Dynamic Data Aggregation and Transport in Wireless Sensor Networks

Mario O. Díaz and Kin K. Leung
Electrical & Electronic Engineering Department, Imperial College, London, UK
{orne.diaz06, kin.leung}@imperial.ac.uk

***Abstract*—In wireless sensor networks, in-network aggregation is the process of compressing locally the data gathered by the sensor nodes, so that only the compressed data travel across several hops to their destination. We address the problem of aggregating data generated by sporadic events in random locations of the monitored area. The sensor nodes keep their transceivers off most of the time in order to preserve their batteries, and these sleep periods dominate the time to react to the events. We propose a distributed protocol that, after each event, constructs a routing tree to regulate the aggregation process. It is cross-layer because, in order to accelerate the tree construction process, the routing decision considers the sleep periods of the nodes. If the nodes sleep for long periods, our protocol divides the tree construction time by the number of hops when compared to centralized protocols. For a fixed maximum tolerable delay, this allows us to extend the sleep periods and thus to save energy. Our simulations reveal that this comes at the price of an aggregation tree with degraded performance, but we retain an advantage over trees not customized to each event. Our protocol requires global time synchronization and periodic link state monitoring, especially as the network size increases.**

Keywords: data aggregation, tiers, tree, wireless sensor

I. INTRODUCTION

Wireless sensor networks (WSNs) often require battery lifetimes of several years. Duty-cycling and in-network data aggregation are two very efficient techniques to enhance the energy efficiency. Duty-cycling consists of turning off the radio transceiver periodically. It reduces idle listening, which typically consumes most part of the sensors' energy, but it increases latency. In-network data aggregation [1] is the process of compressing the correlated data of neighboring nodes locally before transmitting them to their destination, possibly across many hops. When the data are highly correlated, data aggregation greatly shortens the transmissions, and thus saves energy and time.

In order to aggregate data, most systems construct tree structures for routing and aggregation. Each node waits for data from its children, compresses the data, and it forwards the result to its parent. These systems neglect the overhead they incur in constructing the trees because they assume periodic monitoring applications with long-lasting data flows. Their overhead is excessive for event-triggered applications that report events that do not last for a long period of time. In this paper, we address applications that, in addition to that, are time-sensitive They arise in the WINES research project [2], to which this work belongs. This project researches the use of WSNs in civil engineering applications. As an example, WINES investigates how to detect fractures in bridges and report them promptly to a data center connected to the Internet with sensor nodes equipped with microphones that listen for the acoustic emissions that the fractures release. Prompt reporting is necessary to stop the traffic over the bridge as soon as possible when severe fractures occur.

In order to report this kind of events quickly, we propose a fast cross-layer protocol to construct an aggregation tree. Constructing the aggregation tree is essentially a routing layer task, but our protocol also involves the application layer, as it considers which nodes have relevant data, and the MAC layer, as it considers the transmission schedule of the nodes.

II. PROBLEM FORMULATION

Consider a multi-hop WSN monitoring an area in which randomly located events occur sporadically and need to be reported to a special node referred to as the sink within D_t after the event, where D_t is the maximum tolerable delay. Only some sensor nodes, referred to as data sources, detect each event, and they generate moderate data volumes. The data of the sources are highly correlated, so the neighboring nodes can and should aggregate these data before forwarding them towards the sink. We say an aggregation protocol is structured if it builds a routing structure prior to any data transmission and we call it unstructured otherwise.

The radio transceivers of the sensor nodes are turned off most of the time in order to achieve battery lifetimes of a few years. They check periodically for incoming messages (data), consuming energy E_{check}, and then sleep during the period T_{sleep}.

The data reporting process consists of building an aggregation structure and transmitting the data over that structure. Let (D_{build}, E_{build}) and (D_{tx}, E_{tx}) be the time and energy of each of these processes. The total delay also involves the delay due to the duty-cycled operation before the building process begins, which can be as high as T_{sleep}. The average power consumption is

$$P = \frac{E_{check}}{T_{sleep}} + \frac{E_{build} + E_{tx}}{T_{event}}, \qquad (1)$$

where T_{event} is the average interval between events. When T_{event} is large, the power of the periodic channel checks dominates P, and we can reduce it by increasing T_{sleep}. The longest value of T_{sleep} that allows reporting the event within D_t is $D_t - D_{build} - D_{tx}$. Therefore, to save power we have to reduce D_{build} or D_{tx}.

