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
initial.surname@sssup.it
Scuola Superiore Sant'Anna
Pisa, Italy

***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.

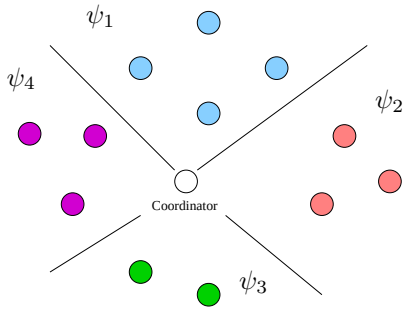


Fig. 1. Architecture of a WSN monitoring four observables.

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

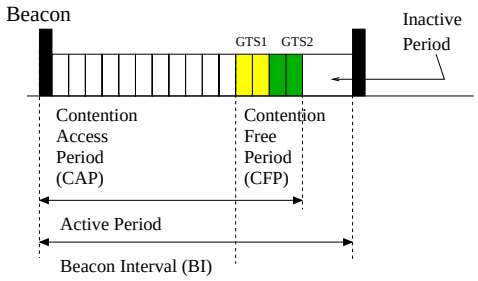


Fig. 2. Superframe Structure in the IEEE 802.15.4 standard.

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. Some information

τ_i	the i-th data flow
$\Gamma = \{\tau_i\}$	the set of all the flows in the system
$\overline{C}_i$	message size, in bytes.
$J_{i,j}$	The j-th instance (message or job) of the i-th flow.
$\overline{T}_i$	period or minimum inter-arrival time between two messages of the i-th flow.
$\overline{D}_i$	the flow relative deadline.
$\overline{a}_{i,j}$	message activation time.
$\overline{d}_{i,j} = \overline{a}_{i,j} + \overline{D}_i$	message absolute deadline.
$\mathrm{J}(t) = \{J_{i,j}\}$	The set of all the active messages in the network at the time t.

TABLE I
List of Symbols

must be collected by the Coordinator at each Superframe instance in order to accommodate a feasible schedule of the flows. A flow τ_i, periodic or sporadic, is described by the worst-case size (in bytes) $\overline{C}_i$ of its messages; the relative deadline $\overline{D}_i$, which represents the maximum transmission delay for a message; and the period (or the minimum inter-arrival time) $\overline{T}_i$ between two consecutive messages.

Using the real-time formalism, we call *job*, denoted by $J_{i,j}$, the j-th instance of a message of the i-th flow.

A. Flow semantics and classification

To reconstruct an event it is necessary to aggregate the readings of the nodes looking at the same event from comple-

mentary perspectives (we call these perspectives *observables*).

We classify the information needed to reconstruct an event in the independent set $\Upsilon = \{x_1, x_2, \ldots, x_K\}$, 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 Υ, as $\Psi = \{\psi_1, \psi_2, \ldots, \psi_K\}$, where $\psi_k = \{\tau_i^k\}$ is the set of the flows that carry data for the observable k. We call "k" 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 ψ_k, i.e. the number of flows that carry information on x_k, with $N = \sum_1^K w_j$. 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 ψ_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 σ_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 ψ_k, we define w_k^G 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 $w_k^G \leq w_k$, where w_k, is the maximum number of k-flows that can arrive at the Coordinator in a certain interval. Let $\psi_k^G \subseteq \psi_k$ be any subset of w_k^G elements of ψ_k. Since all flows belonging to ψ_k have the same parameters, all possible subsets of ψ_k with the same cardinality are equivalent, we can choose any one of them during the off-line schedulability analysis.

We denote as Γ_G, the *guaranteed subset*, the union of all ψ_k^G. The off-line analysis, which will be described later on, checks that all flows in Γ_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 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 Γ_G or Γ_R sets. At every association request the Coordinator carries out an admission test called Guaranteed Flow Admission (GFA) to check the schedulability of Γ_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 τ_i to the Coordinator;
2: **if** τ_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.

