Rapid and Energy Efficient Data Transmission Technique using Event Aggregation in WSN

Raxit G. Jani[1] Prof. Vaseem Ghada[2]
[1]M.Tech Student, Information and Computer Technology
[2]Assistant Professor, Computer Engineering Department
[1]Institute of Technology, Nirma University
[2]B.H. Gardi College of Engineering & Technology, Gardi Vidhyapith, Rajkot

Abstract--- In this Paper we analyze WSNs and show that it's possible to achieve better performances in terms of energy consumption and latency. Event aggregation in WSNs is a process of combining several low-level events into a high-level event to eliminate redundant information to be transmitted and thus save energy. Existing works on event aggregation consider either latency constraint or aggregation function, but not both. Moreover, existing works only consider optimal aggregation for single high level event, but many applications are composed of multiple high-level events. This paper studies the problem of aggregating multiple high-level events in WSNs with different latency constraints and aggregation functions. We propose relation matrix to define aggregation function, which describes the similarity among limited number of primitive events rather than the growing number of high-level events. Based on it, we propose an event aggregation algorithm jointly considering the two issues for single high-level event. This algorithm supports partial aggregation which is more general than fully aggregation. Through selection the optimal base events, the work is extended to multiple high level events and consider the practical reliable constraint. The simulation results show that our algorithm outperforms existing approaches and saves significant amount of energy (up to 35% in our system).

I. INTRODUCTION

Wireless Sensor Network (WSN) has developed in recent years. WSN because of its edibility in arrangement as well as the less effort demanded for maintenance, have exhibited promising applications in many fields like military, healthcare, environmental applications, etc. WSN comprises of large number of tiny sensor nodes. Because of their small size and use of wireless medium for communication these nodes can be deployed in the phenomenon or close to it. These sensor nodes because of its size have some limitations. They have limited computation power, memory, communication capabilities and energy. WSNs have been extensively studied with the objective of energy efficiency whereas throughput, bandwidth utilization, fairness and latency were considered as the secondary objectives. One of the most prominent operations of WSN is converge casting. Converge cast, namely the collection of data from a set of sensors toward a common sink over a tree-based routing topology, is a fundamental operation in wireless sensor networks (WSN). Two types of data collection: (i) aggregated converge cast where packets are aggregated at each hop (ii) raw-data converge cast where packets are individually relayed toward the sink. Aggregated converge cast is applicable when a strong spatial correlation exists in the data, or the goal is to collect summarized information such as the maximum sensor reading. Raw-data converge cast, on the other hand, is applicable within each sensor reading, is equally important, or the correlation is minimal. Aggregated converge cast also results in energy conservation and one of the most popular techniques used. In this it reduces the number of packets to be transmitted from source to sink which saves the energy of transmitting each packet individually. This process does increase the latency of communication. The collection of data formal set of sensors toward a common sink over a tree based routing topology, called as converge cast, is a fundamental operation in wireless sensor networks (WSNs).it is essential to provide a guarantee on the delivery time as well as increase the rate of such data collection in many applications. For instance the applications in which the actuators and the controllers need to receive data from all sensors within a specific deadline, failure of which might lead to unpredictable and catastrophic events. This falls under the category of one-shot data collection. On the other hand, applications such as permafrost monitoring require periodic and fast data delivery over long periods of time, which falls under the category of continuous data collection. However since various sensor nodes often detect common phenomena, there is likely to be some redundancy in the data that the source nodes communicate to a sink. In-network filtering and processing techniques like data aggregation can help to conserve the scarce energy resources. In this paper we study the energy savings and the traffic reduction at the expense of latency that can be obtained by introducing data aggregation in the described scenario.

We also show how such performances are affected by factors such as the radio range of the nodes, the number of nodes, the Maximum out-degree of the nodes in the broadcast tree. Most previous works on data aggregation have mainly focused on energy saving issue. Building an energy efficient topology (usually a tree construction) has been investigated. Tree topologies are often employed for wireless sensor networks not only because their structure is suitable for a network with one or a few sink nodes, but also because their simplicity is very attractive when network resources are limited. It has been shown that ending the optimal aggregation tree that minimizes the number of transmissions or maximizes the network lifetime is an NP-Hard problem and efficient approximation algorithms have been provided for constructing such trees. This is not the focus of our paper. Instead, we assume that an aggregation tree has been provided. Given a tree topology, we study the impact of in-network aggregation on the delay performance of wireless scheduling.

