Aggregation Protocols for High Rate, Low Delay Data Collection in Sensor Networks *

Jie Feng, Derek L. Eager, and Dwight Makaroff

Department of Computer Science, University of Saskatchewan,
Saskatoon, SK S7N 5C9, Canada
{jif226, eager, makaroff}@cs.usask.ca

Abstract. Sensor network applications commonly require sensor data to be periodically collected. Aggregation protocols can make this process considerably more efficient. This paper considers the problem of devising aggregation protocols for applications that must achieve as "real-time" a view of the monitored area as possible, entailing a high sampling rate and a low data collection delay, at the possible cost of some modest amount of data loss. We examine in particular broadcast-based protocols that minimize the number of packet transmissions, relying on multipath delivery rather than ARQ for reliability, and consider the question of whether such protocols can achieve lower collection delays and support higher sampling rates than conventional aggregation protocols. Our results suggest that broadcast-based protocols can yield significantly improved performance in some scenarios, when sensor data can be aggregated into packets of size that is independent (or largely independent) of the number of values being aggregated.

Key words: sensor networks, aggregation protocols, performance evaluation

1 Introduction

Sensor networks consist of a potentially large number of sensor nodes capable of capturing measurements of their immediate environment, together with one or more "sink" nodes at which sensor data can be collected. Data collection may be either periodic [1][2][3][4], as in environmental monitoring applications for example, or as needed (aperiodic) [5][6][7] in response to exceptional events. We consider here the case of monitoring applications that must achieve as "real-time" a view of the monitored area, at a single sink node, as is possible. For this purpose, periodic data collection is required, with a high sampling/collection rate and a low data collection delay.

The efficiency of data collection can be vastly improved by aggregating the sensor data received at intermediate nodes into fewer numbers of packets, as

* To appear in *Proc. IFIP/TC6 Networking 2009*, Aachen, Germany, May 2009. This work was supported by the Natural Sciences and Engineering Research Council of Canada.

this data is being forwarded to the sink [1] [2][3][4][5][6]. By reducing the number of transmitted packets, aggregation can reduce energy usage, increase the achievable data collection rate, and (owing to reduced network contention) decrease the data collection delay. Aggregation may be achieved in an application-independent manner by simply concatenating (possibly in compressed form) the sensor values received from multiple sensors in one packet [8]. Alternatively, aggregation may make use of application semantics, such as when only the maximum sensor reading is required; in this case, only the largest value need be forwarded [1].

Existing protocols for data collection using aggregation ("aggregation protocols") may be classified as either asynchronous or synchronous and according to whether they use unicast or broadcast communication [1][2][3][9][10][11]. The former distinction concerns how a node determines when to wait for additional data from upstream nodes, and when to send downstream an aggregate packet containing whatever data it has received up until then. In TAG, for example, a synchronous approach is used, in which all nodes i network hops from the sink forward their (aggregated) data to their parent nodes in the aggregation tree during interval $d - i$ of each "round" of communication, where one set of sensor readings is collected each round, and where d is the maximum number of hops from the sink [1]. In contrast, with asynchronous protocols, each node adaptively determines when to send versus when to wait based on its local history of past packet receptions from upstream nodes.

Most prior aggregation protocols use unicast transmission, with reliability achieved using an acknowledgement/retransmission (ARQ) facility, as provided by the link layer for example. The prior work concerning aggregation protocols using broadcast transmission has focused on protocol mechanisms for "duplicate sensitive" aggregation [9][10], in which the sink must never receive multiple aggregates including the same sensor value (or the same share of a sensor value, if "value splitting" [1] is employed).

We focus on protocol mechanisms that broadcast-based protocols can employ to maximize the achievable sampling/collection rate and minimize the collection delay with some modest amount of data loss, and the question of whether such protocols can achieve better performance in these respects than unicast-based protocols. It is assumed acceptable for the sink (and intermediate nodes) to receive multiple aggregates including the same sensor value, either because aggregation is duplicate insensitive (e.g., only the maximum sensor value is needed), or duplicates can be filtered (each aggregate is a concatenation of sensor values).

Both synchronous and asynchronous broadcast-based aggregation protocols are developed. They take advantage of the fact that multiple downstream nodes may be potential receivers of a single broadcast transmission, and of the ability of a node to listen to transmissions from neighbouring nodes so as to determine which (if any) have included that node's broadcast data in their own transmissions. The latter ability is used as a substitute for acknowledgements: in addition to the reliability improvement that arises from potential multipath delivery to

the sink, we also achieve improved reliability through use of two-phase protocols in which a node may repeat (once) its broadcast.

