e.g., averaging temperature readings across a network's coverage area) is crucial for such applications as environment monitoring and surveillance. Previous studies (e.g., [15]) show that *in-network* data aggregation — intermediate nodes compute partial aggregation results and propagate them towards the base station (BS) or the data sink — is significantly more efficient in terms of communication cost and energy consumption than routing all sensor readings to the BS which then processes them.

Data-aggregation accuracy in a WSN is affected by the fidelity of both computation and communication within the network. First, computation of aggregation results may be exact or approximate, incurring zero or non-trivial computation error. Second, intermediate aggregation results may be routed via a spanning tree or multiple paths, resulting in different degrees of tolerance to message loss. Figure 1 shows different data-aggregation schemes in the design space.

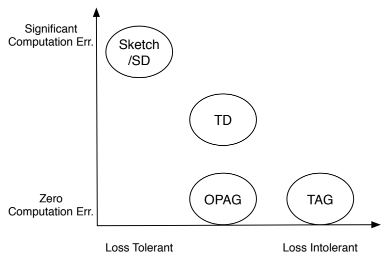


Figure 1. Design space of in-network data aggregation with respect to computation error and tolerance of data message loss. OPAG aims to achieve zero computation error and high tolerance to moderate message loss.

On one side of design space, TAG [14, 15] and Cougar [21] construct a spanning tree rooted at the BS. Each node of the tree accurately computes partial aggregation results and then forwards them to its parent in the tree. This spanning tree scheme is simple, but susceptible to data loss during communication, because losing a data message over a link leads to the loss of all sensor readings or partial aggregation results from the sub-tree below the link, thus possibly resulting in significant loss of aggregation accuracy. In order to reduce data loss, each node may dynamically select appropriate parent nodes to avoid using poor-quality links [15], and retransmit lost messages at the expense of much more energy consumption.

On the other side of design space, Sketch [5], Synopsis

Diffusion (SD) [17], and Tributary-Delta (TD) [16] aggressively exploit multi-path routing to combat message losses. A sensor reading may be aggregated with other readings along multiple paths before it reaches the BS, in contrast to the case of using just one path on the spanning tree. Because of high communication redundancy, Sketch and SD are highly loss-tolerant. However, their multi-path routing is *uncontrolled*—each node has no or little control of which nodes get to aggregate its data, and therefore, a partial aggregation result may be aggregated multiple times into the final result. To deal with duplicate-sensitive aggregates, such as COUNT, SUM, and AVG, statistical counting [13] is used to encode a partial result into a bitmap (a.k.a. Sketch or SD), and convert duplicate-sensitive aggregates to duplication-insensitive *OR* of such bitmaps. The bitmap obtained at the BS provides an estimate of the aggregation result. The estimation error is non-negligible, and in particular, the variation of estimation is quite high. Compared to other schemes, Sketch and SD work well under a very poor network condition. However, they do not work well for networks under a relatively good condition—which are more common in real-world—because the estimation error persists irrespective of the network condition.

We would like to develop a data-aggregation scheme with zero computation error and good tolerance to moderate message losses because most of real-world networks have been designed and deployed to operate under reasonably good conditions. Moreover, the network condition may be improved by adding or upgrading hardware, dynamically switching to channels with less interference, or even altering the deployment. A majority of wireless links are reported to have 0.8 or better delivery probability in both sensor networks [19] and wireless mesh networks [1]. As elaborated on in Section 4, our experience with Motelab [20], the WSN testbed at Harvard University, also confirms this observation.

Opportunistic Data Aggregation (OPAG) is a new data-aggregation scheme designed to take advantage of relatively good network connectivity, wherever possible. OPAG allows a node to autonomously choose a *Data-Aggregation Node* (DAN) within a few hops of itself. Then, if the successful delivery ratio over multiple paths to the DAN is above a given threshold, the node has the DAN aggregate its partial results. The DAN aggregates the partial results accurately, so as to avoid computation errors, while the other nodes on the multiple paths just relay it. Such *controlled* multi-path routing is likely to be as loss-tolerant as the *uncontrolled* under a relatively good network condition, although it does not provide as much path redundancy. If there is not enough path redundancy to provide a satisfactory success ratio, the node sends the partial results to its parent node as in TAG, and compensates communication loss by retransmitting them.

OPAG *opportunistically* uses multi-path routing to compensate communication losses and achieve better energy-efficiency than other schemes using retransmission. This is attributed to a key observation that, when sending a message, the radio (e.g., CC2420, a widely-used, low-power, and high-speed radio) may consume much more energy in idle listening during the backoff period and the time of awaiting the acknowledgment than transmitting the bits. Retransmitting a message is not energy-efficient because it incurs more idle listening on backoff and more time of waiting for the acknowledgment.

In order to avoid extra idle listening, OPAG uses multi-path routing that differs from traditional multi-path routing. Each node dynamically pads multiple partial results and/or sensor readings in a message, i.e., a message may carry multiple partial results each of which traverses a different set of paths. Every receiver disassembles the message and processes the partial results separately—it may aggregate, forward, or discard a partial result, depending on the specified DAN.

This paper makes the following contributions.

- *Development of a new data-aggregation approach, OPAG*. OPAG incurs zero computation error and provides good tolerance to moderate message losses. It opportunistically uses the multi-path routing to compensate communication losses and achieve better energy-efficiency than those using retransmission.

- *Prototype implementation and experimentation on a real-world testbed*. We implemented OPAG on TMote Sky motes [18] and conducted experiments on Motelab [20]. The experimental results show OPAG performs much better than TAG and Sketch/SD under relatively good network connectivity.

