Visual tracking using Sensor Networks

Paolo Pagano, Francesco Piga,
Giuseppe Lipari
ReTiS laboratory
Scuola Superiore Sant'Anna
Pisa (I)
p.pagano@sssup.it,
francesco.piga@gmail.com,
lipari@sssup.it

Yao Liang
Department of Computer Science
Indiana University Purdue University Indianapolis
Indianapolis (IN)
yliang@cs.iupui.edu

ABSTRACT

New[1] trends in Wireless Sensor Networks envisage deployments for distributed applications requiring real-time support at the kernel level and Quality of Service at the network level.

In this domain, at the design stage, particular attention must be devoted to individual data packets as those entities carrying unique (not redundant) information. The performances of the deployed system (hereby felt as a black box) must be tracked against the reliability and timeliness offered in message delivery.

A Visual Tracking case study is discussed throughout this paper with the support of a simulation package modelling real-time scheduling policies at the device node kernels and bandwidth allocation techniques for network reliable communications as standardized in the IEEE 802.15.4 suite of protocols.

A set of results is carried out estimating the performances of the Visual Tracking system in two contexts (those of a monitored junction in an airport taxiway and in a parking area) very different for criticality and average volume of network traffic.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

Keywords

WSN, Operating Systems, Simulation, Data Analysis

1. INTRODUCTION

The philosophy underlying the standardization of the IEEE 802.15.4[9] protocol for low-rate Wireless Personal Area Networks (WPAN) is to support the large majority of present, planned, and envisioned networked applications deployed through cost-effective and autonomous wireless nodes.

The standard appropriately encompasses features coming from scheduled-based (through the so-called Guaranteed Time Slots, GTS) and contention-based (through the CSMA/CA mechanism) medium access paradigms thus supporting real-time and best-effort traffic types. Moreover particular attention is devoted to energy policy issues (for battery life extension) allowing the nodes in a network to synchronously sleep and wake up according to a broadcasted schedule.

In the Wireless Sensor Networks research domain, the device nodes are thought to have self-configuring, adaptiveness, and reactiveness capabilities to reduce the need of any kind of human intervention in the network procedures.

In recent years, many application scenarios with diversified constraints investing the communication policies have been proposed (and sometimes deployed) stressing some of the peculiarities of WSNs.

The most challenging proposals refer (but are not limited) to telemedicine, health care [11], industrial assembly [19], traffic control [4]. At the sensor level, peripherals providing simple scalar information (like temperature and illumination) are now complemented by more complex devices providing vector-type readings. Examples are given by a set of gyroscopes [25] defining the position with respect to a uniform magnetic field, and a set of accelerometers defining the acceleration vector in a reference frame. At the same way, CMOS and CCD manufactured cameras can be connected to Sensor Boards, to provide 1-D[27] or 2-D[20] photo frames; these devices (geometrically displaced in a volume) can build up a Multi-View Vision system to exchange data and reconstruct a scene[17].

Due to the limitation in node equipments (notably speed and memory) and network resources (like bandwidth), these applications are only possible with a coherent design at node and network levels. Simulation studies complement off-line analytical models and offer a good insight on the impact induced by low level mechanisms (e.g. intra-node settings like message queue size and scheduling policies) and high level protocols (e.g. in-network processing of sensor readings) to the performances of a system thought as a black box.

Especially when real-time and best-effort transmissions are scheduled together (by means of parallel active data flows) and the node kernels run multiple tasks sharing some local resources (notably CPU and memory), the capabilities carried out by simulation packages must appropriately model the hardware and software architecture of real-world set-up's. Throughout this paper we will show that Real Time Network Simulator (RTNS)[15, 16, 21], a real-time extension of the NS-2 simulator, is a novel, unique and effective response to this need in the domain of Wireless Sensor Networks.

[1]This work has been partially funded by the Italian MIUR ART-DECO project (FIRB/ RBNE05C3AH).

SIMUTOOLS'09 March 2-6, 2009, Rome, Italy

1.1 Related work

In discrete event simulation, the operation of a system is represented as a chronological sequence of events. Each event occurs at an instant in time and marks a change of state in the system.

Discrete event simulators are usually implemented making use of a re-sizeable event queue where to post and pop events for appropriate processing. For instance, time-triggered activities regularly post expiration events into the queue to produce a periodic sequence of actions. The queue is re-ordered at every post to always keep the closest event in front; the physical notion of time is discretized and incrementally elapses by the interval between the two latest expiration events at every pop.

Popular simulators like OPNET[14], NS-2[12], and TrueTime[3] support most of the features of the IEEE 802.15.4 standard for WPANs especially those related to the MAC layer mechanisms for network formation and management and contention-based transmissions (via the CSMA/CA scheduling algorithm). The support for GTS is absent in TrueTime whereas is provided by external contributions to OPNET [8] and NS-2 (this work).

The authors of [8] motivates the selection of OPNET criticizing the (not native) support of NS-2 for wireless communications. Actually numerical comparisons of the two packages in consistent conditions are rare in literature. Those who tried to perform this comparative analysis sometimes ruled out both of them [10]. Recently [7], many arguments supporting the wireless model of NS-2 have been proposed justifying the unreliable results (sometimes obtained in simulation runs) as driven by a bad setting in some parameters of the wireless modules. The authors show that tuning these parameters permits a strict adherence between real world and simulated data.

Furthermore the remarkable strict adherence to the IEEE 802.15.4 standard of the NS-2 WPAN module [26] and the long debugging stage (from the release 2.26 to 2.31) which has patched the module to fix imprecise behaviors (see for example Chapter "Changes made to the IEEE 802.15.4 Implementation in NS-2.31" in the reference manual [13]) legitimate the use of NS-2 as simulator in the WSN context.

The imitation of kernel mechanisms are natively included in TrueTime (by means of a Kernel block), added by COTS to NS-2 (using the RTNS extension) and totally absent in OPNET. In Table 1 a naïve comparison of the simulators shows that RTNS represents the only software solution, to the best of authors knowledge, for modelling distributed WSN applications with real-time constraints acting both at node and network levels.

	Kernel imitation	802.15.4 CSMA/CA	802.15.4 GTS
TrueTime	N	N	A
NS-2 (RTNS)	EC	N	EC
Opnet	A	N	EC

Table 1: Comparison among simulation packages. N = Native, A = Absent, EC = External Contribution.

1.2 Contribution of this work

At the design stage of a networked application, it is beneficial to model each aspect of the chain connecting the world of the phenomena with the world of the recorded data. To achieve this goal, in this paper we make use of a sophisticated simulation framework capable to model some phenomena and to retrieve the response obtained by a WSN against a set of software design choices and network conditions.

In RTNS, intra-node policies for resource management and task scheduling (usually handled by a lightweight kernel) are modeled together with network-related phenomena by means of a cosimulator engine joining the NS-2 and RTSim[18, 22] packages.

The case studies hereby debated refer to an information system performing target tracking by composing the camera views of wireless nodes installed along the possible trajectories of the moving vehicle.