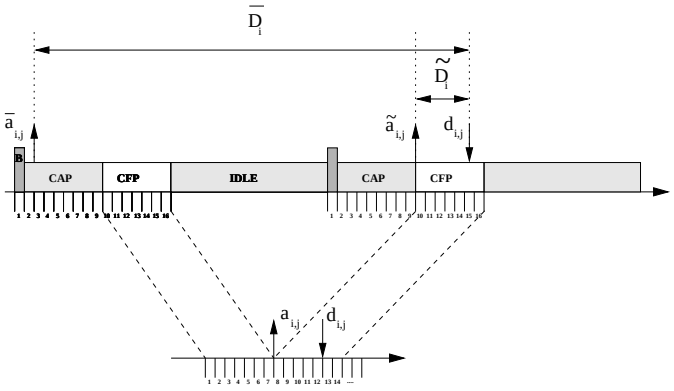


Fig. 3. Virtual-time representation of the IEEE 802.15.4 Superframe

associated to the Γ_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 Γ_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 minimum duration for the CAP: given an interval of time, only a certain portion is available for transmission using the GTS mechanism.

In this real-time analysis we discuss GTS allocation: to ease the mathematical formalism we apply an axis transformation mapping the time slots into a compact "virtual time" as represented in Figure 3. The virtual time is discretized in number of slots.

To define this mapping, we need to convert the parameters of a flow from one representation, ($\overline{T}_i$, $\overline{D}_i$, $\overline{C}_i$), to the other, (T_i, D_i, C_i). In the following, we denote by $<TS>$ the duration of one time slot; by B the transmission bandwidth (measured in bits per second); by $\mathsf{ifs}(\overline{C}_i)$ the *interframe spacing* (IFS); by $<BI>$ the beacon interval, as defined in the standard [5].

The worst-case message size of a flow ($\overline{C}_i$) can be mapped into the virtual time representation as:

$$C_i = \left\lceil \frac{\frac{\overline{C}_i \times 8}{B} + \mathsf{ifs}(\overline{C}_i)}{<TS>} \right\rceil <TS>, \tag{1}$$

where C_i is expressed in number of slots.

The notification of a new instance of the flow is carried by the service message $req_{i,j}$, as described in Algorithm 2. 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 $\overline{a}_{i,j}$, the Coordinator can schedule it in the next CFP, let it be $\tilde{a}_{i,j}$, 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:

$$\begin{aligned} \widetilde{D}_i &= \overline{D}_i - (2<BI>) \\ D_i &= \left\lfloor \frac{\widetilde{D}_i - \mathsf{empty}(\widetilde{D}_i)}{<TS>} \right\rfloor, \end{aligned} \tag{2}$$

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

$$\mathsf{empty}(x) = \left\lceil \frac{x}{<BI>} \right\rceil (<CAP_{min}> + <I>) + <I>$$

where $<CAP_{min}>$ is the minimum duration of CAP and $<I> \equiv <IDLE>$ is the idle time. Notice that, the time needed to transmit a beacon is part of $<CAP_{min}>$.

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

$$T_i = \left\lfloor \frac{\overline{T}_i - \mathsf{empty}(\overline{T}_i)}{<TS>} \right\rfloor \tag{3}$$

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 $C = \{C_1, C_2, ..., C_N\}$ be the set of sorted flows computation times, where $C_i \leq C_{i+1} \forall i \in [1, N-1]$. Let $<CFP>$ 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:*

$$1 + \sum_{i=1}^{6} C_i \geq <CFP>.$$

Proof: A Superframe can accommodate at most 7 different flows. $\{C_1, ..., C_7\}$ 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 $1 + \sum_{i=1}^{6} C_i$. If it is bigger than the available TSs 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 Γ_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 τ_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 $\tau_i(\overline{C}_i, \overline{D}_i, \overline{T}_i)$ and the partition ψ_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 ψ_k, then the new flow does not need to be guaranteed and will be added to Γ_R. If the minimum number of guaranteed flows w_k^G has not been reached yet, Algorithm 3 (GFA), detailing step 2 in Algorithm 1, is executed and eventually τ_i is added to Γ_G.

Algorithm 3 GFA: admission test for new incoming flows.

if *(UC algorithm)* - $\sum_{i=1}^{N} \frac{C_i}{T} > 1$ **then**
  Reject request
  Exit test
end if
if *(UC algorithm)* - $\sum_{i=1}^{N} \frac{C_i}{D_i} \leq 1$ **then**
  Accept request
  $\widetilde{U_G} = \left\lceil \sum_{i=1}^{N} \frac{C_i}{T} \right\rceil_{TS}$
else
  if *(PDC with total utilization)* - $PDC(1)$ fails **then**
    Reject request
    Exit test
  else
    Accept request
    $\widetilde{U_G} = 1$
  end if
end if
while $PDC(\widetilde{U_G})$ succeed **do**
  $\widetilde{U_G} = \widetilde{U_G} - \Delta U$
end while
Set U_G to the last successful $\widetilde{U_G}$.

