BACCARAT: a Dynamic Real-Time Bandwidth Allocation Policy for IEEE 802.15.4

C. Nastasi, M. Marinoni, L. Santinelli, P. Pagano, G. Lipari, G. Franchino

Abstract—Recently, researchers and engineers began considering the use of WSN in time-sensitive applications. For effective real-time communications, it is important to solve the problem of contention to the communication medium providing an efficient bandwidth allocation mechanism.

In this paper we tackle with the problem of performing timely detection of events by a WSN. We propose a real-time bandwidth allocation mechanism for IEEE 802.15.4 that maximizes event detection efficiency and reduces statistical uncertainty under network overload conditions. On-line strategies complement off-line guarantees to enhance the confidence level of the measurements.

I. INTRODUCTION

Wireless Sensor Networks (WSNs) are emerging as enabling infrastructures for various types of large-scale distributed embedded applications. For a number of these applications timeliness is of great importance, e.g. industrial automation, process control [1], [2], [3] and low-cost distributed image processing [4]. Each node is expected to perform real-time computations and to send high quality data with guaranteed Quality-of-Service (QoS). These end applications rely on real-time distributed embedded systems, for which computations and communications must be both logically correct and produced on time.

In wireless networks, uncontrolled concurrent transmissions result in collisions, packet losses, and unbounded delays. Therefore, many MAC protocols for WSN provide disciplined access to the medium. The IEEE 802.15.4 standard [5] for WSN provides the Guaranteed Time Slots (GTS) mechanism where a Coordinator node can manage a TDMA-based access to the medium. The bandwidth allocation policy might be coded at upper layers.

The bandwidth allocation problem has been widely studied for high-end protocols (for example see [6]); concerning WSN, Chitnis et al. in [7] present a survey where various strategies for TDMA/CSMA-based MAC algorithms in a star network are detailed. The general idea is to run a schedulability test (off-line) or an admission test (on-line) to check if new incoming data flows can be accepted into the network. Whenever possible, a certain number of slots are allocated to the incoming flow.

As application-driven networks, WSNs may require high data reliability to maintain detection and response capabilities. Sensor nodes are typically affected by high failure rates so that a certain level of redundancy is required to the network to enforce reliability in the measurements. Moreover cloned observations (nodes) might improve the statistical confidence in the measurement as detailed by Pagano et al. in [4]. While sensor nodes redundancy exploits reliability [8], as a drawback it might increases the probability of network congestion: in an overloaded network important services cannot be guaranteed anymore unless proper scheduling strategies are applied to differentiate upon the flows on the basis of their content.

In this paper we consider the classical WSN application scenario depicted in Figure 1. A WSN is used for monitoring a set of observable variables \( \{x_1, \ldots, x_K\} \), which can reside in a range that goes from simple scalar data, like temperature, to images in a video scene. To cope with the unreliability of typical low-cost sensor nodes, each variable can be monitored by more than one node. In the figure, four identical nodes monitor variable \( x_1 \), three identical nodes monitor variable \( x_2 \), and so on.

A. Related work

The IEEE 802.15.4 protocol is the most popular standard for WSNs and specifies the Medium Access Control (MAC) sublayer and the Physical Layer of Low-Rate Wireless Personal Area Networks (LR-WPANs).

In Figure 2 the beacon-enabled operation of the WSN is represented. As it can be seen, IEEE 802.15.4 permits to differentiate best effort traffic (scheduled through the CSMA/CA mechanism) and elapsing a portion of the Superframe called Contention Access Period (CAP), from real-time traffic (allocated through the so called GTS) and organized in the Contention Free Period (CFP). Eventually the Superframe can include an IDLE period where no operation is allowed in the...
networking to enhance the lifetime of end-devices.

Bandwidth allocation protocols are proposed to cope with various traffic classes such as hard and soft real-time traffic. Relevant examples are iGAME [9], optimal GTS scheduling algorithm (GSA) [10] and Adaptive GTS Allocation (AGA) [11].

- In iGAME the coordinator assigns the requested GTS slots depending on the availability of the bandwidth and following a First Come First Served (FCFS) scheduling policy [12]. Flows are indistinguishable for what concerns data content, and the bandwidth allocation is based on data rate and packet length only. iGAME shares the same GTS among multiple data flows in order to improve the bandwidth utilization. Moreover, flows are eligible to remain unserved if activated after the saturation of the guaranteed bandwidth.

- The authors of GSA propose an Earliest Deadline First (EDF)-based [13] scheduling algorithm to minimize the total number of unallocated GTSs. GSA tries to smooth out the traffic by distributing the GTSs of a transaction over as many beacon intervals as possible while satisfying its timing constraint. On the other hand, the flows are indistinguishable, so it is not possible to select a subset of packets to be dropped in case of overload.