This work is funded by UK EPSRC Research Grant EP/D076838/1, entitled: "Smart Infrastructure: Wireless Sensor Network System for Condition Assessment and Monitoring of Infrastructure".

There is a tradeoff between D_{build} and D_{tx}. Reducing D_{build} usually comes at the cost of finding a aggregation structure with degraded performance, which increases D_{tx}. Our goal is to achieve a good balance between D_{build} and D_{tx}.

III. Related Work

The optimal data aggregation structure is the Steiner Tree in graphs, which is NP-hard and difficult to approximate, which is what structured protocols attempt to do. Most of them compute the tree centrally in the sink. Oceanus [3] is one of them. It requires the sink to find out each node's neighbors, which in big networks strains the batteries of the nodes close to the sink. Because its expected applications are for periodic flows, it fails to address some of the challenges of event-triggered applications. It does not specify how the sources report themselves to the sink, how the sink distributes its decisions, or what the cost of the whole process is. We discuss these problems later.

Unstructured aggregation protocols, such as DB-MAC [4] and DAA+RW [5], specifically target event-triggered WSNs with small amounts of data. They do not build an aggregation structure in order to suppress D_{build}. Each node chooses as its next hop its neighbor holding the greatest number of packets according to the information it gathered by snooping its neighbors' transmissions. The main drawback of unstructured protocols is that they handle the timing requirements of aggregation poorly. The sensor nodes hold their packets for some time before relaying them to the next hop, so that, if they receive packets during this time, they will be able to aggregate them with the packets they are holding. This mechanism does not guarantee that a node will hold the packets for long enough to aggregate as many packets as possible and introduces artificial delays. These delays affect every packet, as the nodes handle each packet independently. For sources generating over a dozen packets, unstructured protocols are less efficient than structured protocols that invest some time creating an aggregation tree initially in order to aggregate more packets later and suppress the artificial delays.

The existing unstructured aggregation protocols and the tree construction phase of the existing structured protocols neglect the delays introduced by the duty-cycled operation. However, for sleep periods T_{sleep} longer than a second, these delays may exceed the transmission time. DMAC [6] addresses this problem. It organizes the nodes in tiers, as illustrated in Figure 1. A node's tier is the length of its shortest path to the sink. All the nodes are time synchronized and the activity schedules of nodes in different tiers are staggered. This greatly minimizes the sleep latency towards the sink. However, DMAC does not manage the data aggregation process, as it is only a MAC protocol. In addition, we cannot accommodate the abovementioned protocols directly on top of DMAC because DMAC only allows the data to flow towards sink and because its timing policy does not let a node wait for all the data from its children.

IV. The Fast Aggregation Tree (FAT)

We propose the FAT method to quickly and distributedly construct an aggregation tree after each event. It uses DMAC's tiered architecture. In our discussion, when considering Tier i, we refer to Tier $(i+1)$ as the previous tier and to Tier $(i-1)$ as the next tier. In the network initialization, each node executes the distributed Bellman-Ford algorithm to compute its tier and its neighbors in the next tier. All the nodes in the network share a common time reference. This requires periodical synchronization to compensate for clock drifts, either by software or hardware [7].

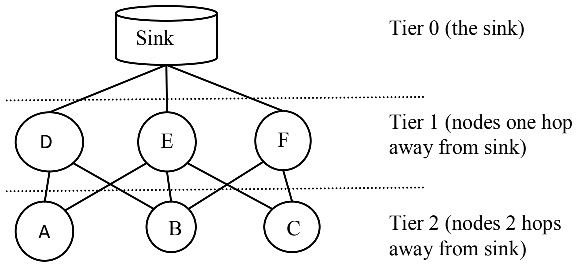
Sink
Tier 0 (the sink)
D
E
F
Tier 1 (nodes one hop away from sink)
A
B
C
Tier 2 (nodes 2 hops away from sink)

Figure 1. Organization of the nodes in tiers of the FAT method. The solid lines link each node with all its neighbors with whom it can communicate.

