WiSE-MNet: an experimental environment for wireless multimedia sensor networks

Christian Nastasi [1], Andrea Cavallaro [2]

[1] *Scuola Superiore Sant Anna, Piazza Martiri della Libertà, 56127 Pisa (Italy)*
[2] *Queen Mary University of London, Mile End Road, E1 4NS London (United Kingdom)*
c.nastasi@sssup.it; andrea.cavallaro@eecs.qmul.ac.uk

Abstract—**We propose a simulation environment for networks for Wireless Multimedia Sensor Networks (WMSNs), i.e. networks with sensors capturing complex vectorial data, such as for example video and audio. The proposed simulation environment allows us to model the communication layers, the sensing and distributed applications of a WMSN. This Wireless Simulation Environment for Multimedia Networks (WiSE-MNet) is based on *Castalia/Omnet++* and is available as open source to the research community [1]. The environment is designed to be flexible and extensible, and has a simple camera model that enables the simulation of distributed computer-vision algorithms at a high level of abstraction. We demonstrate the effectiveness of WiSE-MNet with a distributed tracking application.**

I. Introduction

Recent technological advances have favoured the evolution of traditional Wireless Sensor Networks (WSNs) operating on simple scalar measurements towards Wireless Multimedia Sensor Networks (WMSNs) dealing with more complex vectorial data such as video and audio. Although the literature on WMSNs has mainly focused on multimedia streaming [2] and bandwidth allocation [3], new applications such as human behaviour recognition and wide-area target tracking are demanding the definition of cooperative distributed applications for WMSNs.

The understanding and the modeling of the complexity of distributed applications based on WMSNs require competences from several areas, ranging from networking to control theory, and from computer vision to data management. However, until now researchers have studied algorithms, applications and protocols for WMSNs without an holistic approach that addresses complementary and interconnected issues from all these disciplines. In particular, one of the main obstacles towards an integrated approach is the *lack of a common simulation framework where all the aspects related to different disciplines can be modeled and analysed simulteneously*. Attempts have been made in this direction, specifically tailored to surveillance applications [4].

In this paper we present WiSE-MNet, a Wireless Simulation Environment for Multimedia Networks. WiSE-MNet is a generic network-oriented simulation environment that addresses the need for co-design of network protocols and distributed algorithms for WMSNs. To the best of our knowledge, this is the first unified environment for WMSNs that exhaustively addresses networking issues as well as computer vision and distributed application constraints. A modeling of the WMSNs is presented considering camera nodes with a simplified camera model. A distributed tracking algorithm is also developed over the proposed simulation environment to demonstrate the effectiveness of WiSE-MNet in modeling cooperative applications.

The work reported in this paper was carried out while the first author was visiting Queen Mary University of London

The paper is organized as follows. In Section II we survey the state of the art for network simulators and we discuss the three main approaches adopted by the computer vision community to simulate distributed algorithms. Section III presents our extensions to the *Omnet++* simulation framework and the *Castalia* networking model and a discuss a sample experimental setup based on a distributed target tracking application. We finally draw conclusions in Section IV.

II. Existing simulators and their limits

In this section we first discuss the state of the art in network simulators and then we focus on the main approaches adopted in computer vision to simulate distributed applications with WMSNs.

A. Network simulators

Network simulators including NS-2, Omnet++, TrueTime, and OPNET [5], [6], [7], [8] can be compared based on (i) the programming language they use, (ii) their scriptability, (iii) the support for standard protocols, and (iv) their extendibility/integrability. The last two properties of a network simulators are important for the appropriate modeling of local processing at each node.

NS-2, the most popular network simulator [5], is written in C++ and has a simulation interface based on TCL scripts. It was originally designed for simulations based on TCP/IP computer networks, but extensions have been provided to support mobile ad-hoc networks. Mobility of nodes requires further extensions, and mixing wired and wireless nodes in the same simulation is not straightforward. Since NS-2 was originally designed for networks of computers (high-end systems), the network stack of the nodes is more complex than in actual WSNs. Support for the IEEE 802.11 and IEEE 802.15.4 standards is provided by external contribution modules. An extension of the NS-2 simulator is the Real-Time Network Simulator (*RTNS*) [9], which extends the support for the IEEE

802.15.4 (the *de-facto* standard for WSNs) and adds support for modeling local processing (simulating operating systems).