- AGA has been designed to solve the starvation problem providing fairness and low latency to flows. The coordinator computes the GTS schedule for the new beacon interval depending on the bandwidth requests and traffic priority sent by end devices in the previous beacon interval. AGA is not focused on real-time acting on ordinary network metrics like fairness and average latency.

B. Contributions of this paper

In this work we face the relevant redundancy problem in WSNs taking into consideration different facets. We propose the Bandwidth Allocation for Content based and Context Aware Real-time ApplicatIon (BACCARAT) protocol to be implemented on top of the IEEE 802.15.4 wireless communication stack. BACCARAT reserves a certain amount of bandwidth for off-line guarantees and allocates the remaining part to enhance the confidence level of the performed measurements. Our approach is suited to follow different strategies customized to specific event signatures and arrival rates. Moreover, our approach is explicitly designed to address the overload problem that might derive form the size scaling of the WSNs.

The remainder of the paper is structured as follows. In Section 2 we discuss our theoretical model. In Section 3 we present the system paradigms. In Section 4 we describe the details of BACCARAT, addressing the adopted scheduling policy and addressing the implementation issues on the IEEE 802.15.4 standard. In Section 5, we illustrate a specific case study to assess validity of the analytical model. In Section 6, we state our conclusions and propose future extensions of this work.

II. NETWORK AND TRAFFIC MODEL

In this paper, we consider star topologies, where \( N \) End Devices (EDs) are directly connected to the Coordinator via single hop routes. An ED can send data messages to the Coordinator upon event detection. A message consists of one or more MAC data frames, and is associated with a time constraint (deadline). We denote by flow a sequence of messages of the same type.

The network bandwidth is the resource shared among the nodes competing to send data, much like the CPU is shared among tasks. If we logically associate flows to tasks and messages to jobs, our bandwidth allocation strategy can be analytically studied making use of the real-time theory developed for scheduling systems. We adopt the real-time jargon and make use of the symbols defined in Table I.

![Image](image.png)

Fig. 2. Superframe Structure in the IEEE 802.15.4 standard.

| \( \Gamma = \{ \tau_i \} \) | the set of all the flows in the system | \( \bar{\tau}_{i,j} \) | message absolute deadline. |
| \( \mathcal{U}_i \) | message size, in bytes. | \( \bar{\tau}_{i,j} \) | message absolute deadline. |
| \( J_{i,j} \) | The \( j \)-th instance (message or job) of the \( i \)-th flow. | \( \mathcal{T}_i \) | period or minimum inter-arrival time between two messages of the \( i \)-th flow. |
| \( \mathcal{L}_i \) | the flow relative deadline. | \( \bar{\tau}_{i,j} \) | message activation time. |
| \( \bar{\tau}_{i,j} = \bar{\tau}_{i,j} + \mathcal{L}_i \) | \( J(t) = \{ J_{i,j} \} \) | \( \mathcal{J}(t) = \{ J_{i,j} \} \) | \( \{ J_{i,j} \} \) |

TABLE I

LIST OF SYMBOLS
We classify the information needed to reconstruct an event in the independent set \( \Upsilon = \{ \psi_1, \psi_2, \ldots, \psi_k \} \), with \( k \) \((k \leq N)\) being the number of observables. An observable measure is encoded in a message and transmitted by one or more flows. Therefore, we partition the flow set according to \( \Upsilon \), as \( \Psi = \{ \psi_1, \psi_2, \ldots, \psi_k \} \), where \( \psi_k = \{ \tau^k_i \} \) is the set of the flows that carry data for the observable \( k \). We call \( \psi \) the type of the flow. Note that, from the partition formal properties, \( \psi = \Gamma \) where \( \Gamma = \{ \tau_i \} \) is the set of all the flows in the system. Let \( w_k \) be the cardinality of subset \( \psi_k \), i.e. the number of flows that carry information on \( x_k \), with \( N = \sum_k w_k \). All flows \( \tau_i \in \psi_k \) are equivalent, in the sense that they carry semantically equivalent information (although the actual values of the measurements can be different). Thus, we assume that all flows belonging to the same subset \( \psi_k \) have the same parameters (\( \overline{T}^k, \overline{D}^k, \overline{C}^k \)).

Note that failure rate (generally high in WSN technology) is not the only reason to generate \( w_k \) independent readings of the same variable. The estimator of an experimental observable \( x_k \) is a random variable affected by statistical uncertainty \( \sigma_k \). Providing a set of \( w_k \) independent measurements for \( x_k \) permits to estimate it through the arithmetic mean \( \overline{x}_k \) with an error of \( \sigma_k / \sqrt{w_k} \). Of course, increasing the level of redundancy imposes to filter the transmission requests introducing an admission control based on the packet content as it is discussed in the next sections.