Some limitations in NS-2 have been overcome to model event-driven transmissions and to support the MAC layer bandwidth allocation procedures introduced in the IEEE 802.15.4 standard and not included in the native WPAN module of the NS-2 package. Moreover a naive image processing module has been included in the node S/W architecture to take into account the discrete nature of the CPU response to stimuli coming from the external world.

The remaining sections are organized as follows: in Section 2 we briefly introduce the IEEE 802.15.4 standard and its instantiation in the WPAN module of the (enhanced) NS-2 simulator; moreover we describe the visual tracking model implemented in RTNS; in Section 3 we introduce the tracking scenario based on distributed vision and the performance metrics we focus on; in Section 4 we discuss the results obtained in the two case studies; in Section 5 we will comment on the results and propose a possible continuation for this research.

2. MODELING WSN IN RTNS

2.1 The IEEE 802.15.4 Standard

The IEEE 802.15.4 protocol specifies the Medium Access Control (MAC) sub-layer and the Physical Layer of Low-Rate Wireless Personal Area Networks (LR-WPANs). The SSCS (Service Specific Convergence Sublayer) abstracts the Access Point of some services offered by the MAC to upper layers.

The IEEE 802.15.4 Physical Layer uses a 16-ary encoding alphabet (4 bits/symbol) in Direct Sequence Spread Spectrum (DSSS) modulation over three operational frequency bands: 2.4 GHz (16 channels); 915 MHz (10 channels); 868 MHz (1 channel). The IEEE 802.15.4 MAC layer supports two operational modes: (1) the non beacon-enabled mode, in which the MAC is simply ruled by non-slotted CSMA/CA; (2) the beacon-enabled mode, ruled by slotted CSMA/CA, in which beacons are periodically sent by a special node (called network coordinator) to synchronize (in time) nodes that are associated with it, and to carry additional information about the transmission structure. In beacon-enabled mode, the Coordinator defines a SuperFrame structure (Fig. 1) which is constructed based on: (1) the Beacon Interval (BI), which defines the time between two consecutive beacon frames; (2) the SuperFrame Duration (SD), which defines the active portion in the BI, and is divided into 16 equally-sized time slots, during which frame transmissions are allowed. Optionally, an inactive period is defined if BI $>$ SD. During the inactive period (if it exists), all nodes may enter in a sleep mode (to save energy).

BI and SD are determined by two parameters - the Beacon Order (BO) and the SuperFrame Order (SO):

$$BI = aBaseSuperFrameDuration \cdot 2^{BO}$$
$$SD = aBaseSuperFrameDuration \cdot 2^{SO}$$

assuming $0 \leq SO \leq BO \leq 14$, $aBaseSuperFrameDuration$ = 15.36 ms (operating at 250 kbps in the 2.4 GHz band), corresponding to the minimum SuperFrame duration at SO = 0. During the SuperFrame Duration, nodes compete for medium access using slotted CSMA/CA, in the Contention Access Period (CAP). IEEE

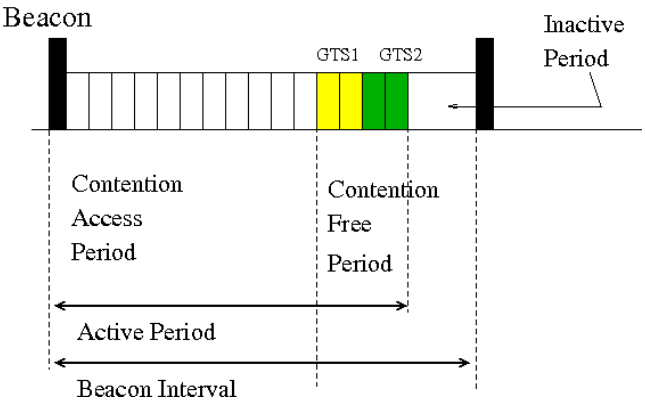

Figure 1: SuperFrame Structure in the IEEE 802.15.4 standard. In beacon enabled mode, the beacon interval can be subdivided into a Contention Access Period (CAP) and eventually into a Contention Free Period (CFP) and Inactive Period.

802.15.4 also supports a Contention-Free Period (CFP) within the SD, by the allocation of Guaranteed Time Slots (GTS). It can be easily observed in Fig. 1 that low duty-cycles can be configured by setting small SO values as compared to BO, resulting in longer sleep (inactive) periods. The standard supports three network topologies: Star, Mesh and Cluster-Tree, illustrated in Fig. 2.

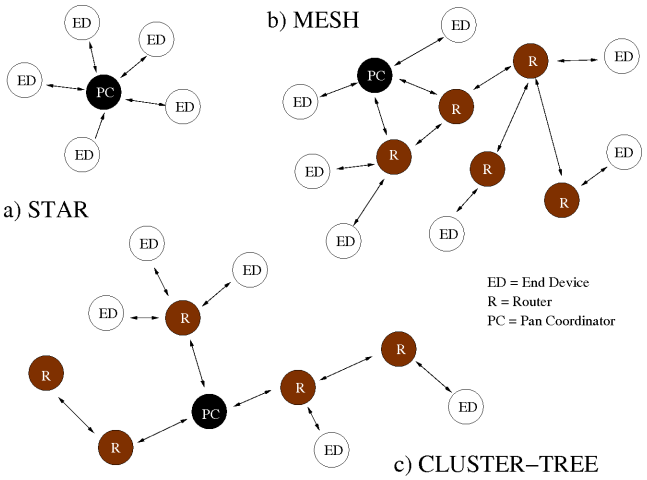

Figure 2: WPAN star, mesh, and cluster-tree network topologies.

In the Star topology (Fig. 2.a) communications must always be relayed through the coordinator; Star networks can operate in both beacon-enabled and non beacon-enabled modes. In the Mesh topology (Fig. 2.b), each node can directly communicate with any other node within its radio range or through multi-hop; Mesh networks must operate in the non beacon-enabled mode. The Cluster-Tree topology (Fig. 2.c) is a special case of a Mesh network where there is a single routing path between any pair of nodes and a distributed synchronization mechanism (operates in beacon-enabled mode).

2.2 The WPAN module in NS-2

In Figure 3 the network stack standardized in the IEEE 802.15.4 suite is shown. In the superimposed call-outs, the corresponding services implemented in the native WPAN module in NS-2 (release 2.33) and the modifications introduced by this research project are

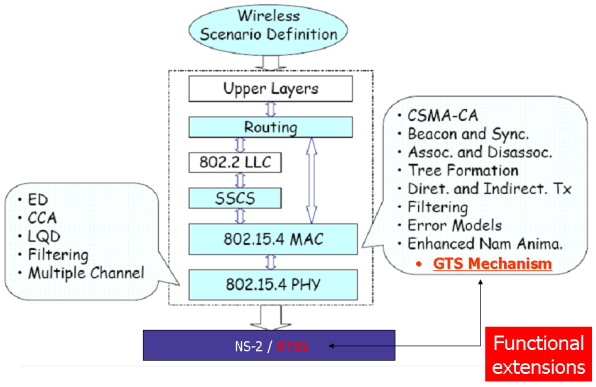

Figure 3: The networking stack as standardized in the IEEE 802.15.4 protocol for WPAN. In the call-outs the services implemented in the NS-2 WPAN module and the functional extensions added within this work.

specified: we namely refer to the GTS mechanism and the link with the RTNS framework.

