ScanTraffic: Smart Camera Network for Traffic Information Collection

Daniele Alessandrelli[1], Andrea Azzarà[1], Matteo Petracca[2], Christian Nastasi[1], and Paolo Pagano[2]

[1] Real-Time Systems Laboratory, Scuola Superiore Sant'Anna, Pisa, Italy
{d.alessandrelli, a.azzara, c.nastasi}@sssup.it
[2] National Laboratory of Photonic Networks, CNIT, Pisa, Italy
{matteo.petracca, paolo.pagano}@cnit.it

Abstract. Intelligent Transport Systems (ITSs) are gaining growing interest from governments and research communities because of the economic, social and environmental benefits they can provide. An open issue in this domain is the need for pervasive technologies to collect traffic-related data. In this paper we discuss the use of visual Wireless Sensor Networks (WSNs), i.e., networks of tiny smart cameras, to address this problem. We believe that smart cameras have many advantages over classic sensor motes. Nevertheless, we argue that a specific software infrastructure is needed to fully exploit them. We identify the three main services such software must provide, i.e., monitoring, remote configuration, and remote code-update, and we propose a modular architecture for them. We discuss our implementation of such architecture, called ScanTraffic, and we test its effectiveness within an ITS prototype we deployed at the Pisa International Airport. We show how ScanTraffic greatly simplifies the deployment and management of smart cameras collecting information about traffic flow and parking lot occupancy.

Keywords: intelligent transport systems, visual wireless sensor networks, smart cameras

1 Introduction

Intelligent Transport Systems (ITSs) are nowadays at the focus of public authorities and research communities aiming at providing effective solutions for improving citizens lifestyle and safety. The effectiveness of such kind of systems relies on the prompt processing of the acquired transport-related information for reacting to congestion, dangerous situations, and, more generally, for optimizing the circulation of people and goods. To obtain a dynamic and pervasive environment where vehicles are fully integrated in the ITS, low cost technologies (capable of strongly penetrating the market) must be let available by the effort

EWSN 2012, LNCS 7158, pp. 196–211, 2012. DOI=10.1007/978-3-642-28169-3_13.
The original publication is available at www.springerlink.com

of academic and industrial research: for example low cost wireless devices are suited to establish a large-area roadside network.

In this paper we describe ScanTraffic, the software managing the data collection layer of an ITS prototype developed within the Ipermob[4] project. Ipermob proposes a pervasive and heterogeneous infrastructure to monitor and control urban mobility. Its multi-tier architecture aims at the integration and optimization of the chain formed by data collection systems; aggregation, management, and on-line control systems; off-line systems aiming at infrastructure planning; information systems targeted to citizens and municipalities to handle and rule the vehicle mobility. Moreover Ipermob proposes to use visual Wireless Sensor Networks (WSNs) to collect traffic-related data. Nodes in a visual WSN are not the traditional WSN motes. Indeed, they are (tiny) smart cameras, i.e., devices equipped with a microcontroller, an IEEE 802.15.4 transceiver and a low-resolution CMOS camera. Smart cameras use image-processing techniques to extract information from the acquired image.

The main advantage of smart camera over classic sensors is versatility. An image (and a sequence of images) contains much more information than a scalar value, therefore camera-based sensors can perform a wide range of tasks, *functionally replacing* different types of sensors. For example, a smart camera can be used as a light sensor, a motion detector, an occupancy sensor, etc. Smart cameras can also *quantitatively replace* classic sensors. For example we use a single smart camera to monitor the occupancy status of up to 10 parking spaces, instead of using one inductive sensor for each space. The major drawback of smart cameras is higher cost and power consumption. Yet the higher cost per unit is offset by the reduction in the number of sensors, that also leads to cheaper deployment and maintenance. We address the power consumption problem adopting on-board image processing: only relevant information (e.g., number of counted vehicles, parking space occupancy status, etc.) is sent over the network, thus reducing the number of exchanged messages which are the main cause of energy consumption in WSNs.

Due to their peculiarities, smart cameras and visual WSNs require a software infrastructure capable of fully exploiting their features. Within the Ipermob project we addressed this issue and we developed ScanTraffic to manage the visual WSNs used to collect traffic-related data. The contribution of our work is fourfold:

- We apply visual WSNs to the ITS domain, providing a new solution having many advantages over current systems based on scalar sensors;
- We identify the minimum set of services to be provided by a software managing a visual WSN;
- We propose a software architecture for such services;
- We discuss its implementation, deployment and validation within an ITS prototype.