II. EVENT AGGREGATION

A. *Event & Data:*

Early works in WSNs are mainly about data processing and the latest ones begin to discuss event processing. Event processing is a natural extension of data processing in WSNs, which encapsulates collected data in events. Correspondingly, while data aggregation generates the summary of raw data, event aggregation deduces high-level events from low-level events. Event aggregation is an effective approach to reduce data amount in WSN communication. In existing works, some analyze the impact of aggregation functions on aggregation structure, where aggregation function describes the correlations among different events. Others consider latency constraints to meet time sensitive requirements. An aggregation with both of the factors is more general but has not been considered yet.

The difficulty partly arises from the aggregation function which is usually assumed as fully aggregation function which means two low-level events with one unit data amount can merge into a composite event with one unit data amount, or more general function which takes low-level events as input and composite event as output. It is not clear how to combine aggregation function with latency constraint. More importantly, hardly any existing works have been conducted for multiple high-level events. Considering a WSN based intelligent traffic system, many events are required by different users for different purposes. Some users may have interests in the average speed of vehicles, while some others may want to know the speed of individual vehicles. The events have different correlations among low-level events hence have different aggregation functions. Moreover, the latency requirements also may be different. Existing approaches have not investigated the diversity of requirements in event aggregation.[3] In this paper, we investigate the aggregation problem of multiple high-level events with different latency constraints and aggregation functions in WSNs. We propose relation matrix as a simple approach to define aggregation function, which use the similarity among limited number of primitive events rather than growing number of high-level events. After that we design algorithm named DBEA to build the aggregation tree for single high-level event considering both the two issues.

This algorithm supports partial aggregation which is more general than fully aggregation. The convicting optimal parent candidates in building the tree are also considered. Finally, we use dynamic programming to select some of the high-level events as base events subject to a reliable constraint, and then build the aggregation tree using DBEA. Firstly, proposed a simple approach to describe aggregation function and enable them to combine with latency constraint. Secondly, considered both latency constraint and aggregation function in individual high-level event aggregation. Thirdly, extended the approach for multiple high-level events aggregation and consider the practical reliable constraint.

Two issues have influence on event aggregation: latency constraint and aggregation function. In this paper, latency is defined as the time duration from the time point when the first event is sent at a source node to the time point when the last event is received by the sink node. In a usual WSN application, the latency is mainly determined by communication distance. When latency constraint is tight, the aggregation tree trends to be the shortest path tree and little opportunity to undertake the aggregation. By contrast, if latency constraint is loose (e.g. no latency constraint), optimal aggregation can be achieved. Aggregation function also affects event aggregation, which describes correlations among the events and hence the data amount reduction. A larger data amount reduction means the event is more sensitive to event aggregation. Existing works encounter two problems when building the aggregation tree. Firstly, they considered only aggregation function but no latency constraint. Secondly, they cannot be directly adopted for multiple high-level events with different latency constraints and aggregation functions. For the first problem, in later section we will introduce an solution for single event.

For the second problem, several straight forward approaches can be proposed but all have deficiencies. The first one is to build the aggregation tree every time an event comes which results in unacceptable overhead and latency. The reasonable approach is to build a number of m aggregation trees in advance according to a number of m latency constraint-aggregation function pairs, where m is determined by available data memory in a sensor node. For simplicity, m events are selected as base events to acquire the m latency constraint-aggregation function pairs. We call the latency constraints of base events as base latency constraints, and the aggregation functions of base events as base aggregation functions. In reality, all the events with base latency constraints are required to meet an reliable requirement. For example, if li is a base latency constraint, all the events with li need bound their performance degradation to some degree.[3]

B. *Problem Formulation:*

The Event Aggregation with Different Latency Constraints and Aggregation Functions (EALA) problem is formulated: Given: a. A wireless sensor network G(V,E), a set of source nodes S and a sink node r. b. A high-level event set CE involving n composite events ce (i = 1...n) which need detected. Each composite event has latency constraint li and aggregation function ag. c. At most m composite events cev1...cevm can be selected as base events to construct the aggregation tree. and a aggregation tree which aggregates all composite events to the sink node with minimal message traffic cost. Subject to a. Each composite event meets latency constraint b. All the events with base latency constraints meet the reliable requirement.