The scarce documentation about the WPAN package in NS-2 refers to the mechanisms exported to the final user interface in some TCL scripting examples[5]: these mechanisms refer for instance to network start-up, node association, network topology and beacon order selection, etc.

Following a strict adherence to the standard, the MLME-GTS.request, MLME-GTS.confirm, and MLME-GTS.indication MAC primitives have been implemented for the GTS allocation as shown in Figure 4.

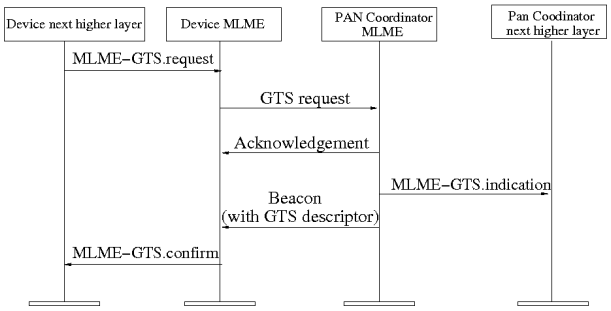

Figure 4: Sequence diagram for a GTS allocation request by a device node.

When an (associated) node wants to transmit (or receive) data in real-time, it makes use of the GTSs. A MLME-GTS.request is generated at Network layer; afterwards the node sends a Command frame to the coordinator. The device waits for the coordinator acknowledges receipt and then parses the list of GTS descriptors in the forthcoming beacon to identify the starting slot assigned to it. After the transmission of the ACK frame to the device, the MAC layer of the coordinator calls the MLME-GTS.indication notification procedure to its agent. At the reception of the beacon, the MAC layer of the device notifies the success to its agent by calling the MLME-GTS.confirm procedure.

A GTS can be deallocated whenever the device node formulates an explicit request as shown in the collaboration diagram of Figure 6/a with a sequence of function calls very similar to the previous

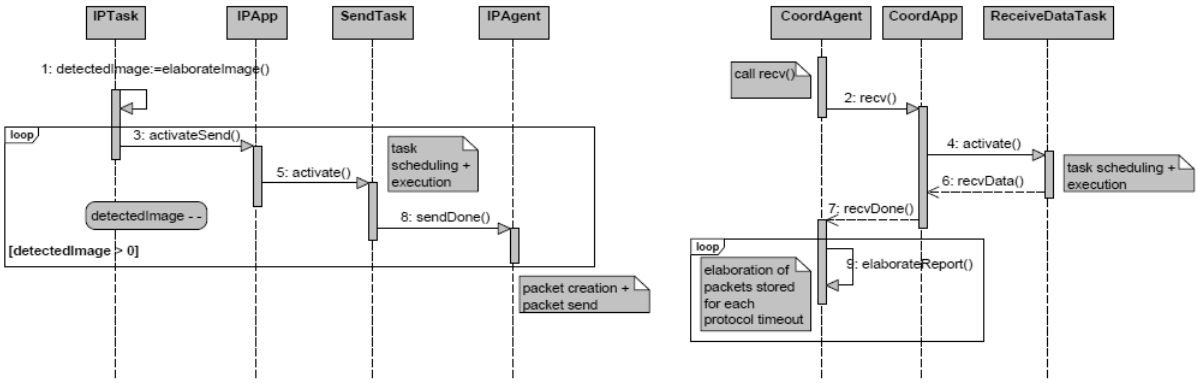

Figure 5: The UML sequence diagrams for sending (left) and receiving (right) reports.

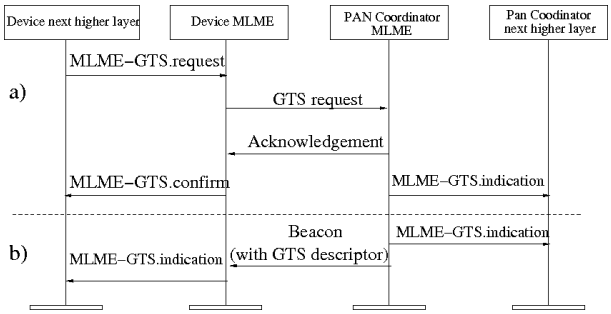

Figure 6: Sequence diagram for a GTS deallocation request by a device node (a). Explicit removal by the coordinator (b).

case.

Moreover the coordinator can deallocate a GTS (see Figure 6/b) inserting that GTS descriptor into the list of "removed" GTS in the forthcoming beacon packets whenever one of these conditions happen:

- the upper layers of the coordinator require the deallocation;
- the device did not make use of the GTS (in reception/ transmission) for 2^n SuperFrames where $n = 2^{8-BO}$ if $0 \leq BO \leq 8$ and $n = 1$ if $9 \leq BO \leq 14$.

To fulfill these functionalities, a GTS DataBase structure has been created. By means of appropriate classes the database is instantiated in both the coordinator and devices memories allowing for:

- preparing the list of the GTS descriptors to be attached to the beacon frame periodically broadcasted by the coordinator;
- activating the hardware timers for transmitting (receiving) data within the CFP of the MAC SuperFrame;
- enabling the GTS re-location and extending the CAP time interval when GTSs are de-allocated.

2.3 Visual tracking in RTNS

The ambitious goal of this work is to make use of a software tool suited to assess the performances of a visual tracking networked application, by means of a comprehensive (although simplified) model acting at the node and network levels.

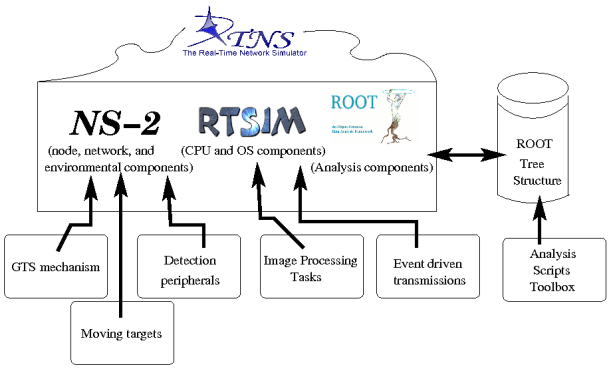

Figure 7: A pictorial view of the simulation and analysis environment for visual tracking in WSN developed within the RTNS package.

A pictorial view of the simulation and analysis environment, based on RTNS and proposed in this work, is shown in Figure 7.

The device nodes are equipped with detection peripherals like pin-hole cameras: each camera captures the portion of the background scene covered by its solid angle. The topology support offered by NS-2 has been extended to introduce self-moving entities like vehicles not involved in communication. The moving targets act as external stimuli inducing transmissions by device nodes: network activity is therefore event-driven differently from the time-driven traffic generation which is generally adopted in NS-2 based simulations.