[4] "Infrastruttura Pervasiva Eterogenea Real-time per il controllo della Mobilità" (*"A Pervasive and Heterogeneous Infrastructure to control Urban Mobility in Real-Time"*). `http://www.ipermob.org`

The organization of this paper is as follows. Section 2 discusses related work. Section 3 provides a brief description of the Ipermob project and describes ScanTraffic software design. Section 4 describes the implementation and deployment of ScanTraffic within the ITS prototype we built at the Pisa International Airport. Section 5 discusses the experimental validation of the system. Section 6 provides concluding remarks and outlines future work.

2 Related Work

WSN technology have already proven to be an effective tool for supporting next generation ITS, enabling pervasive traffic monitoring on surface streets, arterial roads, and freeways. In [6] a WSN is used to provide automatic adaptive lighting in road tunnels with the aim of improving road safety and reducing power consumption. In [4] acoustic sensors are employed to estimate traffic flows and early detect traffic jams. Both projects have been extensively validated on real testbed deployments. VTrack [13] and ParkNet [11] are ITS solutions both based on GPS localization, using mobile nodes to collect information about traffic flows and parking occupancy. The main advantage of those systems is that a fixed infrastructure is not necessary. In VTrack smart-phones or on-board-units equipped with GPS are used to track vehicle positions which are sent to a central elaboration server. The effectiveness of the system relies on the localization accuracy and on the number of vehicles equipped with mobile nodes. In ParkNet a drive-by parking monitoring system is presented. Vehicles equipped with GPS and ultrasonic sensors move around the city detecting the presence of free parking spaces on the road side. Anyway such systems are not suitable for high mobility environments and for applications requiring high precision of the measurements. A Wireless sensor network for Intelligent Transportation System (WITS) [7] gives a systematic approach to design and deploy a WSN system to assist transportation. The WITS system makes use of three different types of wireless sensor nodes called vehicle units, roadside units and intersection units to gather and measure relevant information on vehicle and traffic movement. The system makes use of data aggregation policies depending on the request made by the traffic control execution system. TRACKSS [3] is another project employing WSNs for cooperative sensing and for predicting flow, infrastructure and environmental conditions surrounding vehicle traffic. The goal of the project is to use advanced data fusion techniques for extracting additional information from the collected data, and to develop decision support systems for intelligent transportation.

Traditional cameras are currently used on a regular basis in the transportation domain to monitor traffic and detect incidents, e.g., with observations points in critical motorways. The advances in image sensor technology have enabled the development of low cost sensor networks performing collaborative processing of rich visual information extracted from the environment. In [12] a survey on visual WSN is presented, taking into account specific problems such as on-board image processing, object tracking and detection, communication protocols etc.

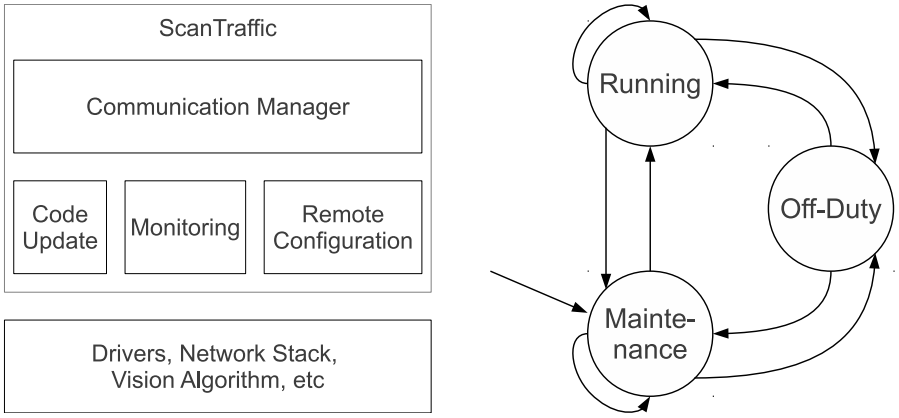
In [5] the concept of sensor network is extended to Internet enabled PC-class devices connected to off-the-shelf webcams. The system lets users query globally distributed nodes acquiring and transmitting data at a high bit rate. An example parking space monitoring application is included. Anyway, compared to ScanTraffic, resource-rich devices and high bandwidth links are used for image processing and data transmission.

Other research work has been focused on WSN in ITS. However, there exists very few real world implementation of systems employing WSNs and even less using video sensor for vehicles detecting and classification. In this regard, Ipermob presents a unique solution with video sensors giving better inference and estimation of traffic-related data.