We compare unicast-based protocols and the new broadcast-based protocols mostly using simulation. Rather than assuming some particular aggregation function, two extreme cases are considered. In one of these, it is assumed that sensor data can be aggregated into packets of size that is independent of the number of values being aggregated. In the other, required packet size is assumed to increase linearly with the number of values included in the aggregate. The first case applies with duplicate-insensitive aggregation functions such as "maximum". The first case would also clearly apply with duplicate-sensitive aggregation functions such as "average", were it not for our assumption that it is possible to recognize and filter out duplicates that have been included in multiple aggregates. Which of these two cases more closely reflects reality may depend on the extent to which sensor values and identifiers can be encoded in highly-compressed form in the aggregates.

We find that for a fixed required packet size, broadcast-based protocols are able to support higher sampling/collection rates with lower collection delays, than unicast-based protocols. The performance improvements are particularly pronounced for synchronous aggregation. When the required packet size is assumed to increase linearly with the number of values included, however, we find no significant performance advantage with broadcast-based protocols.

The remainder of the paper is organized as follows. Sections 2 and 3 review the synchronous and asynchronous unicast-based protocols, respectively, on which our new broadcast-based protocols are based, and then describe the new protocols. Section 4 presents simulation results evaluating the performance of the new protocols in comparison to that of the unicast-based protocols. Section 5 provides initial experimental results. Section 6 concludes the paper.

2 Synchronous Aggregation

With synchronous aggregation protocols, all sensor nodes at the same distance (number of hops) from the sink are given the same interval of time in which to transmit, within each round of communication. Nodes farther away from the sink are given earlier transmission intervals, so as to allow their data to be aggregated with that of nodes closer to the sink before these latter nodes make their own transmissions. We first briefly describe the unicast-based synchronous protocol on which our new broadcast-based synchronous protocol is based, in Section 2.1, and then describe the design of the new protocol in Section 2.2.

2.1 Unicast-based

The unicast-based synchronous protocol is a variant proposed in previous work[3] of the original synchronous aggregation protocol as used in TAG [1]. Aggregation is performed over a tree rooted at the sink. Sensor readings are made periodically with period duration t. Nodes at different tree levels are assigned to different

transmission intervals within each round of communication (i.e., collection of one set of sensor values) based on their distances to the sink. It is assumed that each node i knows its hop count h_i to the sink and the maximum hop count H in the tree, and accordingly chooses its transmission interval within each round.

Let I denote the interval duration. In each round j, node i picks a random value r_i^j between 0 and $\lambda I, 0 \leq \lambda \leq 1$, aggregates the data it has received for this round, and sends out its packet at time $T_0 + t(j-1) + (H - h_i)I + r_i^j$. Here it has been assumed that all nodes agree on the same base time T_0 defining the beginning of the first round. It has been found that randomizing transmissions over λI yields better performance than when all nodes at the same tree level attempt to transmit at approximately the same time, and that setting λ to 0.8 yields good performance over a wide variety of network configurations [3].

As described previously, we make no assumptions regarding the type of aggregation that is performed (application-independent, duplicate-sensitive, duplicate-insensitive, etc.), but rather attempt to model a range of scenarios in our performance experiments through consideration of two extreme cases with respect to how packet size grows with the number of aggregated values.

2.2 Broadcast-based

In our broadcast-based synchronous aggregation protocol, nodes are organized into a ring topology [11], as shown in Fig. 1. The sink is the only node that is located in ring 0, nodes one hop away from the sink are in ring 1, etc. As in the unicast-based protocol, nodes in different rings are allotted different time intervals within each round of communication for their transmissions. As before, the duration of the period between sensor readings is denoted by t, and the interval duration by I.

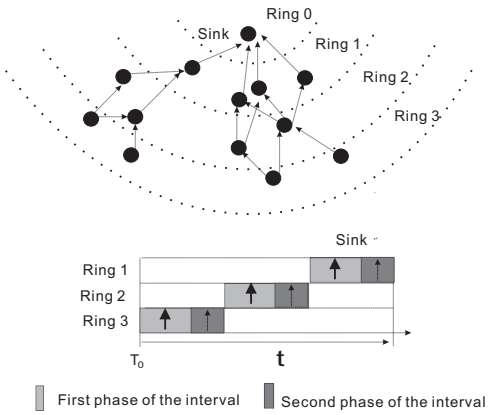


Fig. 1. Synchronous Broadcast-based Aggregation

Unlike the unicast-based protocol, in our broadcast-based protocol each interval is divided into a first and second phase with durations αI and $(1-\alpha)I$, respectively. Each node (except for the sink) broadcasts a packet containing (possibly aggregated) sensor data during the first phase of its interval in each round. Nodes may also make a second broadcast during the second phase, as described below. Each broadcast packet includes a bit vector indicating the nodes whose data is aggregated in the packet. Nodes aggregate all of the data they have received from broadcasts for the current round (including broadcasts from neighbouring nodes in the same ring), for their own broadcasts.