C. *Delay bounded event aggregation algorithm:*

We propose Delay Bounded Event Aggregation algorithm (DBEA) to build the aggregation tree considering both latency constraint and aggregation function. Different from fully aggregation, DBEA supports partial aggregation.

During the tree building, if a source node joins in the tree, the distance increased is not the distance between this node and the tree, but the distance between this node and the sink node.

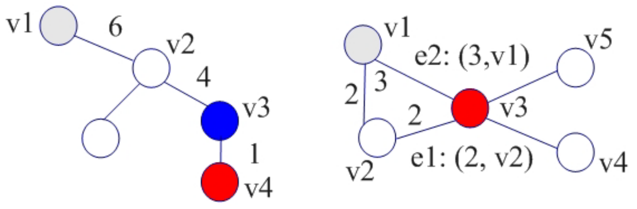

Fig.1. Design rationale of delay bounded event aggregation algorithm

For example, in Figure 1, aggregation tree T already includes v1, v2 and v3; the distance between v4 and T is 1, which is exactly the increased distance in fully aggregation since when a packet traverses from v4 to v3, it merges into the packet of v3 hence not increase the cost from v3. This is not the case in partial aggregation, the packet traversing to v3 may aggregate into a packet with data amount more than 1(eg. 1.5), then it has additional cost from v3 to the sink node (eg. (1.5 1)(6 + 4) = 5)). We also introduce the concept of possible distance, because different events may have convicting optimal parent candidates. As in Figure 1, v1, v2 and v3 are already in T; When event e1 transmits to v3, the optimal distance is 2 subject to the parent of v2, while for event e2, the optimal distance is 3 subject to the parent of v2. In the aggregation tree, v3 have only one parent. This is so called possible and need adjustment when a source node joins in the tree. The details are shown in Algorithm 1. We randomly create networks with minimum latency 4 and aggregation function N0(norm of A2 is 0, norm=0 for short), then issue a set of events with increasing latency constraints from 0 to 25.

We use DBEA to build aggregation tree and calculate the maximum latency. The experiment is repeated with different aggregation function N1 (norm = 1), N2(norm = 2) and N3(norm = 3). The result and each point represent 50 times executions. All the results are between the minimal latency and latency constraint, which shows all latency constraints are strictly meet, at the same time the approach gets benefit compared with the approach simply selecting the minimal latency. Growing benefit goes with increasing aggregation function. The aggregation tree with N3 is longer than all others. For event 12, the latency of aggregation tree with N3 is 13 while that with N0 is only 8. The aggregation characteristic is recognized in the algorithm and generate a longer aggregation tree hence more opportunities for performance improvement. The simulation results show that significant energy (up to 35% in our system) can be saved by using our algorithm.

III. DELAY PERFORMANCE OF SCHEDULING WITH DATA AGGREGATION

Most previous works on data aggregation have mainly focused on energy saving issue. Building an energy-efficient topology (usually a tree construction) has been investigated. Tree topologies are often employed for wireless sensor networks not only because their structure is suitable for a network with one or a few sink nodes, but also because their simplicity is very attractive when network resources are limited. It has been shown that ending the optimal aggregation tree that minimizes the number of transmissions or maximizes the network lifetime is an NP-Hard problem and efficient approximation algorithms have been provided for constructing such trees. This is not the focus of our paper. Instead, we assume that an aggregation tree has been provided. Given a tree topology, we study the impact of in-network aggregation on the delay performance of wireless scheduling. Another group of works have investigated the energy latency trade-off. They have dealt with energy efficiency of wireless sensor networks constrained on the maximum delay of sensed data. Given a deadline, they minimize overall energy dissipation of sensor nodes, minimize the maximum energy consumption or minimize the amount of missed data. we formulate the problem by minimizing the total delay of sensed data, and thus reduce the average delay. Associated with in network aggregation, scheduling also needs to be considered since one has to determine which node forwards data, and which node holds data for aggregation, which might improve the performance in the future by reducing the amount of bits in-right. The problem is further exacerbated by interference in the wireless environment. Our main contributions are as follows:

We formulate the scheduling problem that minimizes the delay sum of data in wireless sensor networks, and qualify the gain of in-network aggregation techniques for tree networks. We study the characteristics of optimal scheduling with aggregation, and analyse the optimal performance by providing a lower bound. We evaluate the performance of scheduling policies against the delay bound, and show that the performance can substantially improve by considering the network state one-step ahead, i.e., by estimating the network state of the next time slot.