3 System Design

The Ipermob project proposes a pervasive and heterogeneous infrastructure to monitor and control urban mobility. Within the project, a prototype of such infrastructure has been implemented and deployed at the Pisa International Airport. The Ipermob architecture has three tiers: *i*) data collection, *ii*) data sharing, and *iii*) data consumption. The data collection tier employs different technologies to acquire data related to urban mobility. Specifically, the prototype employs Vehicular Ad-hoc Networks (VANETs) and visual WSNs to collect traffic data. The data sharing tier provides, to the upper layer, a standard interface for accessing the data produced by the lower layer and stores it for future use. The data consumption tier is the application layer. Applications can be on-line control systems providing real-time information to the users (drivers, police, etc.), or off-line systems aiming at infrastructure planning. The implemented prototype provides example applications of both types.

To manage and control the visual WSNs employed in the data collection tier, we designed and implemented ScanTraffic, a distributed software infrastructure running within the WSN (i.e., in each node of the network) and providing three main services: monitoring, remote sensor configuration, and remote code-update. We argue that this is the minimal set of services to be provided by any visual WSN in order to fully exploit the potentiality of smart cameras. *Monitoring* is the primary service offered by any WSN. It allows to control the sensing activity performed by the nodes and to retrieve the collected data. *Remote sensor configuration* is often not necessary in traditional WSN employing simple sensors which can be configured (e.g., calibrated) before deployment. On the contrary, it is an important service for smart cameras whose visual algorithm needs to be configured for the current camera view, i.e., after deployment, and usually adjusted over time. *Remote code-update* is essential to benefit from smart camera versatility. By replacing sensor firmware, code-update allows to improve or completely change the functionality of a smart camera sensor. Fig. 1a shows the software architecture of ScanTraffic. There is a total of 4 software modules: 3 service modules (one for each of the main services previously discussed), plus the "Communication Manager" controlling and coordinating them.



(a) General architecture (b) Finite State Machine

Fig. 1. ScanTraffic software design.

3.1 Monitoring Module

The purpose of WSNs is to acquire measurements from the environment and transmit them to a main collection point. The way this activity is performed should take into account timeliness, energy consumption, and reliability. Such issues contrast with each other, and each application requires a specific trade-off between them. Within the Ipermob project, we use the Mirtes middleware [2] to implement the monitoring functionality.

Mirtes is a real-time middleware providing a database-like abstraction of the WSN, thus allowing the user to retrieve sensor readings by spawning SQL-like queries. This kind of abstraction has two main advantages: it allows to easily interface the WSN with users and other systems, and it permits to support applications from different domains. The virtual database provided by Mirtes has a table for each physical quantity measured through the WSN: scalar quantities have (at least) three columns, two for the node ID and the time-reference associated to the measurements, and the other for its value; vector quantities have (at least) as many columns as their components, plus the node ID and time reference columns. The time reference associated to the measurements is taken from the MAC layer (see below). The information set required to fully describe the tables is called database schema.

Mirtes supports periodic queries having timeliness constraints. The system guarantees that, using periodic queries, the environment is periodically sampled with no jitter and that the sampling is synchronously undertaken in the WSN, i.e., all sensor nodes acquire their measurements exactly at the same time and specifically at the beginning of the periodic interval. As shown in Fig. 2, the transmission of the sampled data to the sink node can be delayed, but Mirtes ensures that, in any case, the query result is produced before the next activation. This behavior is especially suited for time-discrete control applications: using

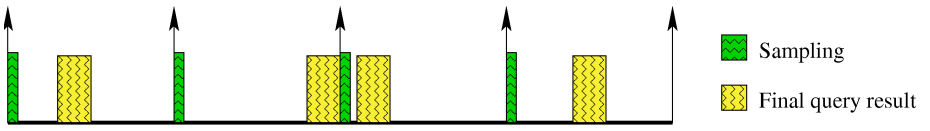


Fig. 2. Example of jitter-free environment sampling by means of MiRTES periodic queries: the sensing is always performed at the beginning of the period, whereas the transmission of the acquired data can be delayed as long as it does not miss its deadline.

MiRTES periodic queries, the system output is measured exactly at time instant t_k and the measured value is available to the controller before instant t_{k+1}. If the user spawns a new periodic query, the system performs an acceptance test (admission control), which guarantees that the new instance does not disrupt the timeliness of the old set while fulfilling that of the new query.