B. Network redundancy

The main problem in WSN, is that the total available bandwidth may not be enough to support transmission of all messages within a given period. However, we could dynamically reduce the redundancy level for some observables in a controlled manner. It is mandatory to guarantee that the Coordinator receives enough information to assess a certain confidence level on the measurement.

For every flow partition \( \psi_k \), we define \( w_k^o \) as the minimum number of flows of type \( k \) (\( k \)-flows) required by the Coordinator in order to guarantee the confidence level of the measurement. Clearly \( w_k^o \leq w_k \), where \( w_k \) is the maximum number of \( k \)-flows that can arrive at the Coordinator in a certain interval. Let \( \psi_k^o \subseteq \psi_k \) be any subset of \( w_k^o \) elements of \( \psi_k \). Since all flows belonging to \( \psi_k \) have the same parameters, all possible subsets of \( \psi_k \) with the same cardinality are equivalent, we can choose any one of them during the offline schedulability analysis.

We denote as \( \Gamma_G \), the guaranteed subset, the union of all \( \psi_k^o \). The off-line analysis, which will be described later on, checks that all flows in \( \Gamma_G \) can be safely admitted into the system. We denote as residual subset \( \Gamma_R = \Gamma \setminus \Gamma_G \).

III. System paradigms

In event-driven communications, the EDs can trigger a transaction asynchronously following the readings of their sensors. Since the bandwidth allocation is centrally managed by the network Coordinator it is reasonable to collapse the distributed system into a “virtual” centralized system where the available bandwidth is represented by a virtual processor. This permits to study bandwidth allocation as a uniprocessor scheduling problem.

A. Networking activities: association and message transmission

The communication protocol is composed by service data frames, called requests. We identify two types of requests: the flow request, used to declare the presence of a new flow in the network; and the job request, used to announce that a new job (a message) is ready on the ED to be transmitted. Both flow and job requests are used to transfer information from EDs to the Coordinator. In our model, we assume that the dimension of the requests are negligible compared with the CAP dimension, so that we may suppose that all of them reach the Coordinator with a null or small delay.

WSNs are systems that largely change structure during their lifetime. Then a node is allowed to associate to the network any time and to create new traffic, sending a flow request to the Coordinator. Thus the Coordinator checks upon the feasibility of scheduling new flows. As already stated, flows and messages are the abstract description of a kind of data frame that an ED can transmit. Messages, instead, are instances of flows; they are the actual data frames transmitted by the EDs.

In the start-up (association stage) the Coordinator is in charge of parsing flow requests and updating the \( \Gamma_G \) or \( \Gamma_R \) sets. At every association request the Coordinator carries out an admission test called Guaranteed Flow Admission (GFA) to check the schedulability of \( \Gamma_G \).

At the end of the association stage, the Coordinator listens for job requests and manages the functional communication by scheduling messages (transmission stage). The acceptance test, called Message Acceptance and Scheduling (MAS), is applied to schedule messages based on the requests \( req_{i,j} \) received at run-time. During this step, if a new flow type requires guarantees (i.e., a new node is asking to join the network), a further admission test must be run in background as the new flow request arises.

The association and transmission stages are respectively based on flow requests and job requests. They can be described in terms of sequence of steps as detailed by Algorithms 1 and 2.

Algorithm 1 Association Handshaking Algorithm

1: A new node declares its flow \( \tau_i \) to the Coordinator;
2: if \( \tau_i \) is new (a new type) and \( \Gamma_G \cup \{ \tau_i \} \) can be guaranteed then
3: Update \( \Gamma_G = \Gamma_G \cup \{ \tau_i \} \);
4: else
5: Update \( \Gamma_R = \Gamma_R \cup \{ \tau_i \} \);
6: end if

B. Scheduling policy

We model the Coordinator as a hierarchical scheduler [14], [15] with two servers managing the bandwidth [16]. The high priority server, \( S_G \), manages the guaranteed bandwidth \( U_G \)
Algorithm 2 Transmission Handshaking Algorithm

1: A job \( J_{i,j} \) is ready to be transmitted by an ED sending \( req_{i,j} \).
2: The Coordinator schedules the message \( J_{i,j} \) according to its policy.
3: The node transmits the \( J_{i,j} \) when scheduled by the Coordinator.

![Diagram of IEEE 802.15.4 Superframe](Image)

Fig. 3. Virtual-time representation of the IEEE 802.15.4 Superframe

associated to the \( \Gamma_G \) flow set. The low priority server \( S_R \), uses the residual bandwidth, \( U_R = 1 - U_G \); \( S_G \) dispatches the message requests \( req_{i,j} \) contained in an ordered queue \( J_G \) to schedule the \( J_{i,j} \) from EDs. \( S_R \), instead, dispatches from a different ordered queue \( J_R \) the residual message requests. Both queues \( J_G \) and \( J_R \) are ordered following the flow classification and the message request arrival as described in Section IV.