The paper is organized as follows. Section 2 illustrates the basic idea of OPAG with an example network. Section 3 presents the detailed design of OPAG. Section 4 describes our prototype implementation and presents the experimental results on Motelab. Section 5 discusses the related work. Finally, Section 6 concludes the paper.

2 Basic Idea

OPAG divides in-network data aggregation into two layers: data aggregation and data routing. At the data-aggregation layer, the aggregation results are computed along an overlay spanning tree; underneath the routing layer, network nodes may opportunistically send intermediate/partial results via multi-path routing.

Figure 2 shows an illustrative example. At the routing layer, the solid edges in the figure indicate child–parent relationships in the spanning tree. Besides the parent node,

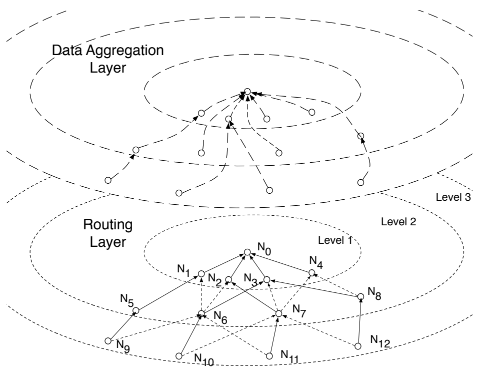


Figure 2. Each edge (e.g., $N_{10} \rightarrow N_2$)in the overlay tree shown at the aggregation layer may correspond to multiple paths at the routing layer ($N_{10} \rightarrow N_6 \rightarrow N_2$ and $N_{10} \rightarrow N_7 \rightarrow N_2$).

each node may also communicate with the other nodes at a lower level in the tree, as indicated by the dotted edges. At the data-aggregation layer, each link in the overlay tree corresponds to a child–parent link or the multiple paths from this node to its DAN.

Suppose N_{10} selects N_2 as its DAN. There are two paths from N_{10} to N_2: $N_{10} \rightarrow N_6 \rightarrow N_2$ and $N_{10} \rightarrow N_7 \rightarrow N_2$. Therefore, N_2 may receive two copies of N_{10}'s data. The multi-path routing uses this data redundancy to combat message losses along either of the two paths. In its partial results, N_{10} specifies N_2 as its DAN. After N_6 receives N_{10}'s message, it extracts the partial results, because the message may contain other partial results forwarded by N_{10} (in this example, we assume N_{10}'s message only contains its own result). Likewise, N_6 receives the data messages from N_9 and N_{11}. N_6 checks each partial result it receives, and aggregates those specifying N_6 as the DAN. Then, N_6 sends the partial results that it should forward as well as its own — N_{10}, N_{11}, and N_6. It discards the partial result of N_9 as N_9's parent node N_5 is to aggregate its partial result. N_9 does not choose N_1 or N_0 to aggregate its partial results because the multiple paths from N_9 to N_1 or N_0 do not provide a satisfactory successful delivery ratio.

After N_2 receives N_6's message, it aggregates the data from N_{10} and N_{11} with its own, because these two data entries specify N_2 as the DAN. Then, N_2's message contains the data of N_6, N_7, as well as the intermediate result of aggregating its own reading and the data from N_{10} and N_{11}. Duplicate data are ignored by N_2.

Because of message space multiplexing, each partial result has more opportunities to reach a specified aggregation node. Thus, OPAG can tolerate message losses without incurring any computation error.

3 Design of OPAG

In this section, we first describe how OPAG selects DANs, then analyze the energy consumption of data aggregation using multi-path routing and retransmission, and finally discuss the protocol overhead and limitation.

3.1 Selecting Data-Aggregation Node

So far, we have described how a partial result is routed to, and aggregated at a DAN. We now describe how each node selects its DAN. Each node maintains a small list of DAN candidates. A candidate entry contains four attributes: <*id*>, <*level*>, <*p*>, and <*flist*>. The attributes <*id*> and <*level*> are the ID and the level of the candidate in the spanning tree below it, respectively. <*p*> is the probability that a candidate node's data will be successfully delivered to the data sink via a routing path, and <*p*> is recursively computed, as we will describe later. <*flist*> is a list of forwarding entries, each of which has two fields, <*fid*> and <p_f>. The former is the ID of a neighbor which can forward this node's partial results to the aggregation candidate, and <p_f> is the probability that the neighbor successfully sends partial results to the candidate.

DAN lists are maintained via DAN announcements. A node N's DAN announcement contains a few DAN candidates that it wants to advertise. Each entry in the announcement has <*id*>, <*level*>, <*p*>, and <$p\prime_f$>. The first three fields are from the corresponding DAN entry, and <$p\prime_f$> is computed from the forwarding list as $p\prime_f = 1 - \prod_{F \in flist} (1 - p_f^F \cdot p(N,F))$. F is a forwarding node in <*flist*>, p_f^F is from <p_f> of F, and $p(N,F)$ is the link quality from N to F, which is maintained in N's neighbor table.

If N selects M as its DAN,

$$\begin{aligned} p &= p(M) \cdot p\prime_f(M) \\ &= p(M) \cdot (1 - \prod_{F \in M's flist} (1 - p_f^F(M) \cdot p(N,F))) \end{aligned}$$

where $p(M)$ and $p_f^F(M)$ are provided by M's DAN entry, $p(N,F)$ is from the link-quality information N maintains. Naturally, N picks a DAN candidate, M, by maximizing p.