Each micro-controller in an individual device node runs the same firmware encoding the activities related to networking and Image Processing (IP). The RTNS Kernel prototype (implemented in the RTNSApp class) abstracts the services related to the scheduling policy and resource access. We define task the computational unit corresponding to one activity. A task consists of a set of instructions that, when executed, book the CPU for a finite amount of processor ticks. The tasks can be run concurrently at a node.

The IPApp class is the specialization of the RTNSApp base class used in this work for instantiating the node kernel. IPApp handles the I/O from the peripherals and executes a S/W task customized for Image Processing (IPTask). Together with IPTask, the SendTask and ReceiveTask implementing the Network layer functionalities for data exchange are spawned at the start-up of the device nodes.

The IPTask task is a periodic activity, parametrized with a likely number of lines of code (Execution Time) and a Period set to the in-

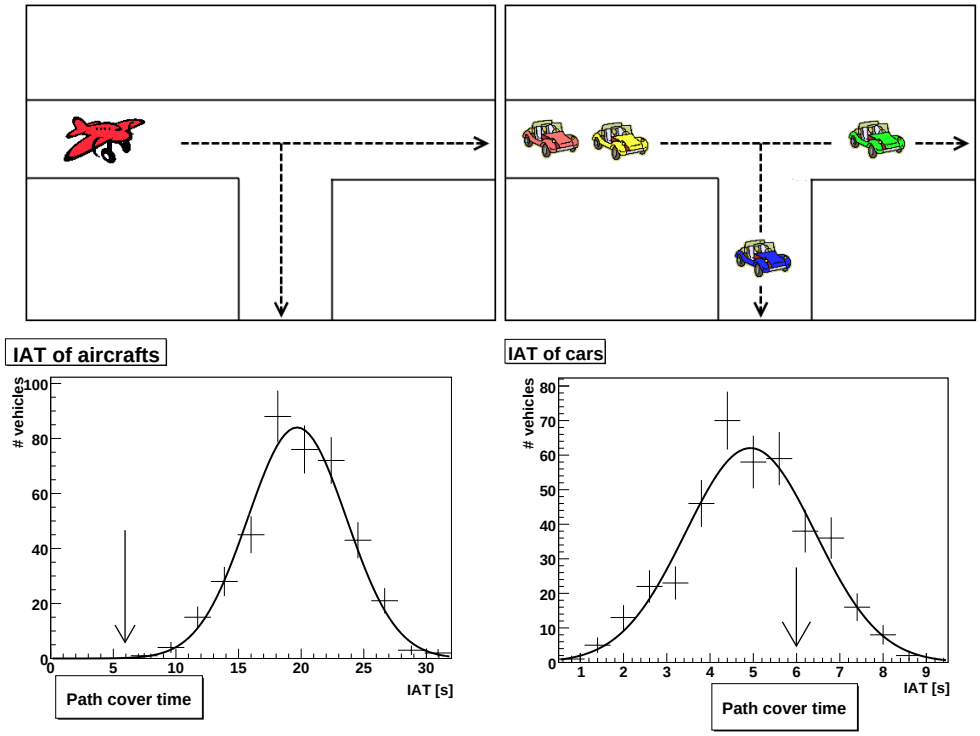

Figure 8: The Inter Arrival Time (IAT) distribution of vehicles in the taxiway and parking scenarios. The (fixed) path cover time is superimposed in the graphs.

verse of the maximum frame rate (T = $\frac{1}{\sharp fps}$) of the camera. When the permanence of an object inside the view of a camera is such that the processing task intercepts it, a notification in the shape of unicasted message ("a report") is sent to the network coordinator. The report transmission is handled by the IPAgent class which represents the UDP-like endpoint of the network stack.

The coordinator is required to collect all the reports in a Detection Window (DW) to take action. The CoordApp class (abstracting its kernel) handles the I/O from the transceiver. The UDP-like agent CoordAgent correlates all the received reports ending in the same DW, and, from their unique signature, predicts the target track on the topological grid. These mechanisms are described by means of UML sequence diagrams in Figure 5. The data coming from the reports are organized in a Tree structure (eased by the on-line availability of the ROOT[2] package classes in RTNS) which is saved on disk during the RTNS simulation runs. This structure is then accessed off-line by means of an analysis toolkit to extract the performances of the system in terms of global metrics as it will be discussed in Section 4.

3. SIMULATED SCENARIO

3.1 Event distribution models

In Figure 9, five nodes build up a WSN and are in charge of tracking a target along two possible directions. The camera views do not overlap and the object provides a unique signature in case of going straight (reports from nodes 0,1,2) or turning right (reports from nodes 0,1,3).

We simulated two scenarios with different statistical distributions of event Inter-Arrival Times (IAT) having fixed the time needed by the target to travel across the camera views (path covered time): a taxiway in a big airport, and a parking area (see Figure 8).

In the first case the distributed system provides a critical service where events expectations are rare with respect to path covered time. If the IAT are Gaussian distributed, this analytically translates to the constraint:

$$\Delta T^{min}(events) = \langle \Delta T \rangle - 3 \cdot \sigma > \frac{S}{V} \qquad (1)$$

being S, V, $\langle \Delta T \rangle$, and σ respectively the path length, the target speed, the average value and the standard deviation of the IAT probability distribution. In this simplistic model, all aircrafts move at the same speed thus path covered times are deterministically computable.

If we drop the constraint 1, we can have a superposition of arrival events in the scope of the cameras ending in the generation of reports related to independent detections.

The distributed system we keep unchanged from the code perspective responds differently in the two cases as we will see in the remaining part of this section.

3.2 The Data AcQuisition model (DAQ)

We model two different types of DAQ, the time-based and event-based (see Figure 10). Since the expected signatures are {0,1,2} or {0,1,3} , if the chain of the events is truncated and the reports split into subsequent DWs, the event is discarded.

The time based DAQ is purely hardware, based on a timer activating at constant intervals the detection window. The reports related to the same event arrive in random order being the DW totally uncorrelated with the report arrivals.

Alternatively we can elect Node 0 as the DAQ trigger since all incoming vehicles pass through its hardware view. Following this

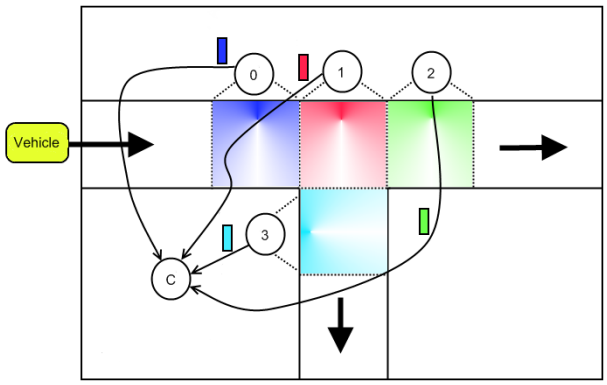

Figure 9: The scenario used for RTNS simulation. Nodes 0,1,2,3 track a vehicle using embedded pin-hole cameras covering complementary views. They send detection reports to the coordinator whenever they recognize the target entered their camera view.