A. *Network Model:*

We consider a tree network T(V,E), where V denotes the set of n wireless nodes and E denotes the set of wireless links. One sink node denoted by vs is located at the root of the tree, and collects data from (n - 1) sensor nodes. The task of the network is to compute a function from the sensed data, and to deliver the function value through the sink node. Data is forwarded via hop-by-hop communications to save energy, which costs a unit of transmission time per forwarding. The function should allow in-network aggregation from a partial collection of data without changing the final results or increasing transmission cost (time) for forwarding aggregated data. Common examples of such functions include min, max, average, histogram, etc. We assume a time-slotted system with a single frequency channel for the entire system. Channels are assumed error free. However, there is wireless interference between channels, which is modelled by the well-known node-exclusive interference model1. Under this model, two links sharing a node cannot transmit at the same time. The model is suitable for Bluetooth and FH-CDMA networks. At each time slot, a link can transmit a single packet, and multiple links can transmit simultaneously only when they are greater than one hop away from each other. It is also denoted by the 1-hop interference model or the primary interference model.[5]

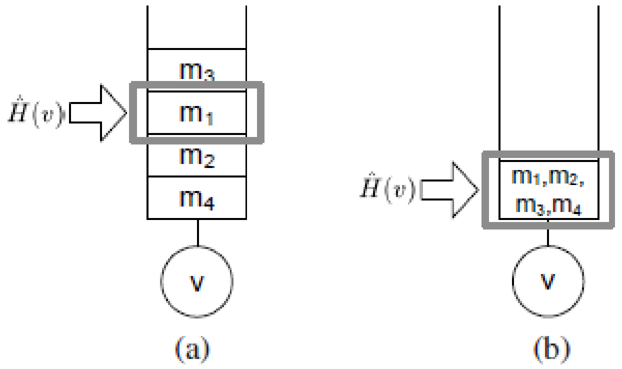

Fig.2. Messages in the queue of node v with and without aggregation. Without aggregation, when node v is scheduled, the message m1 will be forwarded. With aggregation, all messages are aggregated into a single packet and will be forwarded simultaneously. (a) Without in-network aggregation. (b) With in-network aggregation.

The overall system operations are described as follows. Assume that a measurement event occurs at time t = 0. Each node senses its environment and generates data, which is denoted by a message. All messages from sensor nodes have to be delivered to the sink. During the first time slot, each message generated at node v 1) is contained in a packet for transmission and forwarded to the parent node of v, 2) stays at node v due to transmissions from an interfering neighbour, or 3) waits at node v for further aggregation. The parent node that receives the packet may forward the message to its own parent node in the next time slot. After the first time slot, some of the nodes may have multiple messages. Under certain sensor network applications, nodes with multiple messages can compress the data and put them into a single packet without affecting the final results. This procedure is called as in-network aggregation. For example, suppose that the task of the network is to compute an average of the messages obtained from sensor nodes. If we format the content of a packet in a tuple of (the average of messages, the number of averaged messages), an intermediate node can compute an average of all the messages that it has received or generated, and package the computation result into a single tuple without losing any information.