We use the example in Figure 2 to illustrate how each node maintains its DAN list and selects its DAN. Assume $p(N_1,N_0) = 0.9$, $p(N_2,N_0) = 0.9$, $p(N_3,N_0) = 0.8$, $p(N_6,N_1) = 0.8$, $p(N_6,N_2) = 0.8$, $p(N_6,N_3) = 0.9$.

The data sink (i.e., N_0) broadcasts an announcement with $< 0,0,1.0,1.0 >$. <*p*> is 1.0 because N_0 is the data sink, and <$p\prime_f$> is 1.0, because the sender is advertising itself. Upon receiving the base station's announcement, N_1 inserts $< 0,0,1.0,\{< 0,1.0 >\} >$ into its candidate list.

N_1 picks N_0 as its DAN, so the probability that a partial result of N_1 is successfully delivered to N_0 is $p(N_1) = p(N_0) \cdot p_f(N_0) \cdot p(N_1, N_0) = 0.9$, where $p(N_0)$ and $p_f(N_0$ are N_0's entry, and $p(N_1, N_0)$ is from N_1's neighbor table.

Then, N_1 may advertise two entries: $< 0, 0, 1.0, 0.9 >$, and $< 1, 1, 0.9, 1.0 >$. The first entry is to re-advertise N_0, so N_1 simply copies the three fields from N_0's DAN entry — <*id*>, <*level*>, and <*p*>. It also sets <p_f> to $p(N_1, N_0) = 0.9$ because N_1 is considered as the forwarding node. The second entry is to advertise itself with <*p*> as 0.9 (i.e. $p(N_1) = 0.9$) and <$p\prime_f$> as 1.0.

Similarly, N_6 has four candidates: $< 0, 0, 1.0, \{< 1, 0.9 >, < 2, 0.9 >, < 3, 0.8 >\} >$, $< 1, 1, 0.9, \{< 1, 1.0 >\} >$, $< 2, 1, 0.9, \{< 2, 1.0 >\} >$, and $< 3, 1, 0.8, \{< 3, 1.0 >\} >$. In particular, the first entry indicates that N_6 can send partial results to N_0 through three paths.

The probability that N_6's partial results are successfully delivered via these multiple paths to N_0 is $p\prime_f = 1 - \prod_{F \in flist} (1 - p_f^F \cdot p(N, F)) = 1 - (1 - 0.9 * 0.8) * (1 - 0.9 * 0.8) * (1 - 0.8 * 0.9) = 0.98$. So, if N_6 selects N_0 as the DAN, the probability that its partial results are successfully delivered to the sink is $p = p\prime_f * p(N_0) = 0.98$.

Each node should only consider the DAN announcements from other nodes of a lower tree level, and ignore the announcements from nodes of the same or higher level, because only a lower-level node can be its DAN. For example, N_1's announcement should be processed by N_9, but ignored by N_2. Moreover, a DAN announcement entry propagates upward only a few hops to control storage and communication overheads. For instance, if the hop limit is 3, N_9 does not re-advertise N_0 in its announcements.

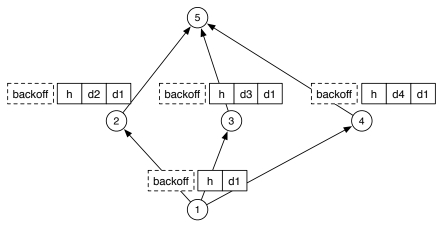
3.2 Energy Cost of Multi-Path Routing and Message Retransmission

Both the multi-path routing and retransmission require the transmission of more bits to tolerate communication losses. So, a natural question is: which one is more energy-efficient?

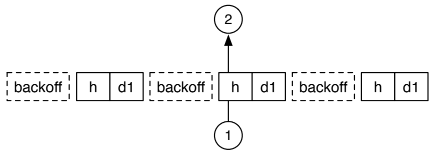
We use the following example to compare the energy consumption of these two choices. Figures 3(a) and 3(b) show how the partial results are transmitted using the multi-path routing and retransmission, respectively.

First, we consider a receiver's energy consumption. The receiver turns on its radio at the beginning of the assigned receiving slot, and sets the radio to the RX mode, waiting for incoming data from other nodes. The radio is kept on until the end of the slot, because the senders may transmit data anytime within the slot. So, the energy consumption is the same regardless whether or not the senders use multi-path routing or retransmission.

Second, we examine a sender's energy consumption. When an outgoing message is posted on the radio stack, the radio is turned on and set to the RX mode for sensing the channel condition. When the channel becomes clear, the radio starts transmission of the message. After completing the transmission, the sender may turn off the radio, or wait for an acknowledgment if the message is unicast and configured to be acknowledged. If the acknowledgment is not received before the acknowledgment timeout, the sender retransmits the message. The sender turns off the radio if the acknowledgment is received, or the retransmission limit is reached.



(a) Multi-path routing



(b) Retransmission

Figure 3. Wireless communication uses CC2420 radio stack in TinyOS-2.x

In Figure 3(a), node 1 broadcasts a message consisting of a common header (h) and its partial result (d1) after a backoff period of waiting for clear channel. In the next slot, the forwarding nodes 2, 3, and 4 broadcast d1 together with their own data (d2, d3, and d4, respectively). The total energy cost for d1 is $P_{rx} * T_{backoff} + P_{tx} * (T_h + 4 * T_{d1})$, where P_{tx} and P_{rx} are the power consumption when the radio is in the TX and RX mode, respectively, and T_h and T_{d1} are the time for actually transmitting d1 and the common header (h), respectively. In Figure 3(b), node 1 unicasts its partial result with two retransmissions. Then, the total energy cost is $(P_{rx} * (T_{backoff} + T_{ack}) + P_{tx} * (T_h + T_{d1})) * 3$.