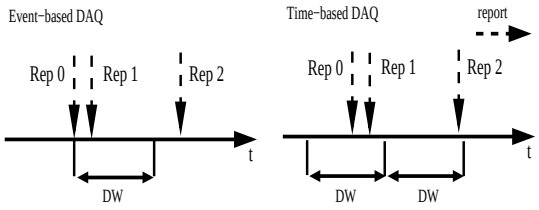

Figure 10: The DAQ logic: the event-based opens the DW at the arrival of a report from Node 0; the time-based opens the DW regularly as a function of the time.

argument we model an event-based DAQ where the DW is opened at the arrival of a report from Node 0.

3.3 A numerical example

In Figure 11 we reported a numerical example for the tracking of a single vehicle which showed up in the view of camera 0 at time 4.3 s, it then moved to the camera 1 view at time 6.3 s and decided to straight entering the camera 2 view at time 8.3 s. It finally left the topological grid at time 10.3 s having a path covered time of 6 s.

In all the nodes the IPApp kernel schedules some tasks concurrently. The camera module installed at the wireless devices is supposed to have a jitter-free frame rate equal to 2 fps. The IPTask is chosen to execute for 250 ms every 500 ms.

If the vehicle is recognized as not (already) present in memory, the sendTask is activated to transmit the report to the coordinator which processes it according to the adopted DAQ mechanism.

Given these parameters (vehicle appearance time, path covered time, CPU activity in the device nodes affecting the actual start time of the tasks), in the figure we show the response (track detected or miss) of the coordinator node for the two DAQ algorithms. The vehicle appeared at 4.3 s and crossing the scene in 6 s is detected in the case of Event-based DAQ whereas it is missed in the Time-based.

3.4 Performance metrics and statistical considerations

For simplicity (and considering statistical effects only although systematics is expected to play a relevant role in this context) we intend the visual tracking system as a generalization of a detector providing binary results (0: track detected, 1: track missed).

In a real-time perspective, the system overhead for event detection is very relevant. We imagine that controller reacts whenever the DW is closed, so that its *response time* for event i happening at time T^i_{event} is:

$$R^i = T^i_{detect} - T^i_{event}\,. \tag{2}$$

Its average value is given by:

$$\bar{R} = \frac{\sum_{i=0}^{k} R^i}{k} \tag{3}$$

being k the number of detected events when n is the true number of events in the WSN scope. To estimate V(R) we take the experimental Root Mean Square of R:

$$V(R) = \frac{\sum_{i=0}^{k} R^{i^2}_{detect} - \bar{R}^2}{k-1} \tag{4}$$

The probability of detecting k tracks out of n by a system having efficiency ε is given by[24]:

$$\begin{aligned} P(\varepsilon;k,n) &= (n+1)\binom{n}{k}\varepsilon^k(1-\varepsilon)^{n-k} \\ &= \frac{(n+1)!}{k!\,(n-k)!}\,\varepsilon^k(1-\varepsilon)^{n-k}\,. \end{aligned} \tag{5}$$

To estimate ε we take the first momentum of the probability distribution function:

$$\begin{aligned} \bar{\varepsilon} &= \int_0^1 \varepsilon\; P(\varepsilon;k,n)\; d\varepsilon \\ &= \frac{(n+1)!}{k!\,(n-k)!}\int_0^1 \varepsilon^{k+1}(1-\varepsilon)^{n-k}d\varepsilon \\ &= \frac{k+1}{n+2} \end{aligned} \tag{6}$$

which tends to $\frac{k}{n}$ when n is large. The variance of ε is defined as:

$$\begin{aligned} V(\varepsilon) &= \overline{\varepsilon^2} - \bar{\varepsilon}^2 \\ &= \int_0^1 \varepsilon^2\; P(\varepsilon;k,n)\; d\varepsilon - \bar{\varepsilon}^2 \\ &= \frac{(k+1)\;(k+2)}{(n+2)\;(n+3)} - \frac{(k+1)^2}{(n+2)^2}. \end{aligned} \tag{7}$$

Following this formalism, we avoid the artefacts of having V(ε) = 0 in the two extreme cases of fully efficient ($k = n$) and fully inefficient ($k = 0$) systems as it would have been adopting a pure binomial distribution. In fact one reasonably finds:

$$V(\varepsilon)|_{k=0,n} = \frac{n+1}{(n+2)^2\;(n+3)} > 0.$$

For large n in this case the variance becomes $\lim_{n\to\infty} V(\varepsilon) = 1/n^2$.

4. SIMULATION RESULTS

4.1 Parameters and metrics

In the scenario sketched in Figure 9, the node coordinator starts up the PAN with $BO = SO = 4$. Depending on the adopted DAQ

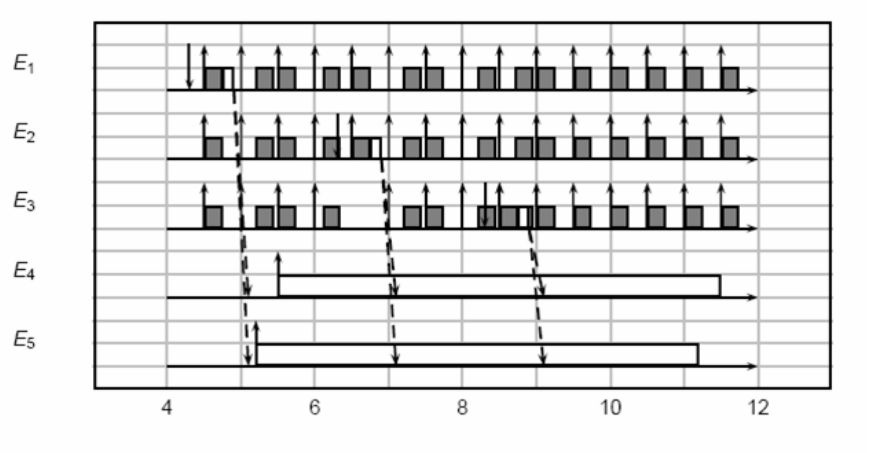

Figure 11: In the plot the CPU time lines of Nodes 0,1,2 are shown in the top rows ($E_{1,2,3}$). The vehicle appearance in the camera views is marked by an arrow pointing downwards. The IPTask activation is marked by an arrow pointing upwards and its execution by a grey-filled rectangle. The ReceiveTask is activated by IPTask when the vehicle image is processed and its execution is shown by an empty rectangle. The bottom rows refer to the CPU activity of the coordinator implementing the DAQ models of time-driven(E_4) and event-driven (E_5). The empty rectangle is DW large. The event is missed in E_4 and detected in E_5.

scheme, it combines the detection reports coming from the device nodes to track the incoming vehicles. A road traffic of 400 vehicles is simulated in each run. The report size, injected by nodes into the medium at each novel detection, is set to 100 bytes unless explicitely mentioned. This means that, at the nominal bit rate of 250 Kbps (in the 2.4 GHz frequency band) and neglecting the overhead for medium access, the time needed to transmit a report fits inside a MAC slot in a SuperFrame.