We assume that aggregation is permanent and messages that are aggregated will not split later. At each time slot, the process of transmission and aggregation is repeated until the sink collects all the messages. Clearly, aggregation will improve scheduling efficiency by reducing the number of packet transmissions. We assume that if in-network aggregation is allowed, it can be done with a negligible cost of energy or computing power. If aggregation is not allowed, we assume that one packet can contain only one message. Our objective is to minimize the total delay in collecting all the messages related to a particular measurement event. Let V denote the set of all messages of the event. Each message is generated at a sensor node during t = 0 and no message is generated for t > 0. Messages do not need to measure the same physical environment, and may have a different meaning with a different importance, in which case, we quantify the importance of each message m using a weight wm > 0. Let fm(t) denote the node where message m is located at2 time t. We denote the system state at time t by its vector f(t) := fm(t). Also let S(t) denote the schedule during time slot t, which is the set of nodes that transmit during time slot t. Note that each node v . S(t) will transmit a packet to its parent node p(v), and those set of active inks (v, p(v)) will correspond to a matching under the node-exclusive interference model. Let S denote the set of all possible schedules. Given a system state, let H (v) denote the set of messages that can be transmitted when node v is scheduled. Without aggregation, the set H (v) will have a single message, which might be the one with the largest weight among the messages located at node v. (See Fig.3.1) With aggregation, we assume that H(v) includes all messages at node v. Use two spaces after periods (full stops). Hyphenate complex modifiers: “zero-field-cooled magnetization.” Avoid dangling participles, such as, “Using (1), the potential was calculated.” Write instead, “The potential was calculated using (1),” or “Using (1), we calculated the potential.” Use a zero before decimal points: “0.25,” not “.25.” Use “cm^3,” not “cc.” Do not mix complete spellings and abbreviations of units: “Wb2/m2” or “webers per square meter,” not “webers/m2.” Spell units when they appear in text: “…a few henries,” not “…a few H.” If your native language is not English, try to get a native English speaking colleague to proofread your paper. After time slot t, the system state will change as follows:

$$f_m(t+1) = \begin{cases} p(f_m(t)) & \text{if } f_m(t) \in S(t) \\ & \text{and } m \in \hat{H}(f_m(t)), \\ f_m(t) & \text{otherwise.} \end{cases}$$

Let tm denote the time when message m arrives at the sink vs, i.e., tm := min ft | fm(t + 1) = vsg. Multiple messages, if aggregated, can arrive at the sink simultaneously and have the same arrival time. Our objective is to solve the following problem:

$$\underset{\{S(t)\in\mathcal{S}\}}{\text{minimize}} \sum_{m\in\hat{V}} w_m t_m$$

We rewrite the above problem using the delays that each message experiences. Let S(t) denote the set of messages that are scheduled during time slot t, i.e., S(t) := m | fm(t) S(t),m H(fm(t)). Further, let h(v, v') denote the number of hops between two nodes v and v', and let hm denote the number of hops between the initial location fm(1) of message m and the sink vs, i.e., hm := h(fm(1), vs). Note that hm is the least number of time slots for m to arrive at the sink when there is no other message. When multiple messages coexist in the network, two packets that contain a different message can compete with each other for scheduling resources. While a node is transmitting a packet, its neighbouring nodes cannot transmit simultaneously due to wireless interference and the messages in those nodes have to be delayed staying at the current location. Let Dm denote the delays experienced by each message m in the network until it arrives at the sink, i.e., Dm := tmhm. Since is a constant, (2) is equivalent to

$$\underset{\{S(t)\in\mathcal{S}\}}{\text{minimize}} \quad \sum_{m\in\hat{V}} w_m D_m$$

At each time slot, we have to determine which message m has to be forwarded, and which should stay for aggregation. Hence, in-network aggregation results in a trade-off between scheduling efficiency and delays. Remark: The problem (3) is similar to the scheduling for minimum delay, where the delay optimal scheduler for linear networks has been provided. Although there is a similarity in the objective, the problem becomes much more complex here because the topology is richer (a tree topology), and we have to deal with aggregation. Hence, the solution of is not applicable.

B. Impact Of Aggregation

In this section, we study the effectiveness of in-network aggregation in terms of the total delay. Given a tree network of n nodes including the sink, we consider a scenario in which each sensor node generates a message, and all (n -1)messages are equally important. Specifically, all weights are set to 1. We show that without aggregation, the optimal delay is lower bounded by O(n2), while it is upper bounded by O(n log n) with aggregation. Therefore, the gain is quite substantial $O(n/\log n)$.

1) Performance without aggregation:

For a given tree network, let d and W denote the maximum depth and the maximum width of the tree, respectively. At time t, a message m is said to be located at depth d if h(fm(t); vs) = d. The following proposition provides a bound on the optimal delay performance.