In general, the total energy cost for the multi-path routing (P_{mp}) and the retransmission (P_{retx}) are

$$P_{mp} = P_{rx} * T_{backoff} + P_{tx} * (T_h + (k+1) * T_{d1})$$

and

$$P_{retx} = (P_{rx} * (T_{backoff} + T_{ack}) + P_{tx} * (T_h + T_{d1})) * (r+1)$$

where k is the number of nodes that forward $d1$, and the r is the number of retransmissions.

To have a back-of-the-envelop calculation of the energy cost, we ran a few tests on the Motelab testbed, which consists of about 65 TMote Sky nodes using the popular CC2420 radio. We measured the time for the initial backoff, transmitting the payload, and waiting for the acknowledgment by setting timestamps at a number of points in the radio stack. We find that $T_{backoff} = 7.6ms$, $T_{ack} = 2.9ms$, and the time of transmitting d bytes is $d * 0.035ms$. From the CC2420 radio datasheet, $P_{rx} = 31.3mW$ and $P_{tx} = 35.5mW$. The common header of the CC2420 radio stack, including both the physical header and the MAC header, is 20 bytes long. These numbers show that transmitting a few extra bytes consumes much less energy than idle listening during the backoff and the wait for an acknowledgment. Therefore, the multi-path routing is more energy-efficient than retransmission.

Based on the analysis, OPAG lets a node exploit multi-path routing if the successful delivery ratio over redundant paths is above a given threshold, and uses retransmission otherwise.

3.3 Protocol Overhead and Limitation

OPAG incurs both communication and storage overheads.

The communication overhead consists of two parts: the messages for DAN announcements, and the extra bits the forwarding nodes transmit to compensate communication losses.

The former can be omitted because the DAN announcements are scheduled in the beacon slots, during which each node has to send and receive the beacons anyway. Of course, the DAN announcements should not make up too many messages, jamming the radio channel. Because each node only advertises a few nodes at the top of the candidate lists, it only needs to send one extra message in each beacon slot. If, for optimization, a beacon slot is omitted since link quality does not change much, the DAN announcements may also be skipped without hurting the performance since DANs are selected using the link-quality information.

The latter is the cost OPAG pays to combat communication losses. We argue that, to tolerate moderate communication losses, transmitting more bits can be much more energy-efficient than retransmitting the entire message. We will show the improvement of energy-efficiency in the next section.

The storage overhead incurs for the DAN candidate list and the data buffers. OPAG only keeps a few good DAN entries, so the space cost is small. Each node needs a data buffer to store the partial results it has to forward. Since all DANs are within a few hops, there are only a limited number of partial results a node may need to forward. In very dense networks, a node can randomly pick a few partial results it may consider to forward based on the availability of space.

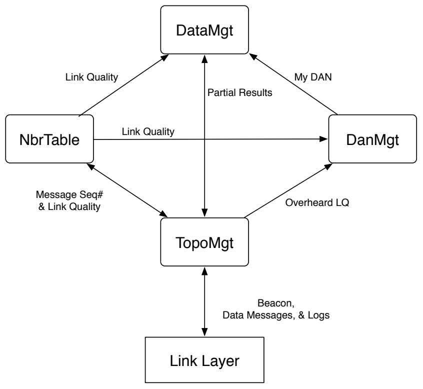


Figure 4. Architecture of OPAG implementation on TMote Sky Node.

4 System Implementation and Evaluation

This section first details the architecture of our implementation, then describes our evaluation methodology and experimental setup, and finally presents our experimental results.

4.1 Implementation Details

We implemented OPAG on the TinyOS 2.0.1, and TMote Sky platform which is equipped with an 8MHz TI MSP430 processor, 10K RAM, and a 250Kbps Chipcon radio operating at 2.4GHz. It supports various sensors, such as light, temperature, and humidity.

The implementation of OPAG is based on the Collection Tree Protocol (CTP) in TinyOS-2.x source[7]. As shown in Figure 4, OPAG has four major components: TopoMgt (Topology Management, a modified version CtpRoutingEngine), NbrTable (Neighbor Table, slightly modified from LinkEstimator), DanMgt (Data Aggregation Node Management), and DataMgt (Data Management, modified from CtpFowardingEngine).

Like CtpRoutingEngine, TopoMgt estimates and exchanges link-quality information by periodically broadcasting beacon messages. Then, based on the link quality, TopoMgt forms a spanning tree. However, TopoMgt makes the following three changes to CtpRoutingEngine: (1) each beacon message bears the current clock to achieve a loose time synchronization; (2) the beacon timer is modified so that each node only broadcasts beacons during the beacon slots, and after each beacon message, a DAN announcement message is sent; (3) in addition to using bidirectional link quality, TopoMgt skips those neighbors with poor outgoing link quality when selecting a parent node.

DataMgt extracts the partial results from incoming data messages, and then processes them in three different ways: (1) if a partial result's DAN is the current node, DataMgt aggregates it; (2) if the result's DAN is not in the DAN candidate list, DataMgt discards it, because this node would have received some advertisements of the DAN if it is on a forwarding path; (3) for the other partial results, DataMgt selectively forwards some of them, depending on the availability of message payload length.

DanMgt maintains the list of DAN candidates and computes the probabilities to select the best DAN. NbrTable maintains the neighbor table and link-quality information, just like LinkEstimator.