The R and ε metrics defined in Equations 2 and 5 will be evaluated by the estimators defined in Equations 3 and 6. The associated statistical error bars shown in the plots are calculated from Equations 4 and 7.

The system performances are evaluated as a function of some strategies concerning data acquisition schemes, task scheduling and communication protocols. These results are obtained making use of the analysis tools provided together with the RTNS package and are presented inhere to prove the effectiveness of the suite for quantitative studies.

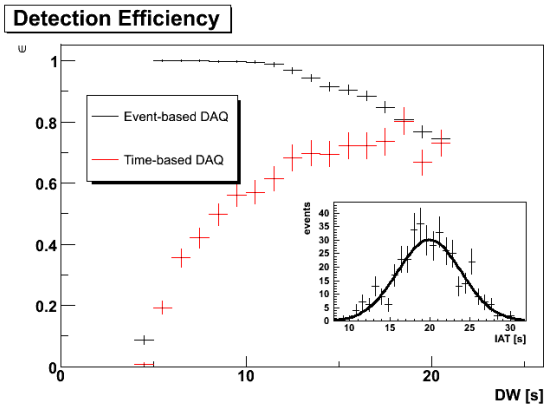

Figure 12: Efficiency behavior as a function of the DW width for the two DAQ models. The embedded plot is a reminder of vehicle IAT.

4.2 Taxiway measurements

4.2.1 The effects from the Detection Window

In this simulation study we focus on the system performances as a function of the detection window width and the adopted DAQ scheme. We consider CSMA/CA for medium access and First Come First Served (FCFS) for task scheduling at the nodes kernels.

In Figure 12 the system efficiency is plotted against the DW size. For event-based DAQ, the efficiency is maximum when DW is of the order of the path cover time ($\frac{S}{V} = 6$ s). As expected the performances degrade as the DW size increase and thus reports coming from independent detections are mixed up.

If we adopt a time-based DAQ, reports arrivals and DW are uncorrelated. As the DW increases we collect more events reaching the ratio of 70% with DW = 20 s.

Depending on the type of DAQ adopted at the coordinator, two different set-points are suggested in this analysis (DW = 6 s for Event-based DAQ and DW = 20 s for Time-based DAQ). Moreover the following relation holds:

$$\varepsilon_{TB} \leq \varepsilon_{EB} \tag{8}$$

where ε_{TB} is the efficiency achievable with a time-based DAQ algorithm and ε_{EB} is the corresponding event-based calculated fixing the DW value.

As a side effect of the uncorrelation between reports arrivals and DW, the average response time is strictly smaller in the case of time-based DAQ with respect to event-based because for the subsample of detected events, the DW results already open at the arrival of the first report thus reducing the response time of the system. This effect is formulated as:

$$\bar{R}_{TB} \leq \bar{R}_{EB} \tag{9}$$

Of course from the design requirements of the visual tracking system (in terms of efficiency and latency) it is possible to select the appropriate DAQ profile privileging either the efficiency or the latency.

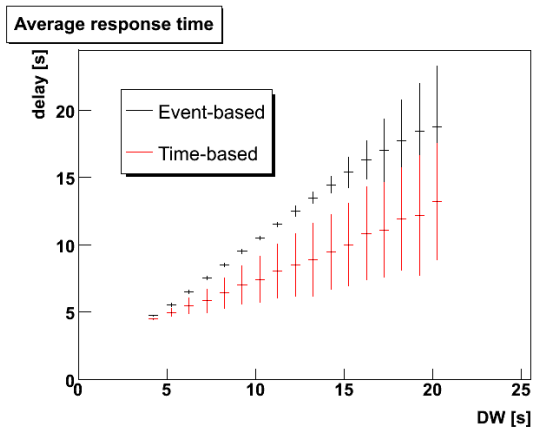

Figure 13: Average response time distribution for Event-based and Time-based DAQ models as a function of the DW width.

4.2.2 *The effects from the transmission schedule*

In Section 1 we discussed the novelty of this work because of the support for the GTS in our simulation tool. Making use of the mechanism implementation as presented in Section 2 we can differentiate real-time and best-effort traffic sources in simulation and assess the benefits of communication over guaranteed band.

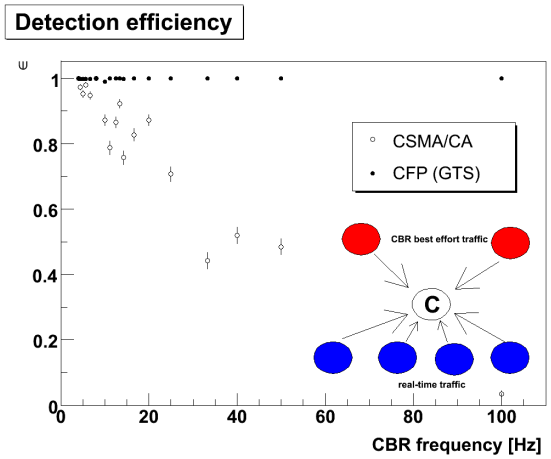

Figure 14: Efficiency as a function of the CBR rate of two disturbing nodes for the case of CSMA/CA (empty markers) and the GTS (filled markers).

In Figure 14 the efficiency (obtained making use of event-based DAQ) is plotted against the rate of disturbance introduced by two nodes generating CBR traffic with tuneable frequency. On the X-axis the nominal rate for one node is reported, thus the disturbance rate felt by the controller is actually the double.

The disturbing nodes attempt to inject 100 bytes packet frames into the network at regular intervals. For example, at a disturbing frequency of 40 Hz, this translates into 64 Kbps demand having the network 250 Kbps as total capacity.

The nodes are associated to the coordinator and transmissions are done at the same frequency as regulated by the IEEE 802.15.4 standard for the star-shaped networks. As it can be seen (empty markers in the plot), although the working conditions have been selected to produce a fully efficient set-up, this is true only in absence of parallel data flows. Already at a disturbance frequency of 40 Hz, using the CSMA/CA schedule for message transmission, the actual value of the system efficiency is about half of the nominal.

If we schedule the traffic related to visual tracking during the CFP, and the concurrent best effort traffic during the CAP, we permit parallel flows in the network without worsening the performances of the guaranteed services.

In this simple case study, we statically allocate a GTS, 2 slots long (to let the transmission report fit into it), to each device equipped with camera. The system is found insensitive to disturbances and the nominal value for the efficiency (filled markers in the plot) is achieved regardless of any non-real time activity present in the network.

4.2.3 *The effect of the CPU load*

So far the multi-tasking capabilities of the kernels did not play any role because the "optional" transmission of the report on IP-Task completion can be easily coded as an ordinary code branch.