Proposition 1: Without in-network aggregation, the delay is lower bounded by when d = O(log n). Proof: Since messages cannot be aggregated, each packet has to include only one message. Then, at the root, the sink can receive at most one packet during each time slot under the node-exclusive interference model. Since it takes at least (n - 1) time slots to receive all (n - 1) messages.

$$\sum_{m\in\hat{V}} D_m \geq O(n^2)$$

2) Performance with aggregation:

In-network aggregation can signi_cantly reduce the delays by integrating multiple messages into a single packet. Since the reduction in the number of transmissions implies less interference, messages can be forwarded faster. We assume that messages can be aggregated into a single packet with no cost if they are located in the same node. Proposition 2: With in-network aggregation, the delay is upper bounded by

$$\sum_{m\in\hat{V}} D_m \leq O(n\log n)$$

Proof: We provide a scheduling policy based on link colouring and show its delay bound, which will serve as an upper bound on the delay of the optimal scheduler. Since each node has at most (W + 1) links, we can colour links using (W + 2) different colours such that two links with the same colour are not adjacent to each other. Then we can schedule links with the same colour simultaneously. We choose a colour in a time slot in a round robin manner. Note that under this policy, each node transmits a packet to its parent every (W +2) time slots. We can estimate the maximum delay that a message experiences before it arrives at the sink.[5]

3) Optimal scheduling:

Let A(t) and T(t) denote the set of messages that have arrived at the sink, and are transient in the network, respectively, at time slot t. That is, A(t) = fm | fm(t) = vsg, and T(t) = fm | fm(t) |= vsg. Clearly, we have A(1) = 0 and T(1) = V . Let Dm(t) denote the delays experienced by m during [0, t]. Hence

$$D_m(t) = \begin{cases} D_m(t-1) & \text{if } m \in \hat{S}(t) \cup \hat{A}(t), \\ D_m(t-1)+1 & \text{otherwise,} \end{cases}$$

C. Numerical Results:

We now investigate the performance of several scheduling algorithms versus our delay bound as a performance benchmark. We assume that the objective function, e.g., average of sensed data, conforms to our aggregation constraints, i.e., in-network computations with a partial collection of data do not change the _nal results and do not increase transmission cost of aggregated data. We classify scheduling policies into myopic and non-myopic ones: Myopic policies make a scheduling decision based only on the current system state. Most conventional scheduling algorithms are myopic. No myopic policies simulate the network with a sequence of schedules, and make a scheduling decision for time slot t based on simulation results. For example, a non-myopic policy that can simulate the network up to tmax steps can solve (3) by simulating S(1), . . . , S(tmax).

In our experiments, we focus on binary tree networks. We first examine the optimal performance for full binary trees and move to binary trees that are randomly generated. We also compare the analytical results with the performance of scheduling policies when weights are assigned at random. Number citations consecutively in square brackets [1]. The sentence punctuation follows the bracket [2]. Refer simply to the reference number, as in [3]. Do not use “Ref. [3]” or reference [3]” except at the beginning of a sentence: “Reference [3] was the first …”

1) Myopic Scheduling Policies

For a feasible schedule S S, let D(f(t), S) denote the total delay increment if S is applied to the system state f(t). 13 Since myopic scheduling policies make scheduling decisions based on the current state of the network, we consider the following conventional scheduling algorithms:

– Maximum Weight Matching (MWM): During each time slot t, it schedules the set of nodes that minimize the delay increment of messages for the time slot.

– Maximum Size Matching (MSM): During each time slot t, it schedules the set of nodes that maximize the total number of scheduled nodes with a message.

– Greedy Maximal Matching (GMM): At each time slot t, it sorts nodes based on the sum of message weights in each node, and includes nodes in the schedule S(t) in decreasing order conforming to the inference constraints.

One may expect that MWM will achieve the best performance since it serves the largest weights during the time slot.

2) Non-Myopic Scheduling Policies:

In this section, we propose three non-myopic scheduling policies and evaluate their performance. Under non-myopic scheduling policies, a scheduling decision is based on the current and the future system states, which can be obtained by simulating the network. A higher performance is expected as further future states are taken into account. However, due to computational complexity in implementation, we consider one-step non-myopic policies only. Specifically, we propose the following algorithms:

– Non-Myopic MWM (NM-MWM): During each time slot t, it schedules the set of nodes that will minimize the delay increments during [t, t + 1]

– Non-Myopic MWM with Analysis (NM-MWM+A): It schedules the set of nodes that minimize the future delay bound, which can be obtained using our analysis.