Furthermore, the two servers \( S_G \) and \( S_R \) need to interact one another in order to cope with the dynamic condition of the system. Indeed, it can happen that \( S_G \) has unused bandwidth that \( S_R \) could use to schedule its messages without jeopardizing the guaranteed schedulability of \( \Gamma_G \) and its entries in \( J_G \).

IV. BANDWIDTH MANAGEMENT FOR IEEE 802.15.4 NETWORKS

The standard provides a slotted mode that enables a TDMA based scheme to access the channel within the so called Contention Free Period (CFP). Every CFP is divided into maximum 7 GTSs, each one spawning one or more slots. The standard limits in time the maximum CFP width, setting a maximum 7 GTSs, each one spawning one or more slots. The request is received in the CAP by the Coordinator. If the job is immediately selected for transmission the Coordinator includes the appropriate GTS descriptor in the next beacon frame. Therefore, if a job is activated at time \( \pi_{i,j} \), the Coordinator can schedule it in the next CFP, let it be \( \tilde{\pi}_{i,j} \), as shown in Figure 3. However, the job absolute deadline must be set considering this activation offset. The worst-case relative deadline can be computed as follows:

\[
D_i = \left\lceil \frac{\tilde{\pi}_{i,j} - (2\times BI)}{TS} \right\rceil,
\]

where \( D_i \) is expressed in number of slots; note that \( 2\times BI \) is the largest value for \( (\tilde{\pi}_{i,j} - \pi_{i,j}) \); the function \( \text{empty}(x) \) calculates the interval from time \( \tilde{\pi}_{i,j} \) to time \( x \) that cannot be used due to the interferences of CAPs and IDLEs:

\[
\text{empty}(x) = \left\lceil \frac{x}{BI} \right\rceil \times (\text{CAP}_{\text{min}} + I) + I
\]

where \( \text{CAP}_{\text{min}} \) is the minimum duration of CAP and \( I = \text{IDLE} \) is the idle time. Notice that, the time needed to transmit a beacon is part of \( \text{CAP}_{\text{min}} \).

Finally, the flow period can be transformed with the simple formula below:

\[
T_i = \left\lceil \frac{\text{TS} - \text{empty}(T_i)}{TS} \right\rceil
\]

a) The flow-cap effect: In the IEEE 802.15.4 standard a big problem is posed by the maximum number of the GTS descriptors in the beacon, equal to 7. We can allocate up to 7 different flows in a CFP so that an application cannot get more than 7 different flows in a Superframe, although some residual bandwidth is available. We denote this effect as the flow-cap effect.

Theorem IV.1 Let \( C = \{C_1, C_2, \ldots, C_N\} \) be the set of sorted flows computation times, where \( C_i \leq C_{i+1} \forall i \in [1, N - 1] \). Let \( \text{CFP} \) be the number of TS in the CFP interval in each Superframe. Let \( S \) be a non-preemptive scheduling algorithm for the flows requests. A sufficient condition for avoiding the flow-cap effect is that:

\[
1 + \sum_{i=1}^{6} C_i \geq \text{CFP}.
\]

Proof: A Superframe can accommodate at most 7 different flows. \( \{C_1, \ldots, C_7\} \) is the subset of the 7 flows with the lowest computation time among all the possible subsets
of 7 flows of \( C = \{C_1, C_2, ..., C_N\} \). The worst case for a Superframe allocation (the minimum numbers of slots allocated) is obtained considering \( \{C_1, ..., C_7\} \), where \( C_7 \) has not been completed in the previous Superframe and requires 1 more slot from the actual Superframe. This way one GTS is used for 1 slot only. In the worst case the total computation time is \( 1 + \sum_{i=1}^{7} C_i \). If it is bigger than the available TSSs in a Superframe <CFP>, the flow-cap effect cannot happen. If the flow-cap effect does not appear with the application worst case then the application is totally unaffected, which demonstrates the theorem.

A. Guaranteed Flows Admission test

In this section we present a schedulability test to guarantee that the set of \( \Gamma_G \) flows can always receive enough bandwidth to meet the real-time requirements.

The schedulability test (GFA) acts as an admission control: every time a new flow \( \tau_i \) wants to join the system, a new instance of the test is executed. We apply the EDF scheduling policy to assign the bandwidth to the messages, considering the Utilization Criterion (UC) and the Processor Demand Criterion (PDC) [16]. However our test can be easily extended to different scheduling algorithms.