Given a new flow τ_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, τ_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, Γ_G is schedulable.

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

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

where

$$D = \{d_k | d_k \text{is a deadline} \wedge d_k \leq min\{H, L^*\}\},$$

where H, L^* are calculated as:

$$H = lcm(T_1, ..., T_n)$$
$$L^* = \frac{\sum_{i=1}^{n} (T_i - D_i) U_i}{U_G - U},$$

using the n flows in Γ_G, and U_G is the resource required to have a schedulable flow set Γ_G. Yet U is the Γ_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

$$\forall t \in D \quad dbf(t) \leq sbf(t).$$

The goal of GFA is to prove the schedulability of Γ_G and to find the "best" sbf that leaves Γ_G schedulable. By *best* bound function $sbf^*(t)$ we intend:

$$sbf^*(t) = min\{sbf(t) | dbf(t) \leq sbf(t)\}, \tag{4}$$

where t is the considered time interval and $sbf^*(t) \leq t$. 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 $\Delta U = \frac{1}{<CFP>}$ where <CFP> is the CFP length in <TS>. Equation (4) converges to the form of

$$sbf^* = U_G \cdot t.$$

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:

$$sbf_R(t) = t - sbf_G^*(t) \equiv U_R \cdot t = (1 - U_G) \cdot t,$$

to schedule the messages in Γ_R.

The complexity of the acceptance test is pseudo-polynomial. Such complexity is affordable since the algorithm is worked out off-line.

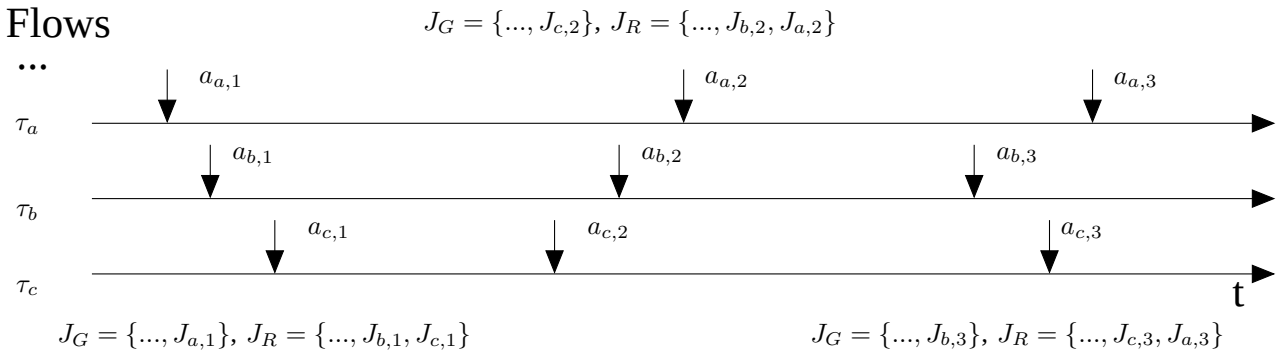


Fig. 4. Message assignment to S_G or S_R and their queues J_G, J_R. Three flows, τ_a, τ_b and τ_c, of the same ψ_k and their messages $J_{i,j}$ have been considered.

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 ψ_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 *MASresidual* 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 ends and triggers the execution of MASresidual.

Within MASresidual we define the guaranteed and residual capacities as $TS_G = \lceil U_G{<}CFP{>} \rceil$ 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 ΔTS_G^k) are given to S_R ($TS_R^k = TS_R + \Delta TS_G^k$). 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 $p_j = p_{x_j}/(N_{x_j} + 1)$, while jobs with the same priority are ordered by deadline. If all the observables have the same "intrinsic" priority ($p_{x_j} = p_0$), a job that belong to the group ψ_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.

Algorithm 4 MASresidual: acceptance test for S_R dispatcher.

```
if S_R queue is empty then
    Exit
end if
set the index i to the first queued request
while (TS_R^k > 0) & (S_R queue not empty) do
    Compute the request finishing time f_i
    if f_i > d_i then
        Skip the request
    else
        Accept the request
        Decrease the S_R server budget (TS_R^k) by C_i
    end if
    if S_R queue is NOT empty then
        increase the index i
    end if
end while
```

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. Denoting by f_0 the finishing time of the last guaranteed job, the following relation holds: $f_i = f_{i-1} + C_i + \left\lceil \frac{C_i - TS_r^k}{TS_G} \right\rceil TS_G$.