Omnet++, a component-based network simulator designed for hierarchical nested architectures [6], provides a discrete-event simulation engine and a GUI interface. The modules are written in C++, while a high-level Network DEscription (NED) language is used to assemble modules in components and to interconnect components to each other. A configuration file is used to specify the parameters of the simulation and the parameters of the modules. Omnet++ is highly flexibile and the simulator can be integrated with external libraries, such as API to access Matlab from C++ programs and framework with advanced statistical analysis. Modeling extensions are provided to support for example mobility, wired/wireless standard and non-standard protocols, and energy models. Different simulation packages provide an implementation of the IEEE 802.15.4 standard. To the best of our knowledge, there is no available extension that integrates OS aspects in Omnet++, although an attempt can be found in [10].

TrueTime, a Simulink toolbox developed for Matlab [7], has been designed to test the effect of task scheduling on control algorithms, and it also includes simple high-level network models. Although TrueTime cannot be considered a full network simulation environment, networking aspects could be added to already existing Matlab-based simulations. The toolbox can be integrated with any type of Matlab functions, since the activities performed by the nodes have to be coded as Matlab m-files or s-functions. A simple implementation of the IEEE 802.15.4 standard is also included.

Finally *OPNET*, a commercial network simulation software [8] whose source code is based on C++, has a GUI to configure the simulation scenarios and to develop network models. There are three level of configuration, namely the network level (to create the topology); the node level (to define the behaviour of the node and the data flow among the node components); and the process level (to define the protocols through a formalism based on finite state machines).

A comparative review of the most popular network simulators is presented in [11].

B. *Early simulation approaches for WMSNs*

Three main simulation categories can be identified based on the level of abstraction with respect to the vision problem used to validate distributed computer vision algorithms. These category are frameworks based on a simplistic world assumption, on virtual reality or on using real-world datasets directly.

Frameworks using a *simplistic world* approach assume point-like objects that have basic projection models in the image plane of the cameras and move on a ground plane with or without obstacles and boundaries (Figure 1). The use of point-like objects simplifies the problem of extracting features from objects. This approach is useful when one wants to focus on high-level problems, such as the coordination of a distributed application, while neglecting aspects related to the vision pipeline. For example, most works in the literature on distributed tracking with WMSNs adopt this simplistic model. An example is [12] for distributed target tracking with smart camera networks. The authors show a Matlab simulation with a simplistic communication mechanism. Cameras have a field of view identified by an angle on the 2D world plane and the connectivity among cameras is based on circular communication ranges.

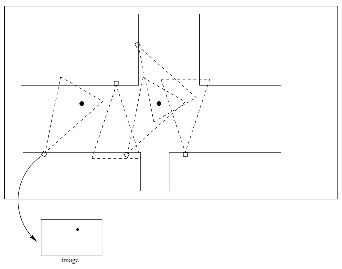


Fig. 1. Example of simplistic world simulation using a poin target and a simple 2D space

A second level of abstraction is reached when using *virtual reality* environments. In this case the objects and the surveillance areas are more complex, and the camera sensors produce more realistic synthetic images. *Virtual Vision* [13] can reproduce complex situations by simulating scenarios where recording real images is difficult, expensive or not feasible. However, the effectiveness of this framework depends on the degree of accuracy achieved when defining and modeling the necessary details in the simulation environment.

The most realistic computer vision simulations use *real world datasets*. While this approach is very common for single-node computer vision, in distributed computer vision there are important issues to be considered in order to use the datasets in realistic communication and topology conditions within a network simulator. Basically, we must be able to reproduce the real world in the simulation environment itself, e.g. obstacles and absolute distances between nodes. For instance, a complete 3D representation of the scene should be provided together with the multi-camera dataset. To the best of our knowledge such a fine simulation approach has not yet been considered for distributed applications in WMSNs.

III. The Wireless Simulation Environment for Multimedia Networks

Our goal is to define a flexible simulation environment for distributed applications in WMSNs. To this end, in this section we describe the *Castalia/Omnet++* network simulator, our proposed extensions and an exemplificative distributed tracking application that makes use of the proposed overall framework.

A. *The* Castalia/Omnet++ *network simulator*

The simulation environment we propose is based on the *Omnet++* framework [6]. The core of the simulator is a discrete event simulation engine based on modules and message exchange among modules. A message exchange mechanism allows to define local events (self messages) and remote events (messages to other modules). In Omnet++ there is no concept of network node: everything is either a simple or a composite module. A node is generally defined as a composition of modules, and, given the aforementioned paradigm, it is possible to define local and distributed behaviors in the same way: the Omnet++ simulation core is extremely flexible in terms of definition of local and distributed elements.