Every flow is denoted by a set of parameters \( \tau_i(C_i, D_i, T_i) \) and the partition \( \psi_k \) it belongs to. First of all, the flow parameters are translated into the virtual-time representation, obtaining \( C_i, D_i, T_i \), defined in Equations (1)-(3). If there are already \( w_k^G \) flows in the system for \( \psi_k \), then the new flow does not need to be guaranteed and will be added to \( \Gamma_R \). If the minimum number of guaranteed flows \( w^G_k \) has not been reached yet, Algorithm 3 (GFA), detailing step 2 in Algorithm 1, is executed and eventually \( \tau_i \) is added to \( \Gamma_G \).

Algorithm 3 GFA: admission test for new incoming flows.

```plaintext
1: if (UC algorithm) \(-\sum_{i=1}^{N} \frac{C_i}{D_i} > 1\) then
2: Reject request
3: Exit test
4: end if
5: if (UC algorithm) \(-\sum_{i=1}^{N} \frac{C_i}{D_i} \leq 1\) then
6: Accept request
7: \( \bar{U}_G = \sum_{i=1}^{N} \frac{C_i}{T_i} \) \( \mod \) TS
8: else
9: if (PDC with total utilization) - PDC(1) fails then
10: Reject request
11: Exit test
12: else
13: Accept request
14: \( \bar{U}_G = 1 \)
15: end if
16: end if
17: while PDC(\( \bar{U}_G \)) succeed do
18: \( \bar{U}_G = \bar{U}_G - \Delta U \)
19: end while
20: Set \( U_G \) to the last successful \( \bar{U}_G \).
```

Given a new flow \( \tau_i \) the GFA verifies if \( \Gamma_G = \tau_i \cup \Gamma_G^{old} \) is schedulable. The test starts with the necessary condition (UC):

\[
U_G = \sum_{i=1}^{N} \frac{C_i}{T_i} \leq 1
\]

If it fails, \( \tau_i \) is rejected (not enough bandwidth), the test returns and the system continues to work with the previous flow set \( \Gamma_G = \Gamma_G^{old} \); otherwise it tries the sufficient UC to the most general case with \( D_i \leq T_i \):

\[
\sum_{i=1}^{N} \frac{C_i}{D_i} \leq 1.
\]

If this test succeeds, the flow is accepted, otherwise the PDC is applied. Whereas the latter succeeds, \( \Gamma_G \) is schedulable.

We denote by resource supply function (sbf) \( \text{sbf}(t) = U_G \cdot t \), by demand bound function (dbf)

\[
dbf(t) = \sum_{\forall i: \tau_i \in \Gamma_G} \left( \left( \frac{t - D_i}{T_i} \right) + 1 \right) \cdot C_i,
\]

where

\[
D = \{d_k|d_k \text{is a deadline} \land d_k \leq \min\{H, L^*\}\},
\]

where \( H, L^* \) are calculated as:

\[
H = \text{lcm}(T_1, ..., T_n)
\]

\[
L^* = \frac{\sum_{i=1}^{n} (T_i - D_i) U_i}{U_G - U},
\]

using the \( n \) flows in \( \Gamma_G \), and \( U_G \) is the resource required to have a schedulable flow set \( \Gamma_G \). Yet \( U \) is the \( \Gamma_G \) bandwidth occupation, \( U = \sum_{i=1}^{n} \frac{C_i}{T_i} \). Thus, in [17], the PDC states that synchronous flows are schedulable if and only if

\[
\forall t \in D \quad dbf(t) \leq \text{sbf}(t).
\]

The goal of GFA is to prove the schedulability of \( \Gamma_G \) and to find the “best” sbf that leaves \( \Gamma_G \) schedulable. By best bound function \( \text{sbf}^*(t) \) we intend:

\[
\text{sbf}^*(t) = \min\{\text{sbf}(t)|\text{dbf}(t) \leq \text{sbf}(t)\},
\]

where \( t \) is the considered time interval and \( \text{sbf}^*(t) \leq t \). This is obtained using the PDC with decreasing utilization until the first failure. The iteration step is the equivalent utilization of a <TS>, computed as \( \Delta U = \frac{<CFP>}{<TS>} \), where <CFP> is the CFP length in <TS>. Equation (4) converges to the form of

\[
\text{sbf}^* = U_G \cdot t.
\]

Since the first server, \( S_G \), receives a capacity equivalent to \( U_G \), the next one, \( S_R \), gets the residual resource left by the first server:

\[
\text{sbf}_R(t) = t - \text{sbf}^*_G(t) \equiv U_R \cdot t = (1 - U_G) \cdot t,
\]

to schedule the messages in \( \Gamma_R \).