Suppose the device nodes run other activities concurrently with visual tracking (for example self-diagnosis, error reporting, etc.). The effects on global metrics depend on the scheduling policy adopted in the kernels, notably upon their preemptive capabilities.

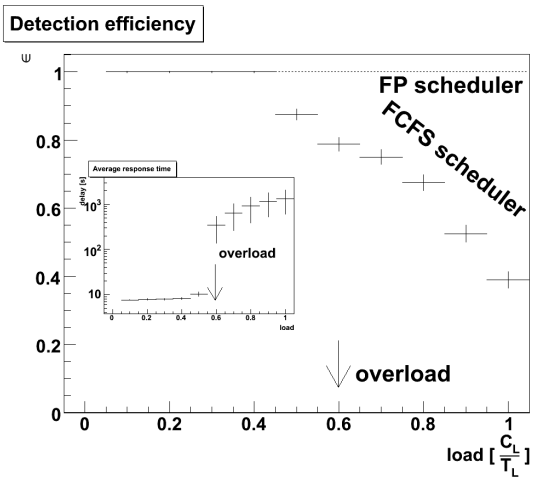

Figure 15: Efficiency as a function of CPU load factor in device nodes for Fixed Priority (FP) and First Come First Served (FCFS). The embedded frame contains the distribution of the Average Response Time in semi-log scale for FCFS. Overload condition occurs when $\frac{c_L}{T_L} = 60\% > 50\%$.

In Figure 15 the system efficiency and latency are tracked against the computational load introduced by a background task scheduled concurrently with the IPTask. As reported in Section 3.3 the IPTask provides a standalone load of $\frac{c_{IP}}{T_{IP}} = 50\%$, so that the overload condition is reached whenever additional load exceeding 50% is scheduled on the node.

If the kernel has no real-time functionality as in the case of TinyOS[23], as the extra load increases, the absolute response time and its jitter increase. Moreover, as the overload condition is met, the system starts missing events and malfunctions show up like that of events being detected after the appearance of many others: in the plot of Latency versus Background load, the trend is discontinuous at the overload condition where response time jumps by two orders

of magnitude meaning that the aircraft track is recognized after 300 s (5 minutes) from its appearance.

The visual tracking system deployed on top of non real-time kernels does not respect safety critical constraints with respect to background task. This result encourages to adopt real-time kernels like ERIKA[6] and Nano-RK[1] supporting Fixed Priority scheduling whenever the nominal performances must be guaranteed in variable (or even unpredictable) conditions of CPU load.

4.3 Parking area measurements

4.3.1 The effects from bandwidth limitations

When the vehicle inter-arrival times are smaller, events are more frequent and the average network traffic generated by report transmission gets larger. The effect is stronger if the inter-arrival time is comparable with path cover time; in such a case reports related to different vehicles are injected concurrently into the network by different device nodes.

The limitation in bandwidth for the low rate nature of the IEEE 802.15.4 standard prevents the system from the full efficiency. Selecting the most favorable options for the DAQ (the event-based), the system achieves 80% efficiency with a DW equal to 20 s (empty markers in Figure 16). Higher values for the DW have not been explored for reasons related to latency matters.

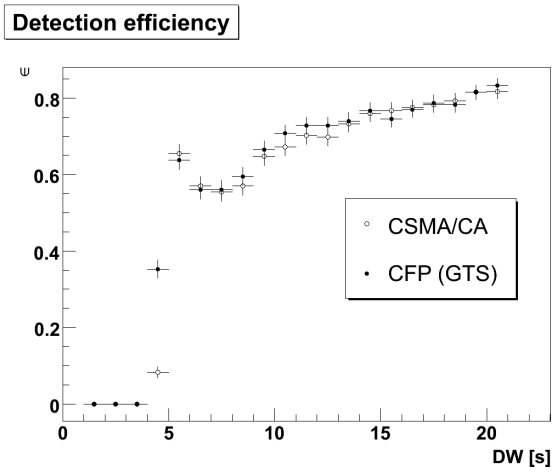

Figure 16: Efficiency behavior as a function of the DW width for the case of CSMA/CA (empty markers) and the GTS (filled markers).

As it can be seen from the plot, the GTS mechanism does not improve the system performances (filled markers in the plot) and the two curves are statistically compatible. The non-monotonic behavior is explained by the fact that the system starts working properly when the detection window is larger than the path covered time.

4.3.2 The effects from report fragmentation

The extra-value provided by the GTS is felt when the report size is such that fragmentation is needed at the Network layer. When the message is composed by more than one packet, using the CSMA/CA mechanism, the node must access the channel and back-off by a random period at least twice per packet. Given a certain probability of packet delivery, the larger the report is in number of packets, the higher the probability of getting incomplete reports at the sink is.

It is worth mentioning that in the CSMA/CA mechanism standardized in IEEE 802.15.4, the MAC layer tries to inject into the medium a new packet for a maximum number of attempts (equal to 4 in our simulation runs) before dropping it; this option is introduced for fault safety reasons to avoid saturation in the Link Layer queues.

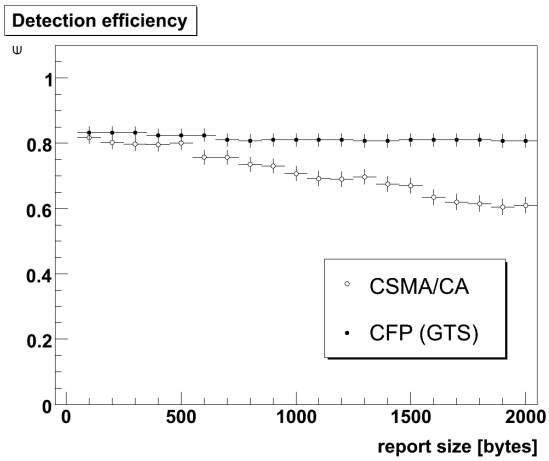

Figure 17: Efficiency as a function of the report size for the case of CSMA/CA (empty markers) and the GTS (filled markers).

In Figure 17 the nominal value of 80 % is reached against a large set of values for the report size only adopting the GTS mechanism (filled markers in the plot). The points referring to the CSMA/CA mechanism (empty markers in the plot) show on the contrary a drop in the performances of the order of 10% as the report size reaches 1000 bytes (10 packets).

The fragmentation introduced at Network layer (performed at the NS-2 Agent layer) is realistic because, for the visual tracking protocol, the reports might include a portion of the detected vehicle transmitted in the form of raw data: in this case a payload of 1000 bytes corresponds to a square of 332 pixels maximum, as follows from the relation:

$$N(\text{pixels, RGB, 1 byte/color}) = \frac{1000-3}{3} \simeq 332 \text{ bytes} \quad (10)$$

where 3 bytes have been reserved to locate the square inside the camera view (X and Y of the top-left corner and size of the square). Even when the lowest resolution mode is selected for camera operations, say 80×60 pixels, this corresponds to about 7% of the camera view only[17].