Simulation models available for the Omnet++ framework are sets of predefined modules providing the user with an interface to simulate state-of-the-art network protocols. Examples of simulation models are the INET framework [14], which contains the most popular wireless and wired protocols (e.g. UDP, TCP, IP, OSPF); and the MiXiM project [15], a collection of mobile and fixed wireless networking protocols. We propose to use the *Castalia* simulation model for Omnet++ [16] as it has been designed to model distributed algorithms for classic WSNs under realistic communication conditions. Castalia enables extensions and includes advanced wireless channel and radio models; a physical process (for events) and sensing model; a model for node CPU with clock drift and power consumption; and the availability of MAC and routing protocols, including IEEE 802.15.4.

The nodes are interconnected through a *wireless channel* module, which is responsible for modeling the wireless link. Each node is also interconnected with one or more *physical processes* that model events occurring in the external environment. Each node is the composition of a communication module, a sensor manager, an application module, a resource manager and a mobility manager. The distributed algorithm to be tested has to be defined in the *application module*. From a networking view-point this is the *application layer*. The *communication module* uses a simplified network stack model based on three layers, namely radio (physical layer definition), MAC and Routing; and provides communication capabilities to the application module. The *sensor manager* is responsible for providing to the application module new samples from the external environment by interacting with the physical processes. The *mobility manager* is responsible for the location of the sensor in the simulation area (however we will consider here only sensors in fixed positions). The *resource manager* can be used to model the local resource usage such as energy consumption, memory usage, and CPU states.

B. New capabilities

Although Castalia has been wisely designed for WSNs, its adaptation to WMSNs requires extensions to the original framework. These extensions include a generalized structure to support *complex data types*, rather than simple scalars; a model for the *target* movement and for the video *sensor* (camera); an *idealistic communication mechanism* to test algorithms without considering the impact of the network, while still designing them in a distributed manner; and a simple *GUI for 2D world scenarios*. Figure 2 summarises our extensions to the network and node model.

One of the main limitations of Castalia is the sensor data-type that is forced to be a scalar value, while WMSNs are

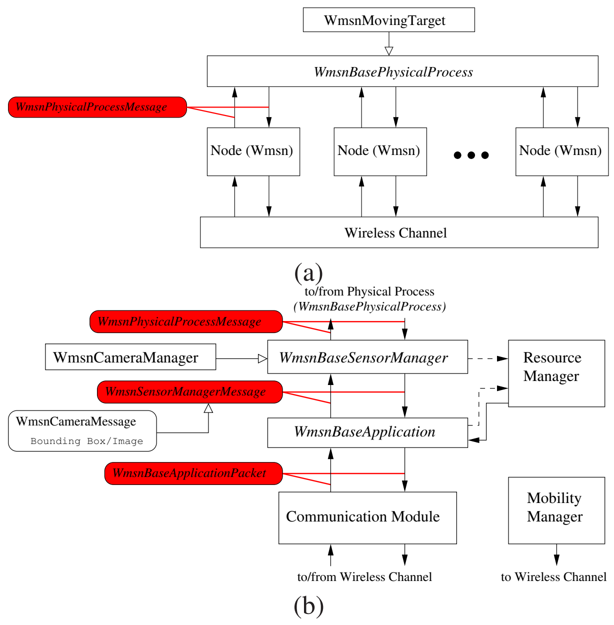


Fig. 2. *Castalia* extensions for (a) network model and (b) node model

based on vectorial data such as images. We propose a generalized sensor data-type exploiting the object oriented paradigm provided by C++ and NED. The information exchanged between the physical processes and the sensor managers is defined through a base class called *WmsnPhysicalProcessMessage* which can be specialized in concrete classes containing any type of data. To support this extension, even the physical process module has been generalized, so that we could either use a *WmsnMovingTarget* class to model the movement of a target in a 2D plane, or use a *WiseVideoFile* to feed the sensor manager with real-world video sequence. Following the same principle, the sensor manager and application modules have been extended. In this case a *WmsnCameraManger* class has been provied to model a simplistic sensor-camera, while different application classes have been developed to test different algorithms. Indeed, the *WmsnBaseApplication* is the basic application class that the user should derive to implement the logic of a distributed application.