The complexity of the acceptance test is pseudo-polynomial. Such complexity is affordable since the algorithm is worked out off-line.
B. Message Acceptance and Scheduling with bandwidth reclaiming

So far we have defined off-line the capacity for the two servers. On-line the server status can change.

The MAS algorithm, executed on-line by the Coordinator, controls the input message requests and classifies them into guaranteed or residual. Figure 4 shows how the service request messages are classified according to their arrival time. Within a set of equivalent flows $\psi_k$, the message requests arriving first are selected to be the guaranteed ones, while the rest become the residual. The MAS algorithm can be divided into 2 sub algorithms; MASguaranteed that schedules guaranteed messages, and MASYielding that schedules the residual messages and reclaims unused resource from the high priority guaranteed server $S_G$. MASguaranteed, once classified the messages, extracts the messages request from the ordered queue $J_G$. The queue is ordered according to the relative deadline $D_i$ carried by the message request. Once the queue is emptied or the bandwidth $U_G$ is exhausted, MASguaranteed and triggers the execution of MASYielding.

Within MASYielding we define the guaranteed and residual capacities as $TS_G = [U_G + CFP]$ and $TS_R = CFP - TS_G$. If the server $S_G$ empties the waiting queue before exhausting the reserved TSs, then the remaining TSs in the $k$-th Superframe (denoted by $\Delta TS_G^k$) are given to $S_R (TS_R^k = TS_G^k + \Delta TS_G^k)$. If in the current Superframe all the guaranteed jobs have been served, the whole $S_G$ server bandwidth is reclaimed. The pending requests for $S_R$ are sorted using a priority based algorithm that takes into account the group relevance and the number of redundant messages already transmitted, $N_{x_j}$.

As discussed in Section II-A we want to reduce the uncertainty of $x_j$ taking the mean from a set of independent observations. In this case the priority assigned to the $j$-th job to be scheduled is given by $p_j = p_x/(N_{x_j} + 1)$, while jobs with the same priority are ordered by deadline. If all the observables have the same “intrinsic” priority $p_x$, a job that belong to the group $\psi_x$ with the minimum number of transmitted copies $N_{x_j}$ is chosen from the pending set.

These requests are served by $S_R$ using the Superframe budget $(TS_R^k)$ following the logic described in Algorithm 4.

Since the accepted requests will be executed in the next Superframe, no preemption can be made by successive requests, hence they are sequentially executed. Then the finishing time $f_i$ of job $i$ is computed starting from the finishing time of the last accepted job $f_{i-1}$. This should take into account the computation time $C_i$ and the interference induced by the server $S_G$.

The complexity of MAS is $O(n)$, where $n$ is the total number of messages to the Coordinator. Due to its low complexity, such algorithm can be applied on-line.

V. PERFORMANCE EVALUATION

Hereby we discuss a set of tests to prove the effectiveness of our approach in event reconstruction.

We model a Wireless Sensor Network composed by nodes observing the same events. The topology is the star one and the communication paradigm is conform with the beacon-enabled mode of the IEEE 802.15.4 standard.

In the proposed case study event reconstruction depends on the reading of four independent variables $x_1, x_2, x_3, x_4$; the output of this measurement process will be formally denoted by the linear function:

$$f(x_1, x_2, x_3, x_4),$$

\[ J_G = \{\ldots, J_{c,1}\}, \quad J_R = \{\ldots, J_{b,2}, J_{a,2}\} \]

\[ J_G = \{\ldots, J_{a,1}\}, \quad J_R = \{\ldots, J_{b,1}, J_{c,1}\} \]

\[ J_G = \{\ldots, J_{b,3}\}, \quad J_R = \{\ldots, J_{c,3}, J_{a,3}\} \]
$f$ being undefined if a minimal set of readings is not available at the Coordinator. Adopting our formalism we need at least $w^G_k$ reports ($k = 1, 2, 3, 4$) for each observable.

We assume that sensor readings are affected by random errors; a fortiori also $f$ is affected by statistical uncertainty. Therefore we define the variance of $f$ as:

$$\sigma^2 = \left(\frac{\partial f}{\partial x_1}\right)^2 \sigma_1^2 + \left(\frac{\partial f}{\partial x_2}\right)^2 \sigma_2^2 + \left(\frac{\partial f}{\partial x_3}\right)^2 \sigma_3^2 + \left(\frac{\partial f}{\partial x_4}\right)^2 \sigma_4^2,$$

where $\sigma_i^2$ is the variance in measuring $x_i$. A good estimator of $\sigma^2$ is $s^2$ calculated through the mean value of the random variables $x_i$ in the statistical sample:

$$s^2 = \frac{1}{n_i} \sum_{i=1}^{n_i} (x_i - \lambda)^2,$$

where $n_i$ is the number of received copies for observable $x_i$, $s_i$ the estimator of $\sigma_i^2$, $\lambda_i$ some c-numbers, assuming the linearity in $f$. The maximum accepted value for $s^2$ is that of events reconstructed by the minimal set of readings. An index of the measurement accuracy can be defined by:

$$\frac{s_M^2}{s^2} \in [1; +\infty[,$$

where $s_M^2 = \frac{1}{n_i} \sum_{i=1}^{n_i} (x_i - \lambda)^2$.