Specifically, in each round j, node i picks a random value $r1_i^j$ between 0 and $(\alpha-0.1)I$, aggregates the data from the broadcasts it has received for this round, and makes its own first broadcast at time $T_0+t(j-1)+(H-h_i)I+r1_i^j$. Node i then picks another random value $r2_i^j$ between 0 and $(1-\alpha-0.1)I$. A broadcast is made in the second phase of the round, at time $T_0+t(j-1)+(H-h_i)I+\alpha I+r2_i^j$, if, by this time, node i has not heard a broadcast transmission from some other node in ring i that has included node i's data (owing to the other node having heard node i's first broadcast, prior to its own first broadcast). Similarly to the unicast-based protocol, randomizing the transmissions within each phase yields better performance than when all of the nodes in the same ring attempt to transmit at approximately the same time.

The second broadcasts are important to improve reliability. For nodes with few neighbours, or nodes that are the last, among the nodes in the same ring, to make their first phase broadcast, a second broadcast increases the likelihood that their data is received by at least one node in the next ring. We have found that this two-phase strategy can reduce the overall end-to-end loss rate significantly, at minimal additional cost in terms of numbers of transmissions.

3 Asynchronous Aggregation

With synchronous aggregation, the time interval during which each node transmits is statically determined. In contrast, in the asynchronous aggregation protocols considered here each node adaptively determines when to transmit based on its history of past packet receptions from its children in the aggregation tree. We use the unicast-based "adaptive asynchronous" aggregation protocol proposed in previous work [3] as a basis for our new broadcast-based asynchronous aggregation protocol. The former protocol is reviewed in Section 3.1, while Section 3.2 describes the design of the new broadcast-based protocol.

3.1 Unicast-based

In each round of the unicast-based asynchronous protocol, a timeout is set at each non-leaf node establishing the maximum time the node will wait to receive packets from its children in the aggregation tree. The timeout value is determined adaptively, based on the timings of packet receptions from the child nodes in the previous rounds. The node transmits its packet (aggregating its own data

with whatever it has received from its children) either when packets have been received from all children, or when the timeout expires. Note that the choice of the timeout is critical: a long wait until timeout may cause excessive delay, while substantial data loss may be incurred if the timeout occurs too soon, since in this case packets may arrive too late to be aggregated (and will be dropped).

As before, we assume all nodes agree on the same base time T_0 defining the beginning of the first round. Each node i picks a random value r_i between 0 and R, where R is a protocol parameter, and at time $T_0 + r_i$ unicasts a packet containing its sensor data for the first round to its parent.

In each subsequent round j, each node i that is a leaf in the aggregation tree sends its sensor data at time $T_0 + r_i + (j-1)t$, where t denotes, as before, the duration of the period between sensor readings. Each non-leaf node transmits a packet containing aggregated sensor data when it has received packets from all of its children for round j, or when its timeout for that round has expired. Timeouts are established as follows. Let L_i^j denote the time at which a non-leaf node i receives the last packet for round j from its children in the aggregation tree. (Note that "last packet" means the last received packet; some packets sent by the children may not be received, owing to communication errors.) Let TO_i^j denote the timeout for round j at node i (i.e., the time at which node i will transmit if packets have not yet been received for round j from all of node i's children). After the first round, the timeout at node i for round two is set to $TO_i^2 = L_i^1 + t$. After each subsequent round $j, j > 1$, the timeout for round $j+1$ at node i is set according to the following rules:

1. If node i received packets for round j from all its children prior to time TO_i^j, its timeout for round $j+1$ is set to $TO_i^{j+1} = (1-\delta)(TO_i^j + t) + \delta(L_i^j + t + e)$. The parameter δ is used to implement an exponential weighted moving average, so as to control how quickly a node reacts to changes in network conditions. Parameter e allows for transmission variance.
2. If the timeout for round j expires before node i receives packets from all of its children, its timeout for round $j+1$ is tentatively set to $TO_i^{j+1} = TO_i^j + t$. If node i receives one or more packets for round j from its children subsequent to the expiry of its timeout for that round, it updates TO_i^{j+1} to $L_i^j + t$. The packets that arrive too late to be aggregated are simply dropped because only up-to-date values are of interest in real-time monitoring.

It is important to randomize the transmission times of leaf nodes to avoid congestion at the beginning of each round. Parameter R controls the duration of the randomization interval. The adaptive protocol achieves improved performance by adjusting the value of R so that it is kept within a certain range, as measured relative to the *average data collection delay* D. Intuitively, if R is "large" relative to D, a substantial portion of the delay D may be due to the randomization delay at the leaf nodes. In this case, it may be advantageous to reduce R. If, on the other hand, R is "small" relative to D, R might be fruitfully increased, so as to better spread out the transmissions of the leaf nodes.