The idealistic communication mechanism is necessary because there is no possibility in Castalia to entirely bypass the communication components. More precisely, Castalia gives the opportunity to implement a pass-through communication stack with ideal radio. Unfortunately, with this configuration, if two nodes attempt to communicate at the same time, the result will be a failure in the radio component as the Castalia ideal radio cannot send and receive at the same time. We propose an alternative module that bypasses the Castalia communication modules by interconnecting directly application layers of the nodes.

The underlying *network infrastructure* is configurable and the following setups are possible: (i) a fully connected network with infinite bandwidth and each node can communicate with all the other nodes without delays; (ii) limited transmission ranges with infinite bandwidth; (iii) limited transmission ranges with finite bandwidth (communication delays); and (iv) realistic wireless stacks, with standard protocols. Energy consumption is not currently considered, although it is already supported by Castalia.

The current version of WiSE-MNet works with a simplistic world model that can be extended to the case of real-world datasets. Similarly to [12], the surveillance area is modeled as a 2D plane and the extension to the 3D world is simplified by the support provided in Omnet++ and Castalia. Unlike [12] that represents objects as points, to model the results of an object detection stage applied to real-world images we describe objects with a (bounding) box. The cameras are assumed to be top-down facing, thus obtaining a simplified projection model. More complex placement and projection models will have no major impacts on the definition of the distributed application. The relationship among cameras (calibration, overlapping/non-overlapping fields of view) are statically defined.

C. A distributed application example

In this section we present an example where a distributed particle filter (DPF) inspired by [17], [18] is implemented in a network of cameras and is used to estimate the position of a moving target [19]. The nodes exchange information about their (partial) posterior probability using a Gaussian Mixture Model (GMM) to reduce the amount of data to be transmitted. We configure the algorithm to operate with a set of 500 particles and with a 5-component GMM approximation. The field of view (FOV) of a camera is calculated considering its view angle and assuming a distance of $6m$ from the ground plane, thus resulting approximately in a FOV of $10m \times 6m$. The camera are positioned according to a random uniform distribution, and the overall area under surveillance is $100m \times 60m$.

The communication modules provided with the Castalia package and the proposed extensions allow us to configure the network in different ways. In this example we adopt the T-MAC communication protocol [20] and we configure the protocol as follows. The *request-to-send* (RTS) and the *clear-to-send* (CTS) mechanisms are used. The RTS/CTS is a classic mechanism to avoid (or to reduce) the number of collisions in wireless communication protocols. The basic idea is that a node can transmit only when the access to the channel has been granted. The RTS/CTS mechanism has an overhead, which reduces the effective available bandwidth for transmission, although retransmissions due to collision might be saved. We use of *acknowledged transmission* as we want a packet to be delivered after a collision takes place. If for some reason (generally a collision) the packet is not received, the source node can be notified by a missing acknowledgement from the destination and then restarts the transmission. If the acknowledgements are not supported, a different mechanism to detect the failure might be required. The inactive period is designed to extend the lifetime of the nodes by periodically switching from an active state (normal operation condition) to a sleep state, where the node cannot perform any operation (including communication). The inactive period has been disabled in this example to avoid the situation in which a node attempt to communicate during the sleep mode. However, it is possible to consider a different definition of the DPF algorithm, which is aware of inactive intervals, thus saving node energy. The number of *retransmission attempts* is 10 and the *transmission bandwidth* is $250kbps$.

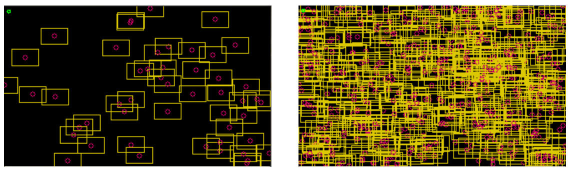
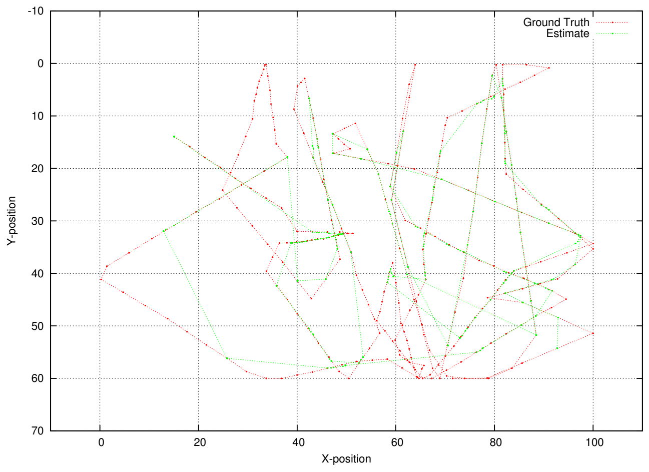