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:

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

f being undefined if a minimal set of readings is not available at the Coordinator. Adopting our formalism we need at least w_k^G 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 σ_i^2 is the variance in measuring x_i. A good estimator of σ^2 is s^2 calculated through the mean value of the random variables x_i in the statistical sample:

$$s^2 = \lambda_1^2 \frac{s_1^2}{n_1} + \lambda_2^2 \frac{s_2^2}{n_2} + \lambda_3^2 \frac{s_3^2}{n_3} + \lambda_4^2 \frac{s_4^2}{n_4}$$

where n_i is the number of received copies for observable x_i, s_i the estimator of σ_i^2, λ_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 $s_M^2 = \lambda_1^2 \frac{s_1^2}{w_1^G} + \lambda_2^2 \frac{s_2^2}{w_2^G} + \lambda_3^2 \frac{s_3^2}{w_3^G} + \lambda_4^2 \frac{s_4^2}{w_4^G}$.

We now define event reconstruction efficiency $\mathcal{E}$ 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:

$$Q = \mathcal{E} \frac{s_M^2}{< s^2 >} \in [0; +\infty[$$

where $< s^2 >$ 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 (μ) of 1 second and having a standard deviation (s) 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 ($\overline{T}_i$) is the same for all the flows and is approximated with $\mu - 3s$.

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 $max\{d_{i,j}\} < min\{a_{i,j+1}\}\ \forall i, j$.

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 $\overline{D}_i$ for each flow to 90% of the period $\overline{T}_i$.

We assume for simplicity that the report size $\overline{C}^k$ 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 $\mathcal{E}$ and Q as functions of the system load. In the cases showed in Figure 5 and Figure 7, we study $\mathcal{E}(U_G)$ 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. $\mathcal{E}(U_{TOT})$ and $Q(U_{TOT})$ varying the U_G parameter.

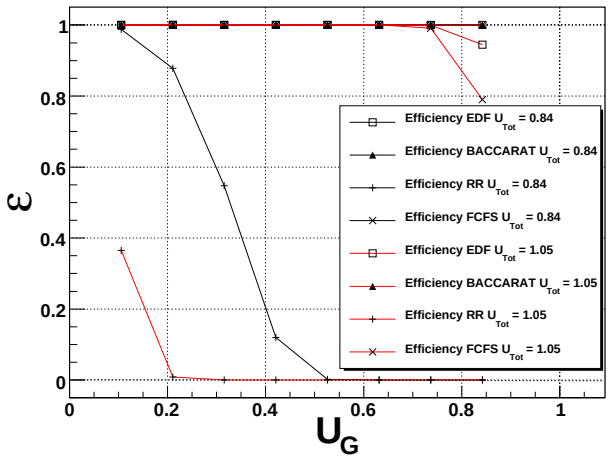
The variation of U_G and U_{TOT} is obtained by uniformly increasing, respectively, either the minimum number of required reports per observable (w_k^G) or the total number of flows (w_k). Such uniform distribution of flows depicts the optimal scenario for EDF, FCFS and RR because of the balanced composition 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:

$$\lambda_1^2 = \lambda_2^2 = \lambda_3^2 = \lambda_4^2 = 1 , \quad s_1^2 = s_2^2 = s_3^2 = s_4^2 = 1 .$$

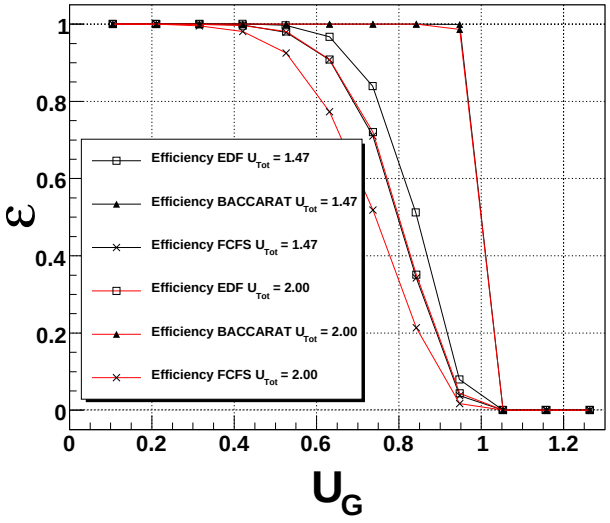
In the test case shown by the Figure 5-a, the load setting for U_G is $w_k^G = 1, ..., 8$ and for U_{TOT} is $w_k = 8, 11$, where $k = 1, 2, 3, 4$. In the Figure 5-b the setting is $w_k^G = 1, ..., 12$ and $w_k = 14, 19$ for $k = 1, 2, 3, 4$ resulting in overload conditions.