Specifically, the data collection delay for each round j is measured at the sink as the time duration from the start of the round $(T_0 + (j-1)t)$, until the

sink has received the last packet for round j. The average data collection delay is measured as $D = \alpha D + (1-\alpha)D^*$, where D^* is the latest measurement of the data collection delay, and α is a smoothing factor that determines the weight given to the old value. The adaptive protocol attempts to keep the value of R/D in the interval $[\beta-\Delta, \beta+\Delta]$, where β and Δ are protocol parameters. When R/D moves out of this range, the value of R is updated (and sent by the sink to all of the sensor nodes) as follows. If $R/D < \beta - \Delta$, R is updated to $R = D(\beta + \Delta)$. If $R/D > \beta + \Delta$, R is updated to $R = D(\beta - \Delta)$.

3.2 Broadcast-based

In the unicast-based synchronous aggregation protocol used in TAG, aggregation is performed over a tree structure, with one parent for each node except the sink [1]. The authors observe that a node could potentially transmit to multiple parents, each one hop closer to the sink, depending on the density of the network, and propose a "value splitting" aggregation protocol in which each node may have two parents. In the broadcast-based protocol proposed in this section, nodes may similarly have two parents[1], but the same sensor value (rather than only distinct shares of a sensor value) may be aggregated at each parent and forwarded on up the tree. Such multipath routing can yield improved reliability, but requires that nodes be able to receive multiple aggregates including the same sensor value, either because aggregation is duplicate insensitive, or because aggregation is performed in such a way that it is possible to recognize and filter out duplicates.

As with our broadcast-based synchronous protocol, each node (excepting the sink) broadcasts either once or twice during each round. Each broadcast may include not only data from its children, but also data overheard from broadcasts from other neighbours in the tree. Each broadcast packet includes a bit vector indicating the nodes whose data is aggregated in the packet. A non-leaf node makes its first broadcast during a round either when it has received the data from each of its children (either directly from a broadcast by that child, or indirectly via a broadcast from some other child), or upon timeout. Timeouts are established in a similar manner as in the unicast-based asynchronous protocol described in Section 3.1. A second broadcast is made if the node does not hear a broadcast transmission from any other node that includes its data, or if it receives additional data from its children, before a second timeout occurs.

As in the unicast-based protocol, each node i (other than the sink node) picks a random delay r_i between 0 and R, where R is a protocol parameter that is adapted so as to keep the ratio of R to the average data collection delay D within a certain range, as defined by protocol parameters β and Δ. At time $T_0 + r_i$, node i broadcasts a packet containing its own data for the first round.

In each subsequent round j, each leaf node i makes its first broadcast at time $T_0 + r_i + (j-1)t$, aggregating its own data and any other data it has overheard from other broadcasts. Let A_i^j denote the time at which a non-leaf

[1] When a node has only one neighbour closer to the sink it can choose a sibling that has multiple neighbours closer to the sink as its second parent.

node i receives all of the data from its children for round j that it will receive during this round. Let L_i^j denote the time at which non-leaf node i receives the last of the first broadcasts from its children for round j that it will receive during this round. Note, A_i^j may not equal L_i^j. Let TO_i^j denote the timeout for the first transmission of round j at node i. We set $TO_i^2 = L_i^1 + t$.

After each round j, $j > 1$, TO_i^{j+1} is set as follows:

1. If node i received the data for round j from all of its children prior to time TO_i^j, its timeout for round $j+1$ is set to $TO_i^{j+1} = (1-\delta)(TO_i^j + t) + \delta(A_i^j + t + e)$, similarly as in the unicast-based asynchronous protocol.
2. If the first transmission timeout for round j expires before node i receives the data from all of its children, its timeout for round $j+1$ is tentatively set to $TO_i^{j+1} = TO_i^j + t$. If node i receives one or more first-time broadcasts for round j from its children subsequent to the expiry of its timeout for that round, it updates TO_i^{j+1} to $L_i^j + t$. Data from these packets has been received too late to be aggregated in node i's first broadcast for round j, but can be included in node i's second broadcast.

As noted above, following the first broadcast, a second broadcast is made if the node does not hear a broadcast transmission from any other node that includes its data, or if it receives additional data from its children, prior to a second timeout. The second timeout is set to a time duration e following the time of the first broadcast.