(a) (b)

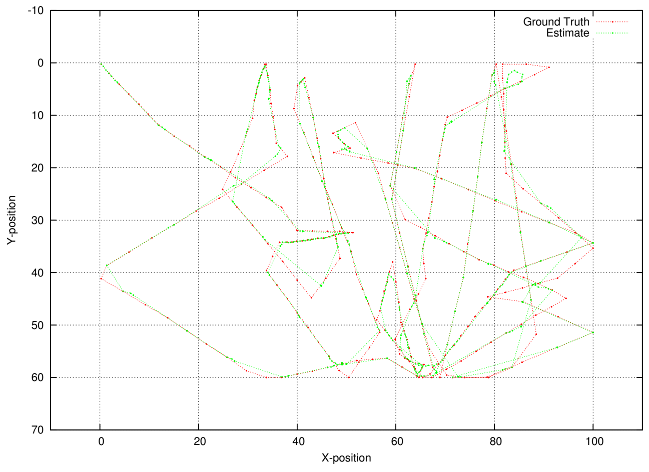
Fig. 3. Examples of sensor deployment in WiSE-MNet: coverage of the area (a) 50 nodes and (b) 500 nodes. Yellow boxes represent FOVs of the cameras, whose center is identified by a magenta circle. The green box in the top left side of each figure is the target

Let us consider two setups, one with a network deployment of 50 nodes (Figure 3(a)) and one with the number of nodes increased to 500 (Figure 3(b)). The true trajectories of the target (ground truth) and the corresponding estimation produced by the network using the DPF are shown in Figure 4. As it can be observed by comparing the results for the trajectories, the performance obtained in the deployment of 500 nodes are much better than the case with 50 nodes. This is mainly due to the poor coverage that is obtained in the latter case. When the target moves in the portion of the surveillance area uncovered by the cameras' FOVs, the estimation of its position is not possible. In case of the deployment with 500 nodes, the surveillance are is completely covered. There is also a second contribution that is related to the redundancy of camera views. In the setup with 500 nodes the number of camera observing the target at the same time is larger than the case of 50 nodes. These multiple observations are exploited by the distributed particle filter to improve the estimation with respect to few (or single) observations. Notice that this consideration is valid under the assumption that the bandwith demand related to the communication of the 5-GMM components is compatible with the available transmission bandwidth.

To understand the effect of the network delay we runned a simulation for the setup with 500 nodes, comparing two configurations of the DPF: with the GMM approximation of 5 components (5-GMM); without the GMM (0-GMM), i.e. nodes exchange the whole particle set. The results are shown in Figure 5. Although avoiding the Guassian-mixture approximation is theoretically more efficient, since the partial posterior of the DPF is represented with the entire particle set, in a realistic networking scenario the situation is different. Owing to the larger amount of data produced in case of 0-GMM, the network delay to transfer the partial posterior from a node to the other increases considerably with respect to the 5-GMM case. Consequentely, the estimation process is slower in case of 0-GMM, resulting in a loss of performances with respect to 5-GMM. A detailed analysis of efficiency and



(a)



(b)

Fig. 4. True and estimated trajectory using the DPF 5-GMM for a sensor deployment (a) of 50 and (b) of 500 nodes

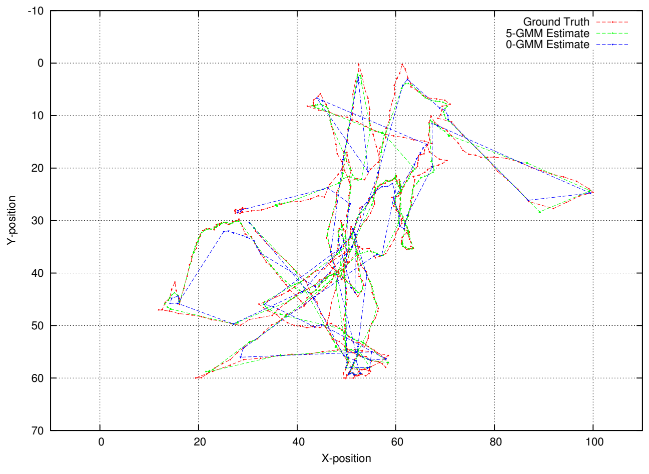


Fig. 5. True and estimated trajectory for 0-GMM DPF and 5-GMM DPF for a sensor deployment of 500 nodes

network delay is given in [19].