– Non-Myopic GMM (NM-GMM): It is similar with NMMWM except that for each feasible S, it chooses the second schedule S' using the GMM algorithm.

This analysis is accurate, in particular when the network size is small. We also evaluate the performance of several myopic and non-myopic scheduling policies, and show that the one-step non-myopic scheduling policies achieve substantially lower delays than the conventional myopic ones. In particular, the non-myopic GMM seems to achieve good performance with moderate complexity making it attractive as a viable alternative over the optimal solution.[6]

There are many interesting directions for future work. One can obtain analytic results when the underlying tree networks have a larger width, and develop a simple distributed algorithm with high performance. Performance of scheduling policies will depend on different interference models, e.g., matrix based interference model or K-hop interference models, and needs to be quantified in those settings. It could also be interesting to study the scheduling performance when the energy cost of in-network aggregation itself is not negligible.

IV. SIMULATION RESULTS

In our simulation, 200 nodes are put into a $10\times10(m^2)$ space. Any nodes within communication radius *rc* can communicate with each other. We use the energy consumption as the performance metric. Assuming the packet with data amount *I* and transmission distance *d*, the transmission cost is $I(\beta d\gamma + E)$ when $d < rc$, where _ denotes energy consumption per bit on the transmitter and receiver circuit [3].The parameters are set that $\gamma = 2$, $\beta = 100(pJ/bit/m^2)$, _ = 40(*nJ/bit*). The numbers of hops are used to describe the latency.

We randomly create networks with minimum latency 4 and aggregation function *N*0(norm of *A*2 is 0, norm=0 for The word "data" is plural, not singular. The subscript for short, then issue a set of events with increasing latency constraints from 0 to 25.

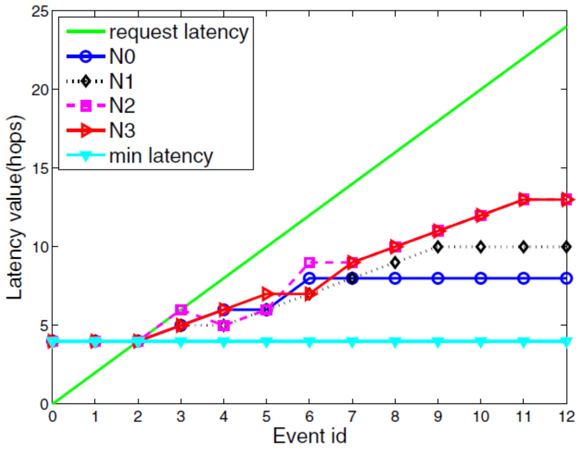

Fig 3. Delay bounded event aggregation

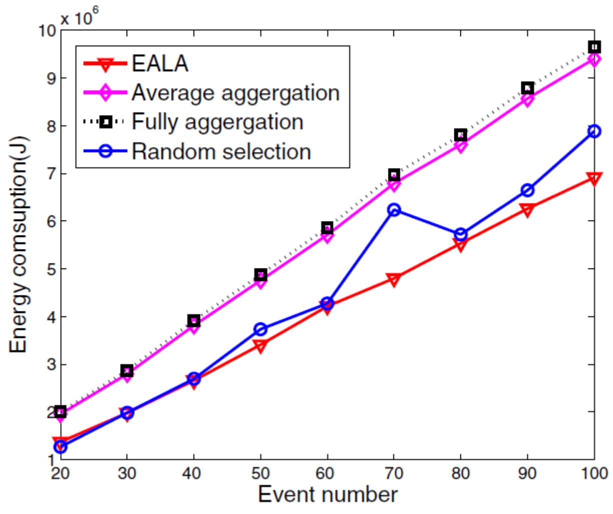

Fig. 4. Aggregation with different event number

We use DBEA to build aggregation tree and calculate the maximum latency. The experiment is repeated with different aggregation function *N*1(*norm* = 1), *N*2(*norm* = 2) and *N*3(*norm* = 3). Figure 3 shows the result and each point represents 50 times executions. All the results are between the minimal latency and latency constraint, which shows all latency constraints are strictly meet, at the same time the approach gets benefit compared with the approach simply selecting the minimal latency. Growing benefit goes with increasing aggregation function. The aggregation tree with *N*3 is longer than all others. For event 12, the latency of aggregation tree with *N*3 is 13 while that with *N*0 is only 8. The aggregation characteristic is recognized in the algorithm and generate a longer aggregation tree hence more opportunities for performance improvement.