4 Simulation-based Performance Evaluation

4.1 Goals, Metrics, and Methodology

The performance of the unicast-based and the broadcast-based protocols is evaluated through ns2 simulation. The primary performance metrics are: (1) the end-to-end loss rate (the ratio of the number of sensor readings not included in the aggregates arriving at the sink to the total number of readings the nodes generate), (2) the maximum data age (t plus the average data collection delay D), a measure of how "stale" the data received at the sink from one round can become, before that for the next round is received, (3) the average number of MAC layer packet transmissions per round (for unicast, including both ACK and data packet transmissions), and (4) the average number of bytes transmitted per round. The last metric yields insight into relative energy usage.

The sensor fields are generated by randomly scattering nodes in square areas. The sink is located in the center of the network. The physical layer packet loss rate is specified as a simulation input parameter. The uniform random error model is used for all experiments. An 802.11 MAC layer is simulated for unicast-based protocols, without RTS/CTS [12], with a transmission range of 40 meters and rate of 2Mbps. Different maximum numbers of MAC layer retransmissions are simulated for when the sender fails to receive an ACK. The same transmission range and data rate are used for the broadcast-based protocols in the simulations.

The protocols are evaluated with both fixed and increasing packet sizes. Fixed-sized packets are 52 bytes. Otherwise the packet size grows linearly with the number of values aggregated in the packet at a rate of 4 bytes per value.

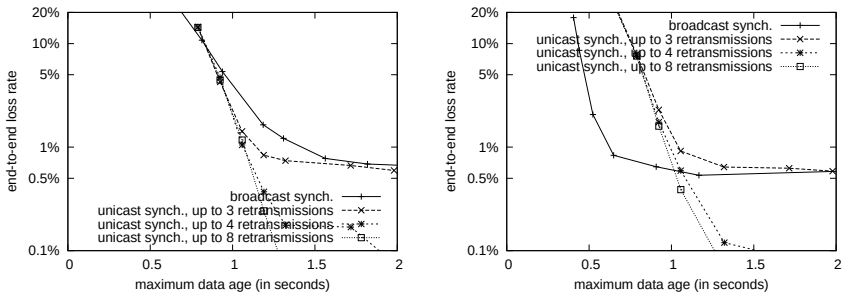
4.2 Principal Performance Comparison

Fig. 2 and Fig. 3 show the performance of the synchronous and asynchronous aggregation protocols, respectively, with our default system protocol parameter settings. The different points on each curve are generated by varying the sampling period duration t, and measuring the resulting maximum data age and end-to-end loss rate. For synchronous aggregation, the interval duration is set to t divided by the maximum hop count to the sink. The initial value for R is 0 for the asynchronous protocols.

Previous work shows that for asynchronous unicast-based aggregation, all parameters except R can be fixed to certain values that yield good performance for a broad range of network configurations [3]. Through experimentation, we find that the same conclusion holds for asynchronous broadcast-based aggregation as well. In all the simulations whose results are reported here, e is fixed at 0.03 seconds and 0.02 seconds for asynchronous broadcast-based aggregation with increasing packet size and fixed packet size respectively. For asynchronous unicast-based aggregation, e is set to 0.15 seconds for increasing packet size and 0.1 seconds for fixed packet size. Parameter β is fixed at 0.7, 0.6, 0.7 and 0.5 for asynchronous unicast-based aggregation with fixed and increasing packet size, and asynchronous broadcast-based aggregation with fixed and increasing packet size, respectively. Other common parameters for both asynchronous unicast-based and broadcast-based aggregation, δ, α, and Δ are fixed at 0.05, 0.875, and 0.15 respectively. Figures showing the impact of the parameters are omitted due to space limitations.

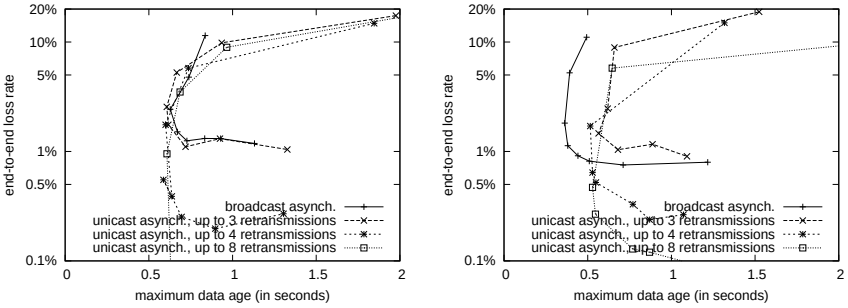
For each specific t, we measure the end-to-end loss rate and the maximum data age of the protocols and plot the results in the figures. As t gets smaller, the end-to-end loss rate starts to grow as more packets are lost because of collisions. For synchronous aggregation, the interval duration gets shorter as t decreases. When the interval is too small, nodes in the same ring cannot make their transmissions within their own transmission interval. Packets that arrive after the receiver sent out its partial aggregate are not aggregated, and the end-to-end loss rate increases sharply. With asynchronous aggregation, not only does the loss rate increase sharply when the sampling rate is too high, but the maximum data age also increases, since nodes increase their timeout values (and therefore their delays before sending their aggregates) to reflect the late arrival times of packets that have been retransmitted. This explains why the curves turn sharply and move to the upper right corner in Fig. 3.