When fragmentation is included, unless sophisticated algorithms are coded at the coordinator to reassemble the data in lossy conditions, the GTS mechanism prevents the system to work inefficiently.

5. CONCLUSIONS

Wireless Sensor Networks are becoming more and more intelligent extending the domain of interest to real-time applications. Building up such kind of systems requires support for bounding the delay in packet transmission and real-time scheduling policies at the node kernel level. Distributed imaging techniques are particularly charming for the intrinsic load they infer at both network (for

the volume of exchanged packets) and CPU levels (for implementing imaging techniques in low cost and power constrained hardware products). Simulation is very useful in the design of envisioned applications. RTNS, for its capabilities coming from the upgraded NS-2 (enriched by the GTS mechanism of IEEE 802.15.4 mechanism) and RTSim simulation packages, permits to assess the performances of a simplified networked application for visual tracking.

In this paper we discussed the performances in two case studies of vehicles running in a taxiway or in a parking area. We showed that the efficiency and response time strongly depend on algorithms instantiated in the nodes for the scheduling of the S/W tasks, and on medium access paradigms (contention based or contention free) adopted to transmit data. Moreover set-up parameters as the DAQ model in the system coordinator and the length of the report transmitted at vehicle detection are very relevant and deviate the metrics from the results obtained using a naive approach.

As a future work we want to make use of RTNS for validating dynamic bandwidth allocation mechanisms implemented in the Network layer on top of the IEEE 802.15.4 MAC and Physical layers. In realistic scenarios we can assess how effective are the proposed protocols when including, on one hand redundancy of information over set of nodes, and on the other hand the possibility of having faulty, missed or late reports from the end devices.

6. REFERENCES

[1] A. Eswaran, A. Rowe and R. Rajkumar. Nano-rk: An energy-aware resource-centric operating system for sensor networks. In *Proceedings of IEEE Real-Time Systems Symposium*, 2005.

[2] R. Brun and F. Rademakers. ROOT. An Object Oriented Data Analysis Framework. `http://root.cern.ch`.

[3] A. Cervin, M. Ohlin, and D. Henriksson. Simulation of networked control systems using TrueTime. In *Proc. 3rd International Workshop on Networked Control Systems: Tolerant to Faults*, Nancy, France, June 2007. Invited talk.

[4] L. Chen, Z. Chen, and S. Tu. A realtime dynamic traffic control system based on wireless sensor network. In *ICPPW '05: Proceedings of the 2005 International Conference on Parallel Processing Workshops*, pages 258–264, Washington, DC, USA, 2005. IEEE Computer Society.

[5] 802.15.4 and ZigBee Routing Simulation at Samsung/CUNY. `http://www-ee.ccny.cuny.edu/zheng/pub/file/WPAN_ZBR_pub.pdf`.

[6] E.R.I.K.A. `http://erika.sssup.it/`.

[7] S. Ivanov, A. Herms, and G. Lukas. Experimental validation of the ns-2 wireless model using simulation, emulation, and real network. In *Proceedings of the 4th Workshop on Mobile Ad-Hoc Networks (WMAN'07)*, pages 433 – 444. VDE Verlag, 2007.

[8] P. Jurčík, A. Koubaa, M. Alves, E. Tovar, and Z. Hanzálek. A Simulation Model for the IEEE 802.15.4 Protocol: Delay/Throughput Evaluation of the GTS Mechanism. In *Proceedings of MASCOTS 2007, 15th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, pages 109–116, Piscataway, 2007. IEEE.

[9] 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.

[10] G. F. Lucio, M. Paredes-Farrera, E. F. Jammeh, and M. J. Reed. Opnet modeler and ns-2 - comparing the accuracy of network simulators for packet-level analysis using a network testbed. *WSEAS Transactions on Computers*, 2(3):700–707, July 2003.

[11] D. Malan, T. Fulford-jones, M. Welsh, and S. Moulton. Codeblue: An ad hoc sensor network infrastructure for emergency medical care. In *In International Workshop on Wearable and Implantable Body Sensor Networks*, 2004.

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

[13] The ns Manual (formerly known as ns Notes and Documentation). `http://www.isi.edu/nsnam/ns/ns-documentation.html`.

[14] OPNET Technologies, Inc., Bethesda, MD, USA. The OPNET Simulator. `http://www.opnet.com/`.

[15] P. Pagano, P. Batra, and G. Lipari. A Framework for Modeling Operating System Mechanisms in the Simulation of Network Protocols for Real-Time Distributed Systems. In *"Proceedings of the 21st IEEE International Parallel and Distributed Processing Symposium (IPDPS)"*, Long Beach (CA), USA, 2007.

[16] P. Pagano, M. Chitnis, and G. Lipari. RTNS: an NS-2 extension to Simulate Wireless Real-Time Distributed Systems for Structured Topologies. In *Proceedings of WICON 2007*. ACM Press, Oct. 2007.

[17] P. Pagano, C. Nastasi, and Y. Liang. The Multivision problem for Wireless Sensor Networks: a discussion about Node and Network architecture. In *International Workshop on Cyber-Physical Systems Challenges and Applications (CPS-CAŠ08). Proc. of the DCOSS 2008 conference.*, Santorini island, Greece, June 2008. Invited talk.

[18] L. Palopoli, G. Lipari, G. Lamastra, L. Abeni, G. Bolognini, and P. Ancilotti. An object oriented tool for simulating distributed real-time control systems. *Software: Practice and Experience*, 2002.

[19] The RI-MACS EU project (NMP2-CT-2005-016938). `http://www.rimacs.org`.

[20] A. Rowe, A. Goode, D. Goel, and I. Nourbakhsh. CMUcam3: An Open Programmable Embedded Vision Sensor. Technical Report RI-TR-07-13, Carnegie Mellon Robotics Institute, 2007.

[21] The RTNS simulation suite. `http://rtns.sssup.it`.

[22] The RTSim simulator. `http://rtsim.sf.net`.

[23] University of California, Berkeley CA, USA). The TinyOS operating system. `http://www.tinyos.net/`.

[24] T. Ullrich and Z. Xu. Treatment of errors in efficiency calculations. `http://www.citebase.org/abstract?id=oai:arXiv.org:physics/0701199`, 2007.

[25] A. D. Young, M. J. Ling, and D. K. Arvind. Orient-2: a realtime wireless posture tracking system using local orientation estimation. In *EmNets '07: Proceedings of the 4th workshop on Embedded networked sensors*, pages 53–57, New York, NY, USA, 2007. ACM.

[26] J. Zheng and M. J. Lee. A comprehensive performance study of ieee 802.15.4. In *Sensor Network Operations*, pages 218–237. IEEE Press, Wiley Interscience, 2006.

[27] J. Y. Zheng and S. Sinha. Line cameras for monitoring and surveillance sensor networks. In *MULTIMEDIA '07: Proceedings of the 15th international conference on Multimedia*, pages 433–442, New York, NY, USA, 2007.