Because the multi-path routing is more energy-efficient than retransmission, a node exploits the multi-path routing opportunistically. If the best DAN can meet a given probability requirement, it sends its partial result in a broadcast message, and therefore, the partial result may be forwarded via multiple paths; otherwise, it sends the partial result in a unicast message to its parent node. In the former case, broadcast messages are not acknowledged; in the latter case, the parent node only sends an acknowledgment if it successfully receives the child node's partial result, rather than the entire message.

To support the acknowledgment of a specific partial result rather than the entire message, we added in the CC2420 radio stack a segment CRC checking [10]: instead of having one CRC checksum for the entire message, a 1-byte CRC checksum is added to each of the partial results within the message. The original CRC checksum only verifies the message header. Upon receiving a unicast message from a child node, the parent node first checks the header's CRC. If the header is corrupted, the whole message is dropped. If the child's partial result was received correctly, the parent node sends an acknowledgment. Other partial results, if received correctly, are processed as usual, or discarded if corrupted. The child node needs to retransmit the unicast message unless it receives the acknowledgment of its partial result or the retransmission limit is reached. Other nodes overhear the unicast message and process the partial results individually.

Aggregation query and accuracy: In this paper, we focus on commonly-used, duplicate-sensitive aggregates, such as COUNT, SUM, and AVG. The size of a partial result for these simple aggregates is usually small, so a data message can hold several partial results, thereby making message multiplexing possible. For complicated queries, like histogram and Sketch, the partial result size may vary, depending on the amount of data involved and the accuracy of the partial result [11, 9]. So, there is contention for limited message space between the accuracy of a partial result itself, and the need for message multiplexing to tolerate message losses. This accuracy–energy tradeoff for complicated queries is left as our future work.

We use the relative error to evaluate the performance of different aggregation approaches. The relative error is defined as $\frac{|x-\hat{x}|}{|x|}$, where $\hat{x}$ is a result and x the true value. In order to exclude the effect of the sensor data, we run the COUNT aggregate over the entire network. In the figures of aggregation accuracy, unless otherwise noted, we show the median values as well as the 5 and 95 percentiles. We do not use the percentage contribution because it does not reflect the computation error incurred by statistical counting in SD.

Energy-efficiency: Each node stays awake throughout the beacon slot and its receiving slot, and hence the difference is the energy it spends in sending data messages. So, we measured the time of sending data messages and converted it to energy consumption.

We timestamped a few points in the CC2420 radio stack and measured the time each node spends in TX/RX mode during the sending slots. To get the energy consumption, we multiply the time in RX mode by $P_{rx} = 35.5mW$ (i.e., the receive power), and the time in TX mode by $P_{rx} = 31.3mW$ (i.e., the maximum transmit power). We also used a lower transmit power, $P_{tx} = 22.5mW$ in our evaluation.

4.2 Experimental Setup

For the purpose of performance comparison, we implemented OPAG (OPAG without the redundancy control), TAG (TAG's tree-based approach), TAG-RETX (TAG with retransmission), and SD (Synopsis Diffusion over the ring structure), based on the CTP source. Because the level-1 nodes are just one hop away and cannot establish multiple paths to the BS, SD allows these nodes to send each data message three times [17]. For a fair comparison, the level-1 nodes in TAG also send each data message up to three times. TAG and SD do not retransmit lost messages except for the level-1 nodes. TAG-RETX retransmits a message unless it is acknowledged or there have been 4 retransmissions. We do not include Sketch, because it is almost identical to SD, except that SD uses 32-bit bitmaps, while Sketch uses 16-bit bitmaps.

To compensate for message losses, OPAG uses multipath routing if the redundant paths give a success ratio no less than p, and use retransmissions if there is no redundant path or the multiple paths yield a success ratio less than p. A node sets p to 0.95 if it has aggregated the sensor readings of more than 3 nodes, and 0.9 otherwise. Therefore, the partial results with more contribution to the final aggregation result are unlikely to be lost.

All of the approaches under consideration follow the same scheduling as follows. Each epoch is 120s long and consists of 24 slots. A common slot is assigned for all nodes to exchange beacon messages. Each node is assigned a receiving slot to receive partial results from other nodes, and a sending slot to send its own partial result (and the partial results it has to forward under OPAG, depending on the level of the node in the topology.

To evaluate our implementation, we used Motelab, the wireless sensor network testbed at Maxwell Dworkin Laboratory of Harvard University. Motelab has about 190 Tmote Sky nodes scattered across a number of rooms on three floors, and about 65 nodes can be programmed. The environment should have enough multi-path effects from obstacles and interference from other wireless communications as a realistic sensor network deployment does.

Each node uses a CC2420 radio operating at the default radio frequency — channel 11 of IEEE 802.15.4 (2.405 GHz) with the maximum transmit power. It uses the CC2420 radio stack in TinyOS-2.0.1, which sends 802.15.4-compliant packets, but does not implement the 802.15.4 MAC protocol. The physical header and the MAC header use 20 bytes. The message payload length is limited by the size of the physical data buffer (120 bytes). We set the maximum payload size to 60 bytes. Each partial result is 10 bytes long, including 1-byte reserved for filtering probability and 1-byte CRC checksum.

We used the maximum transmit power), corresponding to 0 dBm, respectively. We set node 2 to be the data sink, and then have 59 nodes join the spanning tree. Therefore, we have a topology of relatively strong connectivity with an average node degree of 9.7, and a network diameter of 6.