The figures show that both synchronous and asynchronous broadcast-based aggregation are outperformed by their unicast-based counterparts for increasing packet size. For fixed packet size, broadcast-based aggregation is able to achieve lower maximum data age than unicast-based aggregation. Comparing Fig. 2 and Fig. 3, it appears that a broadcast-based approach may be most promising for



(a) Increasing Packet Size (b) Fixed Packet Size

Fig. 2. Synchronous Unicast-based and Broadcast-based Aggregation (160 nodes, 250m×250m, 20% physical layer loss rate)



(a) Increasing Packet Size (b) Fixed Packet Size

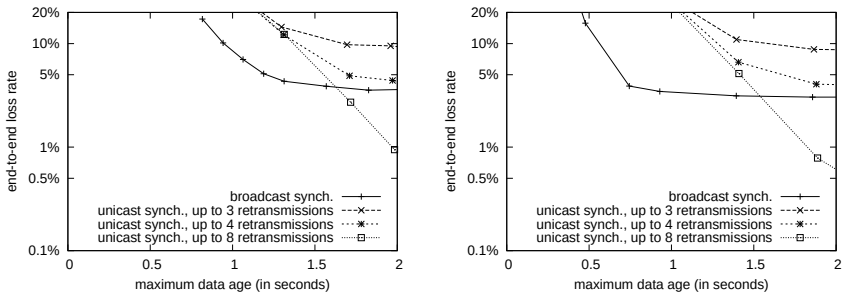
Fig. 3. Asynchronous Unicast-based and Broadcast-based Aggregation (160 nodes, 250m×250m, 20% physical layer loss rate)

synchronous (rather than asynchronous) aggregation. For this reason, and owing to space limitations, in the remainder of the paper only results for synchronous aggregation are presented.

4.3 Loss Rate, Density, and Traffic Volume

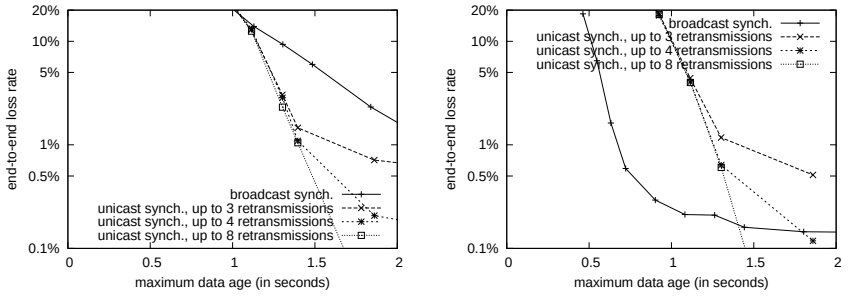
Aggregating the same data at different nodes improves reliability but also increases the packet size for broadcast-based aggregation with increasing packet size. When the physical layer loss rate is 20% as in Fig. 2, the cost of the retransmissions in unicast-based aggregation is relatively modest, and, as seen in Fig. 2(a), unicast-based aggregation outperforms broadcast-based aggregation. As the physical layer loss rate increases, however, the cost of the retransmissions in unicast-based aggregation becomes more substantial.

Fig. 4 shows that for both increasing and fixed packet size, synchronous broadcast-based aggregation is able to achieve lower maximum data age than



(a) Increasing Packet Size (b) Fixed Packet Size

Fig. 4. Impact of Physical Layer Loss Rate on Synchronous Aggregation (160 nodes, 250m×250m, 40% physical layer loss rate)

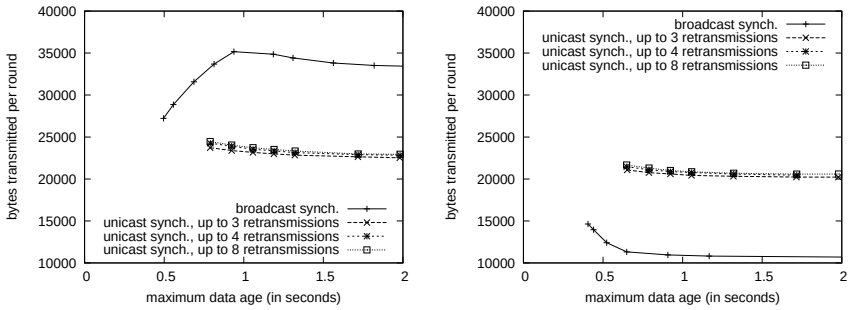


(a) Increasing Packet Size (b) Fixed Packet Size

Fig. 5. Impact of Density on Synchronous Aggregation (240 nodes, 250m×250m, 20% physical layer loss rate)