As expected, BACCARAT works as an ideal algorithm ($\mathcal{E} = 1$) if the off-line schedulability condition for the guaranteed subset is valid. Note that since $\overline{D}_i < \overline{T}_i$ the schedulability of the guaranteed set holds until $w_k^G = 9$ ($U_G = 0.94$). 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



(a) $U_{TOT} \leq 1.05$



(b) $U_{TOT} \geq 1.47$

Fig. 5. Event detection efficiency as function of U_G.

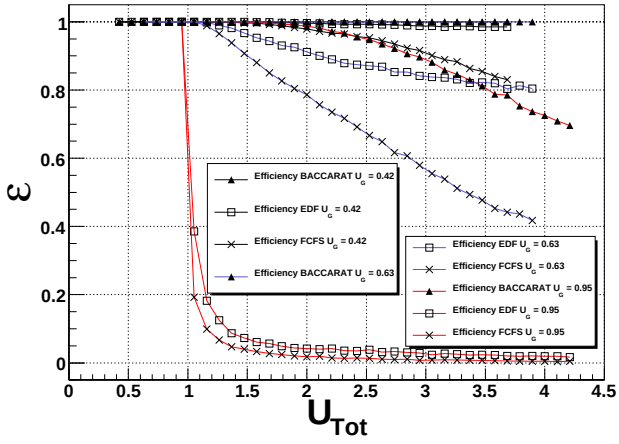


Fig. 6. Event detection efficiency as function of U_{TOT}

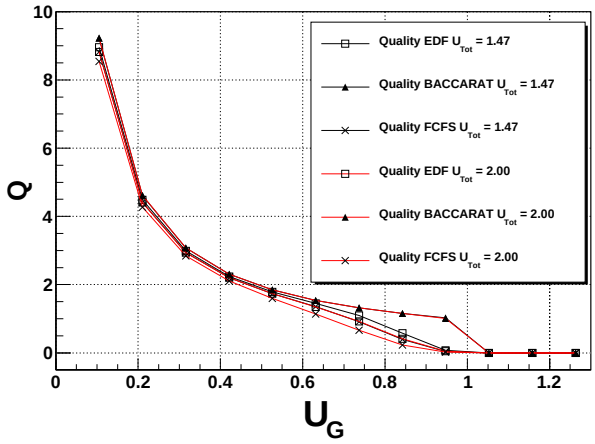


Fig. 7. Measurement quality as function of U_G

required flows provided U_G is small (flat part of EDF data sets with $\mathcal{E} = 1$ 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 $\mathcal{E}$ 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 $\mathcal{E}$. 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 $\mathcal{E}(U_{TOT})$ for different values of U_G. Only BACCARAT, EDF and FCFS results are shown. We set $w_k^G = 6, ..., 10$ with $w_k = w_k^G, ..., 40$ for $k = 1, 2, 3, 4$. It can be seen that BACCARAT shows full efficiency provided the off-line guaranteed condition is valid ($w_k^G < 9$, 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. $w_k^G = 1, ..., 12$ 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

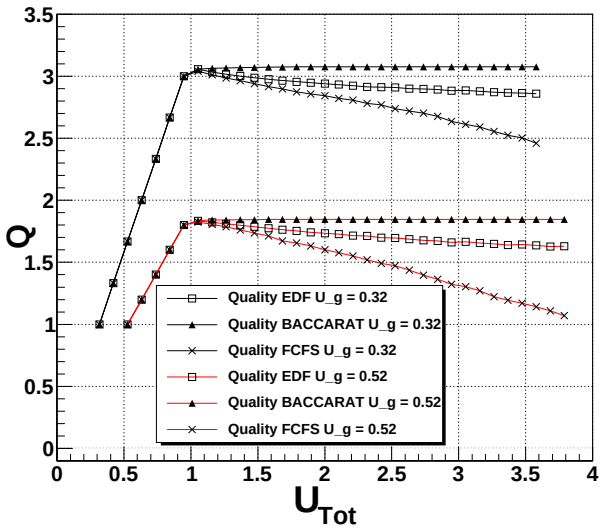


Fig. 8. Measurement quality as function of U_{TOT}

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 $\mathcal{E} = 1$ 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 $s^2 = s_M^2$, 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