We now define event detection efficiency $\varepsilon$ as the ratio between the number of completely reconstructed events and the total number of events in the simulation.

To keep track of efficiency and accuracy at the same time, we combine the previous metrics into one single quality index defined as:

$$Q = \varepsilon \frac{s_M^2}{s^2} \in [0; +\infty[,$$

where $s^2 > 0$ is the average of $s^2$ for the reconstructed events.

We perform simulation studies comparing BACCARAT performances in terms of reconstruction efficiency and measurement quality with three popular scheduling algorithms: FCFS as the simplest and most used bandwidth allocation policy; Round Robin (RR) that enforces fairness among flows; EDF which is optimal for time constrained scheduling.

The simulation engine is a discrete event generator, written in C code so that the core of the BACCARAT scheduler can be easily ported to other simulators, e.g. NS-2 [18] and RTNS [19], or to existing network stacks for hardware platforms.

A. Simulation scenario

We model event arrivals (detected by the EDs equipped by appropriate sensors) with a Gaussian distribution centered at a mean value ($\mu$) of 1 second and having a standard deviation ($\sigma$) of 4 milliseconds. All the flow instances at the node level might be activated at the arrival of the event; it follows that the minimum inter-arrival time ($T_i$) is the same for all the flows and is approximated with $\mu - 3\sigma$.

In a realistic scenario the local overhead of preparing the reports depends on several factors, e.g. the sensor hardware details, the sampling period, the total software CPU time in the microcontroller and so on: hereby we model them with a random (uniformly distributed) delay in the flow instance activation time with respect to the true time of the event itself. Note that a generic delay distribution may generate an overlap among uncorrelated flows so that some jobs related to the event $j$ can be mixed up with some others related to the event $j + 1$ in a given detection period. We assume that all the jobs related to the event $j$ are scheduled (so that the GTSs are allocated) before the arrival of the event $j + 1$; in other words we constrain $\max\{a_{ij}\} < \min\{a_{i,j+1}\}$ $\forall i, j$.

Event detection misses due to local inefficiency or wireless transmission failures are not considered in this simulation since we focus on overload conditions taking place when all the nodes detect the event and consequently want to send their reports.

To model the elaboration time needed by the Coordinator to take a certain action upon the detected event, we set the relative deadline $D_i$ for each flow to 90% of the period $T_i$.

We assume for simplicity that the report size $\mathcal{C}_j$ for each observable is the same and equal to 300 bytes, corresponding to 3 MAC data frames. The Superframe is configured to have a fixed CFP length of 7 time slots.

B. Performance study

In the following test cases we discuss the behavior of $\varepsilon$ and $Q$ as functions of the system load. In the cases showed in Figure 5 and Figure 7, we study $\varepsilon(U_G)$ and $Q(U_G)$ for all the algorithms considered, varying the $U_{TOT}$ parameter in a set of independent simulation runs (each counting for 10000 events). In the other test cases, showed in Figure 6 and Figure 8, we consider the complementary situation, i.e. $\varepsilon(U_{TOT})$ and $Q(U_{TOT})$ varying the $U_G$ parameter.

The variation of $U_G$ and $U_{TOT}$ is obtained by uniformly increasing, respectively, either the minimum number of required reports per observable ($w^G_k$) or the total number of flows ($w_k$). Such uniform distribution of flows depicts the optimal scenario for EDF, FCFS and RR because of the balanced composition in the flow nature. We selected this as our simulation setup in order to compare BACCARAT with the other algorithms, where the latter are expected to perform at best. Furthermore, for the sake of simplicity, we suppose that all the observables have the same “weight” in the definition of $s^2$. Formally we can state:

$$\lambda_1^2=\lambda_2^2=\lambda_3^2=1, \quad s_1^2=s_2^2=s_3^2=s_4^2=1.$$

In the test case showed by the Figure 5-a, the load setting for $U_G$ is $w^G_k = 1, ..., 8$ and for $U_{TOT}$ is $w_k = 8, 11$, where $k = 1, 2, 3, 4$. In the Figure 5-b the setting is $w^G_k = 1, ..., 12$ and $w_k = 14, 19$ for $k = 1, 2, 3, 4$ resulting in overload conditions.