The nodes check periodically the channel state, which is a process that lasts for the channel sensing time T_{check}. If the channel is idle, they turn their transceivers off and wait for T_{sleep} before checking the channel state again. The checks of nodes in the same tier are simultaneous, and the checks of nodes in neighboring tiers have an offset of T_{tier} as shown in Figure 2.

When some nodes detect an event, they become sources. The sources wait until the channel check time of the nodes in the next tier and transmit signals referred to as *activation tones*, thereby triggering the tree construction process. A node performing a channel check may receive several activation tones simultaneously, but this is fine. The node performing the channel check will sense the channel busy and remain active for a period of duration $T_{tier}-T_{check}$ after the check. During this period, the nodes that performed the check will be looking for children and the sources will be looking for a parent.

The nodes looking for a parent back off for a random time before transmitting a Parent Request (PR) to the node that they wish to have as a parent. The nodes that lost the contention remain quiet and snoop the Parent Confirmation (PC) that responds to the PR of the node that won the contention. Every node looking for children that receives a PR always replies to it with a PC and adds the source of the PR to its children list, which was empty before the reception of the activation tone.

Each node looking for a parent transmits PRs until it receives a PC or its period to do so ends. It may only choose the recipient of its PRs among its neighbors in the next tier. Initially, it makes that decision randomly, but it may change it until it receives a PR. At each point, in order to promote early data aggregation, it chooses the node from which it has overheard more PCs. If the chosen node fails to reply to several PRs, the link to that node is likely to be poor and the node looking for a parent changes its decision.

When a node's period for looking for children ends, it counts the number of nodes in its children list. If it is zero and it does not have any data, the node considers itself unnecessary in the reporting process and it sleeps until its next normal

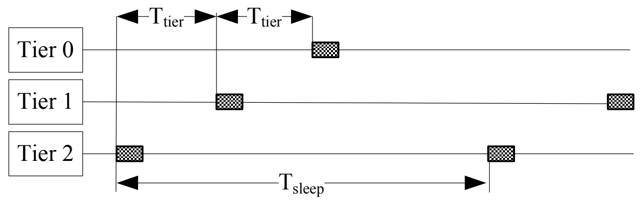

Figure 2. Staggered schedule of the FAT method, whereby the channel checks of nodes in consecutive tiers are shifted by T_{tier}.

channel check, which will occur $T_{sleep}-T_{tier}$ later. Otherwise, it transmits an activation tone immediately, because, due to the staggered schedule, the nodes in the next tier are checking the channel state exactly then. After transmitting the activation tone, it looks for a parent in the next tier as its children did. The tree construction process continues in this way without any delay due to duty-cycling until the nodes in Tier 1 that have data in their subtree have found a parent, which for nodes in this tier is necessarily the sink. The sources can start transmitting data over the tree as soon as the construction process has moved far enough from them so that they will not interfere with it.

Let us illustrate the tree construction process with an example based on the topology of Figure 1. Let nodes A, B and C be the only sources. They simultaneously transmit activation tones when the nodes in Tier 1 check the channel, causing nodes D, E and F to start looking for children. Let us consider the case in which nodes A, B and C initially choose nodes D, E and F, respectively, as the recipients of their PRs. Nodes A, B and C contend to transmit their PRs. If B wins the contention, E replies to B's PR with a PC. Both A and C overhear the PC and set E as the recipient of their PRs and they proceed to transmit their PRs. Eventually, nodes A, B and C obtain E as a parent, achieving early aggregation. However, this would have not been the case if node A had been the first node to transmit a PR, because the constructed tree would not aggregate the data of the three sources in Tier 1.

T_{tier} is the same for the entire network. If it is too long, nodes looking for children wait unnecessarily long for PRs. If it is too short, some nodes may not have time to find a parent. A node also fails to find a parent if all the links to its neighbors in the next tier fail. In both cases, we say that the *normal construction* of the tree has failed and our protocol resorts to a more powerful mechanism to construct the tree that we term *backup construction*. Every node that has not found a parent and whose slot to do so is about to expire transmits a so-called *emergency tone*, which triggers the backup construction. Every node looking for children checks for emergency tones at the end of its slot to look for children. If it senses the medium idle, it proceeds with normal construction. Otherwise, it sends an activation tone and an emergency tone when the nodes in the previous and next tier expect them. The goal is to propagate the emergency tones to all the nodes in the network. Every node that received an emergency tone remains active for enough time to execute a reactive routing protocol such as Ad-hoc On-Demand Distance Vector (AODV). This process ensures that all the sources find a path to the gateway if it exists.