[1] A. Bonivento, C. Fischione, A. Sangiovanni-Vincentelli, F. Graziosi, and F. Santucci, "Seran: a semi random protocol solution for clustered wireless sensor networks," in *IEEE International Conference on Mobile Adhoc and Sensor Systems Conference*, November 2005.

[2] N. Aakvaag, M. Mathiesen, and G. Thonet, "Timing and power issues in wireless sensor networks, an industrial test case," in *Proceedings of the 2005 International Conference on Parallel Processing Workshops (ICPPW)*, IEEE, 2005.

[3] N. Ota and P. Wright, "Trends in wireless sensor networks for manufacturing," *International Journal of Manufacturing Research*, vol. 1, no. 1, pp. 3–17, 2006.

[4] P. Pagano, F. Piga, and Y. Liang, "Real-time multi-view vision systems using wsns," in *SAC*, pp. 2191–2196, 2009.

[5] LAN-MAN Standards Committee of the IEEE Computer Society, *Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs)*. IEEE Press, 2003.

[6] S. H. Shah, K. Chen, and K. Nahrstedt, "Dynamic bandwidth management in single-hop ad hoc wireless networks," *MONET*, vol. 10, no. 1-2, pp. 199–217, 2005.

[7] M. Chitnis, P. Pagano, G. Lipari, and Y. Liang, "A survey on Bandwidth resource Allocation and Scheduling in wireless sensor networks," in *Proceedings of NBIS 2009*, Oct. 2009.

[8] H. Wen, C. Lin, F. Ren, Y. Yue, and X. Huang, "Retransmission or redundancy: Transmission reliability in wireless sensor networks," in *Mobile Adhoc and Sensor Systems, 2007. MASS 2007. IEEE Internatonal Conference on*, pp. 1–7, Oct. 2007.

[9] A. Koubâa, M. Alves, E. Tovar, and A. Cunha, "An implicit gts allocation mechanism in ieee 802.15.4 for time-sensitive wireless sensor networks: theory and practice," *Real-Time Syst.*, vol. 39, no. 1-3, pp. 169–204, 2008.

[10] C. Na, Y. Yang, and A. Mishra, "An optimal gts scheduling algorithm for time-sensitive transactions in ieee 802.15.4 networks," *Comput. Netw.*, vol. 52, no. 13, pp. 2543–2557, 2008.

[11] Y.-K. Huang, A.-C. Pang, and H.-N. Hung, "An adaptive gts allocation scheme for ieee 802.15.4," *IEEE Transactions on Parallel and Distributed Systems*, vol. 19, no. 5, pp. 641–651, 2008.

[12] J. Blieberger and U. Schmid, "Fcfs-scheduling in a hard real-time environment under rush-hour conditions," *BIT*, vol. 32, no. 3, pp. 370–383, 1992.

[13] C. Liu and J. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *Journal of the Association for Computing Machinery*, vol. 20, no. 1, 1973.

[14] J. L. Lorente, G. Lipari, and E. Bini, "A hierarchical scheduling model for component-based real-time systems," in *Proc. of IPDPS'06*, 2006.

[15] A. K. Mok and A. K. Feng, "A model of hierarchical real-time virtual resources," in *RTSS'02, IEEE Computer Society*, pp. 26–35, 2002.

[16] G. Buttazzo, *HARD REAL-TIME COMPUTING SYSTEMS: Predictable Scheduling Algorithms and Applications*. Kluwer Academics Publishers, 1997.

[17] S. K. Baruah, "Dynamic and static-priority scheduling of recurring real-time tasks," in *Real-Time System*, pp. 93–128, 2003.

[18] "Information Sciences Institute (University of Southern California, Los Angeles CA, USA), The Network Simulator NS-2.." http://www.isi.edu/nsnam/ns/.

[19] "The RTNS simulation suite.." http://rtns.sssup.it.