For the results presented in this paper, all DAN advertisements are propagated within 2 hops for the following reason. When the DAN is located 3 hops away, the multiple paths from a node to that DAN are very likely braided. Instead of deriving the accurate success ratio over the braided paths, we simply multiply 0.8 to the ratio which assumes the paths are independent of each other. Then, using the estimated success ratio, a node very rarely selects its DAN from 3 hops or more away.

We ran each test 5 times, and each run lasted about 25 minutes. The first 5 minutes is the warm-up period, in which nodes build up a spanning tree. Then, the query runs for 8 epochs, each for 2 minutes.

4.3 Experimental Results

4.3.1 Aggregation Accuracy

Figure 5 shows the aggregation accuracy of the four schemes. In both topologies, TAG has the worst average relative error due to message losses. Using retransmissions, TAG-RETX can reduce the error by 50-65%, as compared

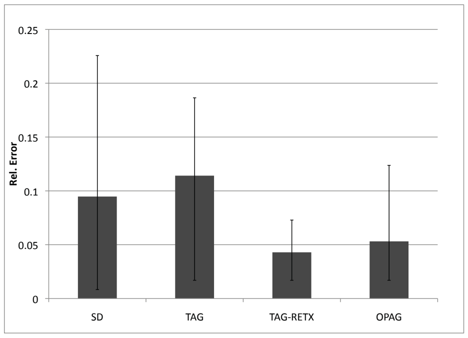


Figure 5. Aggregation accuracy

to TAG. Compared to TAG-RETX, OPAG incurs a slightly higher error, because the nodes have better connectivity and thus form more redundant paths. OPAG uses multi-path routing more aggressively, therefore, with a small probability, a partial result may be lost on all paths, leading to slightly higher aggregation error, as compared with TAG-RETX.

SD is very insensitive to the change of topology and network connectivity because (1) its multi-path routing is extremely aggressive, and (2) the variance of the estimation is significant. More than 5% of the estimated results are off the true value by more than 20%.

4.3.2 Energy-Efficiency

Before analyzing energy-efficiency, we first show the breakdown of time and energy in sending a message by the number of retransmissions in Figure 6 and Figure 7 respectively. They illustrate that the energy cost increases significantly as more retransmissions are incurred. In fact, as shown in Figure 7, idle listening (in the backoff and the ack waiting) takes more than 90% of the energy cost, and sending a message with 4 retransmissions consumes 7 times as much as sending a message without retransmission.

Because the time for transmitting data is much less than the backoff time and the time for waiting for the acknowledgments, it is more energy-efficient to let the forwarding nodes on the multiple paths to send a few extra bytes than retransmitting the lost messages.

In Figure 8, we show the distribution of the transmitted messages by the number of retransmissions. In both topologies, about 88% of the messages are sent successfully without retransmission. Due to better connectivity, OPAG can take advantage of the multi-path routing and increase this percentage to about 95% at the expense of slightly lower accuracy. The energy cost is reduced by about 33% compared to TAG-RETX, and is about the same as SD, as shown in Figure 9. This indicates that our opportunistic schemes can cut the aggregation error by half at roughly the same energy cost in networks of good connectivity.

TAG consumes the least amount of energy because each

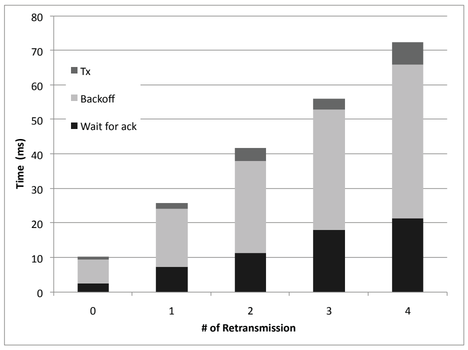


Figure 6. Time breakdown by # of retransmissions

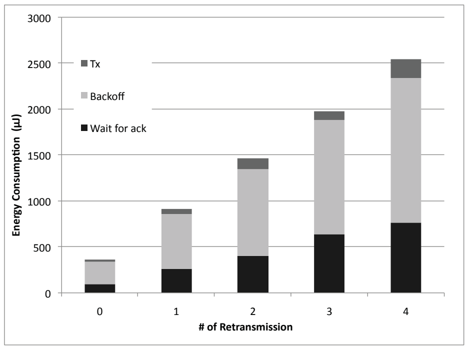


Figure 7. Energy breakdown by # of retransmissions

node sends one short message without retransmission. SD consumes more energy than TAG because it needs to transmit more bytes in each message.

Additionally, Figure 7 and Figure 8 show that OPAG can balance the energy consumption, because some nodes can avoid retransmissions which incur much higher energy consumption, and the forwarding nodes only need to consume a little more energy for transmitting more bits,

5 Related Work

A number of researchers studied data aggregation in wireless sensor networks [15, 14, 21, 23, 17, 5, 16]. Their