V. Performance Evaluation

A. Analysis of the Tree Construction Delay

Let k be the number of the outermost tier that contains a data source and K be the total number of tiers. The construction delay D_{build} for the FAT's method normal construction is kT_{tier}. For the backup construction, D_{build} is much higher and it consists of the propagation time of the emergency tones and the execution time of AODV. The first delay can be as high as $K(T_{sleep}-T_{tier})$, because the propagation of an activation tone from one tier to the previous one involves almost a full sleep period, as can be deduced from Figure 2.

Centralized tree construction protocols for aggregation involve three steps. First, the sources report themselves to the sink. Second, the sink computes the tree. Third, the sink distributes this information to the nodes. The durations of the first and third steps depend on the MAC layer. Centralized protocols fail to study these durations and we do this now with the MAC protocols we believe to be best suited for them. The first and third steps involve flows of data in opposite directions. We claim that this precludes sleep delays as short as those of the normal construction of the FAT, which only involves traffic in a single direction, namely towards the sink. If the centralized protocols use a MAC with globally synchronous wakeup, e.g. T-MAC [8], each of the first and third steps involve k sleep periods. Using a staggered schedule such as DMAC's, the first step involves no sleep periods, but the third steps involves roughly k sleep periods. In contrast, the normal construction of the FAT method involves no sleep periods.

B. Aggregation Tree Quality

A better aggregation tree compresses more information inside the network, thereby reducing the transmission time D_{tx}. We wrote a custom simulator to compare D_{tx} using the FAT with D_{tx} using the trees computed with the following techniques:

- Shortest Path Tree (SPT): This is the shortest path to the sink, which is the default path. As such, it is not adapted to maximize aggregation for the sources related to a specific event.
- Dijkstra1: It is an enhancement of Dijkstra's algorithm to promote aggregation. It uses the number of hops as the cost metric. When choosing deciding the next node to add to the tree, it prefers data sources to nodes without data if their cost is the same. When choosing the next hop for the new addition, it prefers nodes with data if the cost is the same.
- Centralized1: It constructs with Dijkstra1 the tree rooted in the node q closest to the event. Then, it removes all the nodes that are neither sources nor in the path from a source to q. Finally, it links q with the sink through the shortest path. In practice, q may be unknown.

In order to compare fairly the transmission time with all these trees, we use the same MAC protocol for all of them. We choose CSMA although other protocols may be more efficient. We also simulate the DMAC protocol, which provides very short sleep delay and involves no tree construction overhead, to serve as benchmark of the achievable performance without data aggregation.

We simulate 50 nodes randomly deployed over a 200m by 200m rectangle. The transmission and interference ranges are 60m and 150m, respectively. The sink is at one corner of the rectangle. The event occurs in the other corner and the $s = 12$ nodes closest to it become sources. Each source generates ten packets. The transmission time of a packet containing the data of n sources is

$$D_{agg} = D(1+\alpha(n-1)), \qquad (2)$$

where D is 8ms and α is a parameter between 0 and 1. For α = 0, we can compress any number of packets into one packet as short as one containing only one node's data. For α = 1, no compression is feasible. Figure 3 reveals that the performance greatly varies with α. The results may differ widely for other aggregation models. For α = 0, Centralized1 is a reasonable approximation of the optimal solution, which is the Steiner tree. The FAT performs about 20% worse than Centralized1, but it offers a similar improvement over the SPT. The FAT method achieves this improvement without having a longer setup when there is not node failure. This is because in the SPT method each node needs to find out which of its children have data, so that it waits for their data in the transmission phase. The nodes can gather this information very efficiently by using the same MAC as the FAT method, but without updating the recipients of their PRs based on the information they overhear. Dijkstra1 approximates the tree our protocol would obtain if every node, rather than making an initial random choice of the recipient of its PR, made the optimal choice. We do not plot the results of Dijkstra1 because they are very similar to those of the FAT, indicating that the randomness of the initial parent choice hardly degrades the tree quality. For α = 1, aggregation is unfeasible and the SPT is the optimal tree. DMAC outperforms the simulation labeled as SPT, although both use the same tree, because the latter uses CSMA, which does not multiplex transmissions spatially as good as DMAC. Figure 4 shows that the benefit of good aggregation increases with the number of sources s.