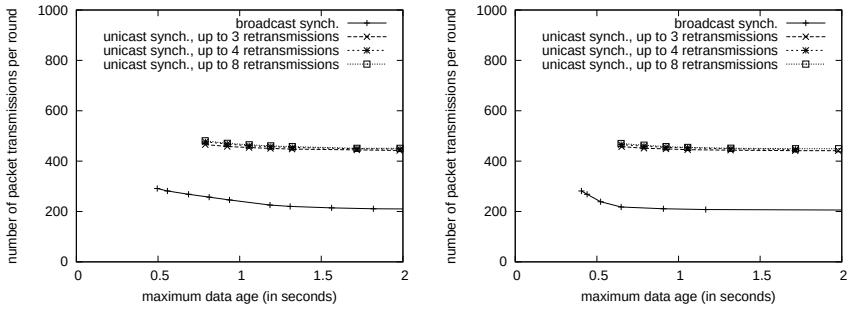
the unicast-based protocol at 40% physical layer loss rate. The increase of the physical layer loss rate has a similar impact on the asynchronous protocols.

As the network density increases, the broadcast-based protocols are expected to achieve higher reliability as a broadcast is likely to be received by more nodes. However, for broadcast-based aggregation with increasing packet size, larger packets are produced as the same data is aggregated by more nodes, which means a longer packet transmission time and greater network congestion. Meanwhile, packet loss recovery is quite feasible for unicast-based protocols with the packet loss rate of 20% that is used for these figures. As the result of these two effects, Fig. 5 shows that synchronous broadcast-based aggregation is outperformed even more by unicast-based aggregation, for increasing packet size, when the number of nodes (and density) is increased. For fixed packet size, however, performance with the broadcast-based protocol improves but degrades with unicast-based aggregation. Similar results are seen with the asynchronous protocols.



(a) Increasing Packet Size (b) Fixed Packet Size

Fig. 6. Average Number of Bytes Transmitted per Round of Synchronous Aggregation (160 nodes, 250m×250m, 20% physical layer loss rate)



(a) Increasing Packet Size (b) Fixed Packet Size

Fig. 7. Average Number of Packet Transmissions per Round of Synchronous Aggregation (160 nodes, 250m×250m, 20% physical layer loss rate)

Fig. 6 shows the average number of bytes that are transmitted per round with the synchronous aggregation protocols, for a 20% physical layer packet loss rate. While broadcast-based aggregation sends fewer packets per round than unicast-based aggregation, as shown in Fig. 7, a larger volume of data is produced by broadcast-based aggregation in the case of increasing packet size. This helps to explain why broadcast-based aggregation yields poorer performance in this case.

5 Preliminary Experimental Results

We report here on results from some preliminary experiments using implementations of the synchronous protocols on Crossbow MICAz motes[2]. For the experiments whose results are reported here, 24 MICAz motes are deployed in an area approximately 9m × 9m, with one mote at the corner as the sink. The

[2] Crossbow: http://www.xbow.com

transmission power level of the motes is set to 3, yielding an approximate transmission range of 2.5m to 3m. (However, the actual connectivity region around each node is highly irregular, as has been observed in previous studies.) In all experiments with unicast-based aggregation, the same aggregation tree shown in Fig. 8 is used, as built using a simple flooding algorithm. For synchronous broadcast-based aggregation, a ring topology is formed with nodes i hops away in ring i. By default, the MICAz uses CSMA/CA at the MAC layer, without packet retransmission. For unicast communication, we activate the automatic retransmission functionality at the motes and set the maximum number of retransmissions to 3, 4 and 8 in the experiments. A fixed payload size of 28 bytes is used.

We carry out multiple sets of experiments, with the experiments in each set using a range of values of the sampling period duration t. Each single experiment runs for 200 sampling rounds. Fig. 9 shows the measured performance of the synchronous protocols from one set of these experiments. Data from the other sets of experiments shows similar results.

Unlike the results from simulations, the experimental results show substantial variability, especially at low sampling rates. Packet loss in the experiments comes from two main sources, contention and physical layer link error. Packet loss due to contention plays a key role in the end-to-end loss at high sampling rates. As t increases, the amount of packet loss from contention decreases consistently and physical layer link error becomes the main reason for packet losses. In our experimental environment, the physical layer link loss is highly bursty, and communication may fail even with large numbers of retransmission attempts.