Several experiments are undertaken to investigate the performance of our solution for EALA (EALA for short). For comparison, we choose several base algorithms including *fully aggregation*, *average aggregation* and *random selection*. In fully aggregation, the minimum latency constraint and fully aggregation function are used for all events. The average aggregation analyses all the aggregation functions and calculates an average one. Random selection uses multiple base events like EALA, but only selects them randomly. We take the experiments with different event number, latency number and base event number. The results are shown in Figure 3, 4. In Figure 4, we fix base event number as 20 and change the total event number from 20 to 100. As the result, EALA is always has the least energy consumption among these four approaches. One interesting thing here is average aggregation is almost the same with fully aggregation, which means the calculation of average value has limited help. This is because with the diversity of the events, only one average value cannot effectively decrease latency gap and aggregation gap. Random selection and EALA use more base events hence have much better performance. When event number is 20, it saves 50% energy than fully aggregation. As event number increases to 50, the energy saved decreases to 30 percentages but the quantity is increased to 1.5 × 106(*J*). Between EALA and random selection, EALA always has better performance. Similar analysis can be put to latency number (graph omitted here), and EALA also has better performance than others. The result about different base event numbers could serves as a guideline to decide proper base event number in the system. As in Figure 4, the energy consumption decreases sharply from base event number 0 to 20. In EALA, about 35% energy is saved. After 20, the plot decreases much more slowly. From 20 base events to 40 base events, less than 10% energy is saved. So in this system, 20 base events is a proper choice. In this figure 4, it is very clear that EALA outperform other approaches in terms of quantity and decreasing speed.

V. CONCLUSION

In this paper, we studied the event aggregation problem considering both aggregation function and latency constrains in multiple high-level events. We proposed solutions to this problem including a simple approach to describe aggregation function, the DBEA algorithm considering both latency constraint and aggregation function for single high-level event, and the optimal base events selection algorithm for aggregating multiple high-level events given a reliable constraint. The simulation results show that significant energy (up to 35% in our system) can be saved by using our algorithm.

REFERENCES

[1] Mr Bhabani Prasadv, Mrs. Chitrakala,"A Fast Convergecast Method for Tree-Based Wireless sensor Networks," Electronics Research Laboratory Memorandum Number M98/2, 2009..

[2] Claudio Barone, Fabrizio Ciarlo, Sebastiano Testa" Energy E_ciency and Tra_c Optimization through Data Aggregation in Wireless Sensor Networks," in Proc. EEE MEMS Workshop, Nagoya Jan. 2009, pp. 350355.

[3] K.-W. Fan, S. Liu,"Structure-free Data Aggregation in Sensor Networks," journal IEEE Transactions on Mobile Computing, vol. 6 Issue 8. August 2007

[4] Weiping Zhu, Jiannong Cao, Yi Xu," Event Aggregation with Di_erent Latency Constraints and Aggregation Functions in Wireless Sensor Networks" IEEE ICC 2011.

[5] H. Luo, Y. Liu and S. K. Das, Routing Correlated Data with Fusion Cost in Wireless Sensor Networks, IEEE Trans. on Mobile Computing, vol. 5, no.11, pp. 1620-1632, 2006.

[6] Changhee Joo, Jin-Ghoo Choi, and Ness B. Shro_, "Delay Performance of Scheduling with Data Aggregation in Wireless Sensor Networks" IEEE INFOCOM 2010.

[7] L. Becchetti, P. Korteweg, A. Marchetti-Spaccamela, M. Skutella, L. Stougie, and A. Vitaletti, Latency Constrained Aggregation in Sensor Networks, in the 14th conference on Annual European Symposium, 2006.

[8] Athanassios Boulis, "Castalia A simulator for Wireless Sensor Networks and Body Area Net- works"Ver 3.1,3.2 NICTA December 2010.

[9] Amitabha Ghosh, Ozlem Durmaz Incel, V.S. Anil Kumar, and Bhaskar Krishnamachari , "Algorithms for Fast Aggregated Convergecast in Sensor Networks" University of Southern California 2007.