C. *Vulnerability to Outdated Topological Information*

The normal construction of the FAT method always finds a path to the sink if it exists and every node's tier number and list of nodes in the next tier are up-to-date. The nodes may refrain from continuously updating this information in order to save energy, but outdated information may force to resort to the backup construction, which is slower and more energy consuming. Therefore, it is not clear whether it pays off to update the topological information. In order to address this question, we obtain here f_{normal}, which we define as the probability that a source will fail to find a path to the gateway using the normal construction when the nodes are unaware of which nodes have failed.

Let each node's failure probability be f. In a regular deployment wherein every node has n neighbors in the next tier and the sources are h hops away from the sink, f_{normal} is $1-(1-f^{\,n})^{h}$. Therefore, in order to reduce the probability of resorting to backup construction f_{normal}, we have to increase n or decrease h.

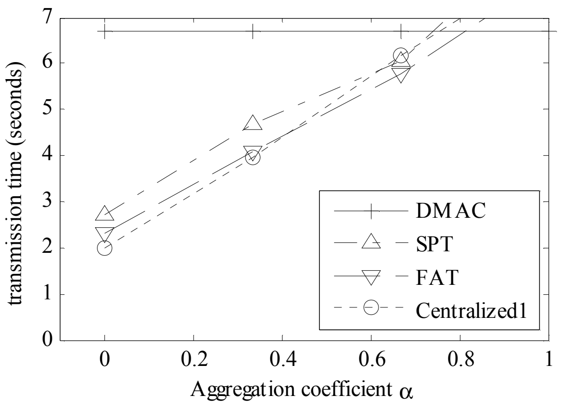

Figure 3. If little aggregation is feasible (α ≈ 1), there is little benefit in aggregating data. For small values of α, FAT outperforms DMAC and SPT and approaches Centralized1.

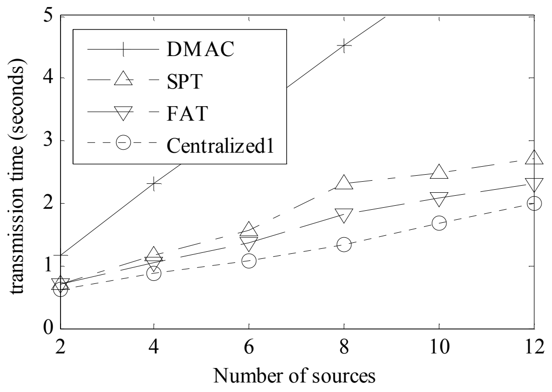

Figure 4. FAT's advantage over both SPT and DMAC increases with the number of sources. We obtained this graph for $\alpha = 0$.

We use simulations to evaluate f_{normal} in a random deployment with node density $\rho = N\pi r^2/A$, where N is the number of nodes, r is the transmission range of the nodes and A is the area being monitored. We place the nodes randomly, but, if more than 20% of the nodes are disconnected, we discard the deployment and try again. This requires few attempts for $\rho \geqslant 7$. The monitored area is a rectangle whose height is $4r$. The sink and the event center are a distance d away from each other. They are in the middle of the left and the right borders of the rectangle, respectively. Figure 5 compares the probability of not finding a path to the gateway for three techniques, namely the default tree (T1), the FAT method's normal construction (T2), and the backup construction (T3). The probability of not finding a path for T2 is f_{normal}. We see that as the number of hops increases, the damage of outdated topological information increases. This is because more hops imply a greater number of potential failure points. For ρ = 7, the probability that failed node disconnect the network (T3) is significant. However, for higher densities, the path exists but T2 and T1 do not find it.