Where a packet loss happens in the tree also has significant impact on the end-to-end packet loss rate. As can be seen in Fig. 8, there are several key nodes in the aggregation tree, the loss of whose packets results in the loss of the data of most nodes. Since each experiment only lasts for 200 sampling rounds, packet losses at those key nodes, even in just one or two rounds, can have a substantial impact on the measured end-to-end packet loss rate.

Despite the performance variability seen in Fig. 9, taking into account the differing characteristics of the experimental and simulated networks, the experimental results appear to be quite consistent with the simulation results shown in Fig. 2(b), 4(b), and 5(b).

6 Conclusions

In this paper, new synchronous and asynchronous broadcast-based aggregation protocols are proposed and compared to their unicast-based counterparts in the context of real-time monitoring systems. The results show that for aggregation with fixed packet size, broadcast-based aggregation is able to achieve lower maximum data age than unicast-based aggregation, particularly in the case of synchronous protocols. However, for increasing packet size, the results show no significant performance advantage (and sometimes poorer performance) with broadcast-based protocols.

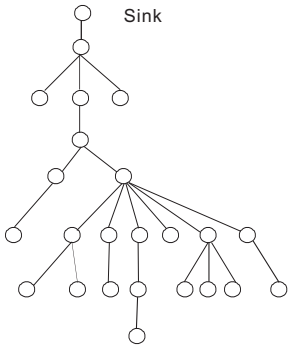


Fig. 8. Aggregation Tree

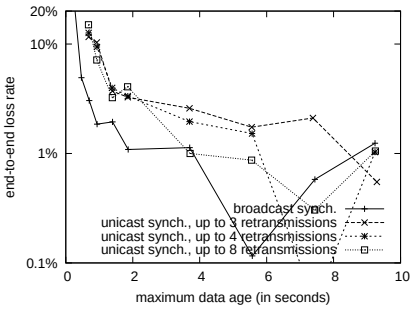


Fig. 9. Experimental Results for Synchronous Aggregation (24 MICAz nodes, 9m×9m, fixed packet size)

References

1. Madden, S., Franklin, M.J., Hellerstein, J.M., Hong, W.: TAG: a tiny aggregation service for ad-hoc sensor networks. SIGOPS Operating System Review **36**(SI) (Dec. 2002) 131–146
2. Solis, I., Obraczka, K.: The impact of timing in data aggregation for sensor networks. In: ICC '04, Paris, France (June 2004) 3640–3645
3. Feng, J., Eager, D., Makaroff, D.: Asynchronous data aggregation for real-time monitoring in sensor networks. In: IFIP Networking '07, Altanta, GA (May 2007) 73–84
4. Heinzelman, W.R., Chandrakasan, A., Balakrishnan, H.: Energy-efficient communication protocol for wireless microsensor networks. In: HICSS '00, Maui, HI (Jan. 2000) 8020–8029
5. Fan, K.W., Liu, S., Sinha, P.: Scalable data aggregation for dynamic events in sensor networks. In: SenSys '06, Boulder, CO (Nov. 2006) 181–194
6. Zhang, W., Cao, G.: DCTC: dynamic convoy tree-based collaboration for target tracking in sensor networks. IEEE Trans. Wireless Commun. **3**(5) (Sept. 2004) 1689–1701
7. Zhang, W., Cao, G.: Optimizing tree reconfiguration for mobile target tracking in sensor networks. In: INFOCOM '04, Hongkong, China (Mar. 2004) 2434–2445
8. He, T., Blum, B.M., Stankovic, J.A., Abdelzaher, T.: AIDA: Adaptive application-independent data aggregation in wireless sensor networks. ACM Trans. on Embedded Computing Systems **3**(2) (May 2004) 426–457
9. Gobriel, S., Khattab, S., Mosse, D., Brustoloni, J., Melhem, R.: Ridesharing: Fault tolerant aggregation in sensor networks using corrective actions. In: SECON '06, Reston, VA (Sept. 2006) 595–604
10. Motegi, S., Yoshihara, K., Horiuchi, H.: DAG based in-network aggregation for sensor network monitoring. In: SAINT '06, Phoenix, AZ (Jan. 2006) 292–299
11. Nath, S., Gibbons, P.B., Seshan, S., Anderson, Z.R.: Synopsis diffusion for robust aggregation in sensor networks. In: SenSys '04, Baltimore, MD (Nov. 2004) 250–262
12. Xu, K., Gerla, M., Bae, S.: Effectiveness of RTS/CTS handshake in IEEE 802.11 based ad hoc networks. Ad Hoc Netw. **1**(1) (Jul. 2003) 107–123