As expected, BACCARAT works as an ideal algorithm ($\varepsilon = 1$) if the off-line schedulability condition for the guaranteed subset is valid. Note that since $D_i < T_i$, the schedulability of the guaranteed set holds until $w^G_k = 9$ ($U_G = 0.94$). Moreover the performances of BACCARAT are almost independent from the $U_{TOT}$ parameter as shown in the Figures 5-a and 5-b.

Concerning the EDF algorithm it can be seen that if the system is overloaded ($U_{TOT} > 0.94$) it can still schedule the
required flows provided $U_G$ is small (flat part of EDF data sets with $\varepsilon = 1$ in the Figures 5-a and 5-b). Since EDF has no knowledge about the guaranteed subset and schedules the jobs on the basis of their deadlines, by increasing $U_G$ the probability of transmitting by chance all the required flows decreases and $\varepsilon$ drops.

The FCFS algorithm shows reduced performances than EDF and BACCARAT, because it does not apply any deadline-based scheduling policy. The RR algorithm performs even worse than FCFS. It equally distributes the available bandwidth among all the pending messages. This way, the more the system load increases the more the messages tend to be transmitted after their deadlines, thus degrading $\varepsilon$. This is the main cause of the low detection efficiency for RR. In Figure 5-b, where the system is overloaded, the RR results are omitted, being the efficiency always zero.

In the test case of Figure 6 we show $\varepsilon(U_{TOT})$ for different values of $U_G$. Only BACCARAT, EDF and FCFS results are shown. We set $w^G_k = 6, \ldots, 10$ with $w_k = w^G_k$, ..., 40 for $k = 1, 2, 3, 4$. It can be seen that BACCARAT shows full efficiency provided the off-line guaranteed condition is valid ($w^G_k < 9$, i.e. $U_G < 0.94$). Whenever that condition is no longer valid the efficiency drops: starting from $U_{TOT} = 2$, the Coordinator reconstructs less events; over that threshold of $U_G$, irrespective of $U_{TOT}$, the efficiency is zero.

As for the previous case, under values of $U_{TOT}$ smaller than 1, EDF is performing efficiently. At larger bandwidth demand EDF performs worse and worse. FCFS is always worse than EDF.

In the test case depicted in Figure 7 we deal with $Q(U_G)$. The experimental setting is the same of Figure 5-b, i.e. $w^G_k = 1, \ldots, 12$ and $w_k = 14, 19$ for $k = 1, 2, 3, 4$. As expected, for all algorithms $Q$ decreases as $U_G$ increases. If the guaranteed requirement is high, little room remains for transmission of the redundant copies of observables so that $s^2$ is close to $s_M^2$. Anyhow, the results show that the
quality obtained by BACCARAT is always higher than the other algorithms.

A more interesting case is the one of Figure 8, where the experimental setup is $w_k^G = 3, 5$ with $w_k = w_k^G, ..., 40$ for $k = 1, 2, 3, 4$. For all the data sets the required bandwidth for the guaranteed flows is low enough that BACCARAT, EDF and FCFS reconstruct the event with $\varepsilon = 1$ provided $U_{TOT}$ is low (portion of the plot with $U_{TOT} < 1$).

The quality index $Q$ starts from 1, being $U_{TOT} = U_g$ thus $s^2 = s^2_M$, and increases with $U_{TOT}$ until the system is overloaded. When $U_{TOT}$ is close to 1, the maximum value of $Q$ is obtained, i.e. all the bandwidth is allocated. The quality obtained by BACCARAT remains to the maximum level, while with EDF and FCFS it starts decreasing as $U_{TOT}$ increases (EDF always overcomes FCFS). This is important because BACCARAT permits to increase the redundancy level in the network, i.e. the number of flows and thus $U_{TOT}$, without reducing the event reconstruction efficiency and the measurement quality.

BACCARAT is proved to perform better than popular algorithms operated at their optimal conditions.

VI. CONCLUSIONS

Through this paper we addressed the issue of improving the bandwidth management in WSNs; in particular we proposed a working solution for the IEEE 802.15.4 standard. We showed the effectiveness of our solution in specific real-time distributed applications devoted to event detection and reconstruction.

Although simplified our approach fits the usual specifications of a real-world distributed system where the sensor nodes are required to interact with a given environment and to extract from it actual physical measurements with a defined confidence.

For the future, the flows formal model has to be improved removing some of the holding assumptions, e.g. allowing more flows per node. Furthermore, the reclaiming algorithm must be revisited to approach different scenarios, like multi-hop and complex network topologies.

Still some work must be done to test the bandwidth management protocols within a simulation environment realistically modeling networked communications like that offered by the NS-2 package and similia.

REFERENCES