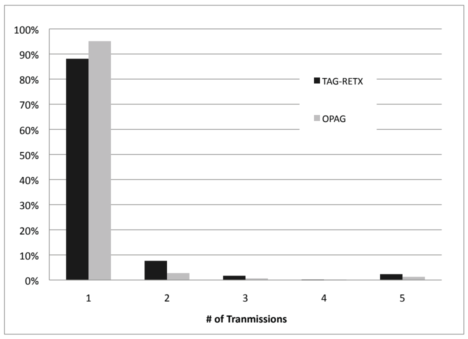


Figure 8. The distribution of retransmission

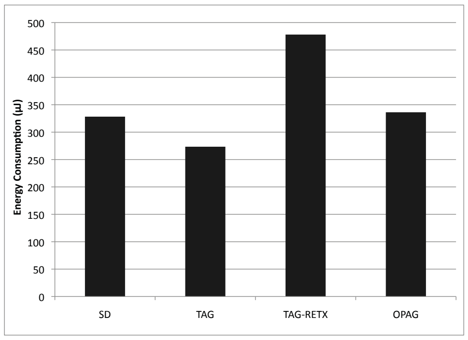


Figure 9. Avg. energy consumption for sending data per epoch per node

approaches use a tree topology (value-splitting can be considered as a special approach based on a tree topology) with exact computation of aggregate results, a ring topology with statistical estimation, or both. The tree-based approaches [15, 14, 21, 23] do not incur any computation error, but are not robust to message losses. On the contrary, the ring-based approaches [17, 5] are very robust against message losses by aggressively exploiting multi-path routing, but statistical estimation leads to significant result inaccuracy.

To combine the advantages of tree topologies and multi-path routing, Tributary-Delta (TD)[16] allows a WSN to form a hybrid topology, visually termed as *tributary-delta*, where a *tributary* refers to an area where nodes in a very good network condition form a sub-tree, and the *delta* refers to the area where the BS and some nodes form a multi-path sub-graph. When the network condition changes, the delta area and the affected tributary areas can be adjusted adaptively subject to the restrictions of topological correctness. TD can significantly improve the accuracy of aggregate results over that of Sketch/SD only if the BS receives exact partial aggregate results that make up a good portion of the final result. This requires the BS to be fed directly by some sub-trees which cover a large part of the network. However, with strict restriction of topological correctness, such scenarios are unlikely to hold in the real world. Only a few random spots under poor network conditions can make the delta area cover almost the entire network.

OPAG is a new data aggregation scheme with zero computation error and good tolerance to moderate message loss. It separates in-network data aggregation into two layers: (1) at the data-aggregation layer, aggregation results are computed exactly along an overlay tree; and (2) at the underneath routing layer, a node opportunistically uses a multi-path routing scheme to send its partial result to a data aggregation node. And the multi-path routing in OPAG differs from that of Sketch, SD and TD.

GRAB [22] proposes a credit-based forwarding scheme to control the redundancy of multiple paths between a data source and a data sink. Each node of the sensor network is assigned a cost, which is the minimum energy to forward a packet from this node to the sink. Every packet is broadcast with a credit, and only the neighbors whose costs are below the credit re-send the packet. Therefore, the amount of credit determines the redundancy of the forwarding mesh. Through simulations, GRAB shows how much credit is needed to achieve a certain success ratio. In contrast, OPAG is motivated to control the redundancy of data entries rather than that of the paths. We analyze the relationship between the data redundancy and the success ratio, and then design an algorithm that probabilistically forwards the data entries with minimum redundancy while retaining a given success ratio.

Boulis *et al.* [3] study the tradeoff between the aggregation accuracy and the energy consumption by taking advantage of the spatial-temporal correlation among sensor values. The idea is to create a system-level energy vs. accuracy knob whereby the more less accurate the aggregation results need to be, the less sensor values are used to estimate the aggregate results by using the data correlation more aggressively, and therefore less messages need to be exchanged. This approach is orthogonal to OPAG, as OPAG takes advantage of good network connectivity rather than data correlation. Proper combination of the two approaches may achieve better accuracy-energy tradeoff.

In the context of splitting a packet over a number of disjointed paths with forward error correcting (FEC) codes, Dulman *et al.* [6] analyzed the relationship between the number of successfully delivering paths and the overall success probability of restoring the packet. By contrast, OPAG delivers each data entry through a set of paths without splitting it, and its analysis focuses on the relationship between the data redundancy and the success ratio of data delivery.

Parametric Probabilistic Routing [2] is a multi-path routing scheme for one-to-one communication in WSNs. Instead of specifying the paths between a source and a sink, the scheme allows each neighbor to forward packets with a certain probability based on the hop counts between the source, the sink, and the forwarding node. OPAG deals with in-network data aggregation, and the forwarding probability of a data entry is based on the data redundancy and the given threshold of success ratio.

Directed Diffusion [12, 8] proposes a data-driven communication paradigm for WSNs. A data sink distributes a sensing task in the sensor network as an "interest," and the distribution process sets up gradients which are used to forward data from the data sources to the sink along multiple paths. Local rules are exploited to reinforce one or a small number of high-quality paths to reduce data redundancy. The tradeoff between the maintenance overhead of alternative paths and the resilience to node failures is also studied. The scheme is not tailored to in-network data aggregation, and do not explore the relationship between the data redundancy and the success ratio, either.

Dozer [4] presents a data-gathering system which achieves impressive power efficiency — in the magnitude of 0.2% radio duty cycles. It uses a spanning tree topology and coordinates the communication between a parent and its children nodes by a TDMA protocol. Because wireless communication is inherently lossy. OPAG can be complement to Dozer in reducing retransmissions and further lowering the energy consumption.

6 Conclusion

In this paper, we presented a novel approach, called *Opportunistic Data Aggregation* (OPAG), to *in-network* data aggregation with no computation error and tolerance to moderate message losses in wireless sensor networks. By space-multiplexing messages, OPAG divides in-network data aggregation into two layers: (1) at the data-aggregation layer, intermediate aggregation results are computed exactly along an overlay tree; and (2) at the underlying routing layer, a node may send intermediate results to its aggregation node via multiple paths.