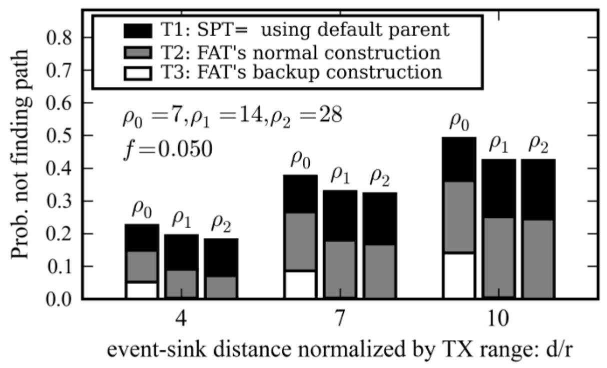

Figure 5. Fraction of nodes that fail to find a path in 3200 simulations of randomly deployed WSNs with node failure probability *f*. For three different event-sink distances and for three different node densities ρ, the figure compares techniques T1, T2 and T3. For each technique, the obtained value should be read from 0 to the upper location of the bar, as opposed to from the difference between the locations of the lower and upper borders of the bar. This representation is possible because T3 always outperforms T2, and T2 always outperforms T1.

VI. CONCLUSIONS AND FUTURE WORK

We have proposed the FAT method, a distributed protocol for WSNs that constructs an aggregation tree to report data generated by events whose location and timing are unpredictable. Its main advantage is that, when the sensor nodes' sleep period is long, it constructs the tree *k* times faster than centralized approaches, where *k* is the tier number of the sources. This is because of its staggered schedule and because it only generates data in the direction towards the gateway. Our protocol is a cross-layer one because the routing decision considers the transmission schedule of the nodes in order to accelerate the construction process.

In our simulations, our protocol constructed trees significantly better than the shortest path tree, because it customizes the tree to the data sources of each event. This is only true when the nodes can compress the data intensely. We implemented a centralized algorithm that constructed a tree substantially better than the FAT. The performance gap is due to our tiered architecture, which restricts potential parents to the neighbors in the next tier.

The FAT method's normal construction always finds a path if it exists and the nodes have up-to-date topological information. If this information is outdated, our protocol can trigger the backup construction, which incurs high overhead. It is possible that our protocol will find a path if the nodes have outdated information. However, our simulations reveal that this is unlikely for events occurring seven hops away from the sink or further, and increasing the node density hardly helps. Therefore, each node needs to monitor periodically the functioning of its neighbors. In the future, we aim to investigate how to do this with little overhead and how to organize transmissions after the tree construction to maximize concurrency and reduce interference.

REFERENCES

[1] E. Fasolo, M. Rossi, J. Widmer, and M. Zorzi, "In-network aggregation techniques for wireless sensor networks: a survey," *IEEE Wireless Communications*, vol. 14, no. 2, pp. 70-87, 2007.

[2] WINES consortium, "Wired and Wireless Intelligent Networked Systems (WINES) - Smart Infrastructure Project," www.WINESinfrastructure.org, 2008.

[3] A. F. Harris III, R. Kravets, and I. Gupta, "Building trees based on aggregation efficiency in sensor networks," *Springer Ad Hoc Networks*, vol. 5, no. 8, pp. 1317-1328, Nov 2007.

[4] G. di Bacco, T. Melodia, and F. Cuomo, "A MAC protocol for delay-bounded applications in wireless sensor networks," in Proc. *Med-Hoc-Net 2004*, Bodrum, Turkey, June 2004.

[5] K.-W. Fan, S. Liu, and P. Sinha, "Structure-free data aggregation in sensor networks," *IEEE Transactions on Mobile Computing*, vol. 6, no. 8, pp. 929-942, 2007.

[6] G. Lu, B. Krishnamachari, and C. S. Raghavendra, "An adaptive energy-efficient and low-latency MAC for data gathering in wireless sensor networks," in Proc. *IEEE 18th International Parallel and Distributed Processing Symposium (IPDPS '04)* 2004, p. 224-235.

[7] R. Mangharam, A. Rowe, and R. Rajkumar, "FireFly: a cross-layer platform for real-time embedded wireless networks," *Springer Real-Time Systems*, vol. 37, no. 3, pp. 183-231, Dec 2007.

[8] K. Langendoen, "Medium access control in wireless sensor networks," in *Medium Access Control in Wireless Networks, Volume II: Practice and Standards*. H. Wu and Y. Pan, Eds. Nova Science Publishers, 2008.