IV. Conclusions

In this paper we discussed the need for a simulation environment for realistic Wireless Multimedia Sensor Networks and presented WiSE-MNet, a Wireless Simulation Environment for Multimedia Networks. After considering the state of the art of network simulators, we focused on the *Castalia/Omnet++* framework and extended its basic functionalities to operate with networks of multimedia sensors, such as cameras. A distributed target tracking application, based on distributed particle filter, has been implemented to prove the effectiveness of the proposed framework. WiSE-MNet, which is available at [1], is an enabling framework for research in computer vision algorithms for realistic multi-camera networks with synthetic and real-world datasets, as well as for the study of more complex networking protocols and for bandwidth and energy management.

References

[1] "WiSE-MNet," http://www.eecs.qmul.ac.uk/~andrea/wise-mnet.html, 2011.
[2] S. Misra, M. Reisslein, and G. Xue, "A survey of multimedia streaming in wireless sensor networks," *IEEE Communications Surveys & Tutorials*, vol. 10, no. 4, pp. 18–39, 2008.
[3] M. Chitnis, P. Pagano, G. Lipari, and Y. Liang, "A Survey on Bandwidth Resource Allocation and Scheduling in Wireless Sensor Networks," *Int. Conf. on Network-Based Information Systems*, pp. 121–128, Aug. 2009.
[4] C. Pham, "A video sensor simulation model with omnet++," http://web.univ-pau.fr/~cpham/WSN-MODEL/wvsn.html, 2010.
[5] "Ns-2 project," http://nsnam.isi.edu/nsnam/index.php/Main_Page, 2010.
[6] "OMNeT++ Project," http://www.omnetpp.org/, 2010.
[7] Lund University, "Truetime project," http://www.control.lth.se/truetime/.
[8] "Opnet," http://www.opnet.com/.
[9] P. Pagano, M. Chitnis, G. Lipari, C. Nastasi, and Y. Liang, "Simulating Real-Time Aspects of Wireless Sensor Networks," *EURASIP Journal on Wireless Comm. and Networking*, vol. 2010, no. i, 2010
[10] S. Höckner, A. Lagemann, and J. Nolte, "Integration of event-driven embedded operating systems into omnet++: a case study with reflex," in *Proc. of the 2nd Int. Conf. on Simulation Tools and Techniques*, Simutools '09. ICST, Brussels, Belgium, pp. 73:1–73:7
[11] A. Timm-Giel, K. Murray, M. Becker, C. Lynch, C. Gorg, and D. Pesch, "Comparative Simulations of WSN," in *ICT-MobileSummit*, 2008.
[12] B. Shirmohammadi and C. J. Taylor, "Distributed target tracking using self localizing smart camera networks," in *Proc. of ACM/IEEE Int. Conf. on Distributed Smart Cameras*, 2010, pp. 17–24
[13] F. Qureshi and D. Terzopoulos, "A simulation framework for camera sensor networks research," in *Proc. of comm. and networking simulation symposium*, 2008, pp. 41–48
[14] "Inet framework for omnet++," http://inet.omnetpp.org/.
[15] "Mixim project for omnet++," http://mixim.sourceforge.net/.
[16] "The Castalia simulation model," http://castalia.npc.nicta.com.au/.
[17] O. Hlinka, P. M. Djuric, and F. Hlawatsch, "Time-space-sequential distributed particle filtering with low-rate communications," in *2009 Conf. Record of the Asilomar Conference on Signals, Systems and Computers*. Nov. 2009, pp. 196–200
[18] X. Sheng, Y.-H. Hu, and P. Ramanathan, "Distributed particle filter with GMM approximation for multiple targets localization and tracking in wireless sensor network," in *Proc. of Int. Symposium on Information Proc. in Sensor Networks, 2005.*, pp. 181–188
[19] C. Nastasi and A. Cavallaro, "Distributed target tracking under realistic network conditions," in *Proceedings of Sensor Signal Processing for Defence 2011*, London (UK), September 2011.
[20] T. van Dam and K. Langendoen, "An adaptive energy-efficient mac protocol for wireless sensor networks," in *Proc. of the Int. Conf. on Embedded networked sensor systems*. 2003, pp. 171–180