OPAG opportunistically uses multi-path routing to combat communication losses and achieve better energy-efficiency than using retransmission. This is attributed to the observation that, when sending a message, the radio (e.g., the widely-used CC2420 radio) may consume much more energy on idle listening during the backoff period and the time on waiting for the acknowledgment than transmitting data bytes. Retransmitting a message is not energy-efficient because it incurs more idle listening on more backoffs and more time of waiting for an acknowledgment. In order to avoid extra idle listening, OPAG uses a multi-path routing scheme that differs from the traditional multi-path routing. Each forwarding node dynamically sends multiple partial results—including the partial results it needs to forward and its own partial result—in one message. The receivers disassemble the message and process the partial results in the message individually—they may aggregate, forward, or discard a partial result, depending on the aggregation node specified in the partial result.

We implemented OPAG on TinyOS-2.x and the TMote Sky node, and evaluated its performance on the Motelab Testbed. With good network connectivity, OPAG has more opportunities to exploit multi-path routing, thus reducing the energy cost by 33%, compared to the reliable TAG (TAG with retransmission). and its relative error is only slightly higher. Compared to SD, OPAG cuts the aggregation error by half at roughly the same energy cost.

In future, we would like to consider the tradeoffs between the accuracy of intermediate results and degree of tolerance to message losses by space-multiplexing messages for complex queries.

Acknowledgement

The work reported in this paper was supported in part by the US Army Research Office under Grant W911NF-05-1-0421, and by the NSF under CNS-0435023.

References

[1] D. Aguayo, J. Bicket, S. Biswas, G. Judd, and R. Morris. Link-level measurements from an 802.11b mesh network. In *SIGCOMM*, 2004.

[2] C. L. Barrett, S. J. Eidenbenz, L. Kroc, M. Marathe, and J. P. Smith. Parametric probabilistic sensor network routing. In *Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications*, 2003.

[3] A. Boulis, S. Ganeriwal, and M. Srivastava. Aggregation in sensor networks: an energy-accuracy trade-off. In *SNPA*, 2003.

[4] N. Burri, P. von Rickenbach, and R. Wattenhofer. Dozer: ultra-low power data gathering in sensor networks. In *IPSN*, 2007.

[5] J. Considine, G. K. F. Li, and J. Byers. Approximate aggregation techniques for sensor databases. In *ICDE*, 2004.

[6] S. Dulman, T. Nieberg, J. Wu, and H. Havinga. Trade-off between traffic overhead and reliability in multipath routing for wireless sensor networks. In *Wireless Communications and Networking Conference*, 2003.

[7] R. Fonseca, O. Gnawali, K. Jamieson, S. Kim, P. Levis, and A. Woo. The collection tree protocol, tinyos-2.x tep 123. http://www.tinyos.net/tinyos-2.x/doc/html/tep123.html.

[8] D. Ganesan, R. Govindan, S. Shenker, and D. Estrin. Highly-resilient, energy-efficient multipath routing in wireless sensor networks. *Mobile Computing and Communications Review*, 1(2), 2002.

[9] D. Ganesan, B. Greenstein, D. Perelyubskiy, D. Estrin, and J. Heidemann. An evaluation of multi-resolution storage for sensor networks. In *ACM SenSys*, 2003.

[10] R. K. Ganti, P. Jayachandran, H. Luo, and T. Abdelzaher. Datalink streaming in wireless sensor networks. In *Proceedings of ACM SenSys*, 2006.

[11] J. M. Hellerstein, W. Hong, S. Madden, and K. Stanek. Beyond average: Toward sophisticated sensing with queries. In *IPSN*, 2003.

[12] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed Diffusion: A scalable and robust communication paradigm for sensor networks. In *Proceedings of 6th Annual International Conference on Mobile Computing and Networking (MobiCom'00)*, 2000.

[13] P. F. jolet and G. N. Martin. Probabilistic counting algorithms for data base applications. *Computer and System Sciences*, 1985.

[14] S. Madden, M. Franklin, J. Hellerstein, and W. Hong. The design of an acqusitional query processor for sensor networks. In *SIGMOD*, June 2003.

[15] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. TAG: a tiny aggregation service for ad-hoc sensor networks. In *OSDI*, December 2002.

[16] A. Manjhi, S. Nath, and P. B. Gibbons. Tributaries and deltas: Efficient and robust aggregation in sensor network streams. In *SIGMOD*, 2005.

[17] S. Nath. Synopsis diffusion for robust aggregation in sensor networks. *SIGMOBILE Mob. Comput. Commun. Rev.*, 8(4), 2004.

[18] S. (previously as Moteiv). Moteiv. http://www.sentilla.com/.

[19] N. Reijers, G. Halkes, and K. Langendoen. Link layer measurements in sensor networks. In *Int. Conf. on Mobile Adhoc and Sensor Systems*, 2004.

[20] G. Werner-Allen, P. Swieskowski, and M. Welsh. Motelab: A wireless sensor network testbed. In *IPSN'05, Special Track on Platform Tools and Design Methods for Network Embedded Sensors (SPOTS)*, 2005.

[21] Y. Yao and J. E. Gehrke. Query processing in sensor networks. In *First Biennial Conference on Innovative Data Systems Research (CIDR)*, January 2003.

[22] F. Ye, G. Zhong, S. Lu, and L. Zhang. A robust data delivery protocol for large scale sensor networks. In *Proceedings of the 2nd International Workshop on Information Processing in Sensor Networks*, 2003.

[23] J. Zhao, R. Govindan, and D. Estrin. Computing aggregates for monitoring wireless sensor networks. In *IEEE SPNA*, 2003.