MiRTES real-time features relay on a communication strategy based on the IEEE 802.15.4 standard. Specifically, MiRTES works with beacon-enabled WPANs supporting the Guaranteed Time Slot (GTS) feature. MiRTES sends the query in the beacon payload and allocates a GTS to each node that can potentially reply with its readings. This approach allows *i*) to know *a priori* the exact query execution time; *ii*) to schedule queries using the well-known EDF algorithm; and *iii*) to use the simple schedulability test based on utilization. The beacon is also used for time synchronization between nodes. The time is expressed in beacon intervals. In case of periodic queries, the jitter-free sampling of the environment is achieved synchronizing the sensing period with the query period by adding additional information (i.e., period and activation offset) to the query request. Sensor nodes always acquire measurements at the beginning of the period and keep them in memory until the transmission request is received.

MiRTES dependency on beacon-mode and GTS features currently constraints the WSN to a star topology, in which the sink node is the central node, called coordinator. We are well aware of the limitations posed by such topology, and in Section 6 we propose alternative solutions for the monitoring module. Nevertheless, a star topology is sufficient for our prototype and the beacon-enabled network also provides some benefits to the power-saving policy and reliability mechanism. Specifically, the superframe structure used in a beacon-enabled IEEE 802.15.4 network can contain an inactive part during which nodes can enter a power-save mode. If the node runs a vision algorithm requiring a continuous monitoring of the scene (e.g., the traffic flow sensor described afterward), it can just switch off its radio. Otherwise, if the vision algorithm can run sporadically (as it happens for the parking space occupancy sensor, also described later), the node can even enter the sleep mode. That is possible because MiRTES synchronizes the sensing activity with the network communication and runs the algorithm in the active part of the superframe. The sensor duty cycle is application specific and can easily be changed, even dynamically, by modifying the corresponding network parameters, i.e., the Beacon Order (BO) and the Superframe Order (SO). Specifically SO defines the duration of the active part, and

must be chosen large enough to accommodate the sensor responses to the query. BO defines the beacon interval length and it must be chosen greater than or equal to SO. Once SO is fixed, the inactive part can be introduced/extended only increasing the beacon interval. Therefore the BO value is a trade-off between the power consumption and the minimum monitoring period achievable.

To cope with the communication issues posed by real-world scenarios, we extended Mirtes with mechanisms aimed at increasing the communication reliability. We implemented a very simple Forward Error Correction (FEC) mechanism to reduce the Packet Error Rate (PER). We replicate the message payload (adding a 2-byte CRC-code for each replica) to increase the probability that at least one copy will be correctly received. For most of the messages, i.e., all the messages from sensors to the coordinator, we do not need to worry about the corruption of the packet header: since they are sent using GTSs, the coordinator (which allocates GTSs) already knows to be the recipient and which node is the sender.

3.2 Code-Update Module

In complex IT systems code management is an essential service: as system requirements and environments change over time users may need to introduce new functionalities. For instance, in our case, the embedded image processing algorithm may be subject to periodic revisions and improvements. In a WSN scenario the need for remote software updates is driven both by the scale of deployment, and the potential inaccessibility of sensor nodes. Moreover, since sensor networks must often operate unattended for long periods of time, the system must be reliably upgradeable without a physical intervention. The main requirements of this service are: *i*) dynamic updates of the entire code on a set of nodes; *ii*) robustness and integrity. The possibility to change every part of the code allows the users to upgrade not only parts of the application layer, but also the lower levels, such as operating system or network stack. Robustness is critical as the ability to download new software into a node introduces a whole set of failure modes. This requirement implies the need to handle failures (or corruptions) during the update process. In our system a separate copy of the upgraded firmware is downloaded to the node in such a way that the original firmware is preserved. If anything goes wrong at any time during the upgrade process, the original firmware can be restored. Reliability is guaranteed through the use of integrity check techniques on every fragment of the firmware. Administrators can perform firmware upgrade through a remote interface, which permits to select the target node and the new firmware to upload. The network coordinator is in charge of distributing firmware fragments to the device. Nodes host a wireless boot-loader to receive the fragments, verify their correctness and store them in non-volatile memory.

3.3 Configuration Module

The ability to remotely view images captured from camera sensors is essential in visual WSN context. The access to the image is needed in system setup, configuration, and maintenance phases to: *i*) check the proper functioning of the device; *ii*) properly orient the lens and capture the desired scene; *iii*) select the regions of interest in the scene. Image transfer over constrained wireless channel is a heavy task from the communication point of view. Since image sizes are relatively large compared to the MAC frame size, a fragmentation is needed to perform data transfer. In ScanTraffic we implemented a simple Stop and Wait protocol between the network coordinator and the devices. Data integrity is not checked since image corruption, due to communication errors, is generally tolerated.

Fig. 3. Snapshot of the graphical user interface for setting ROIs and other parameters for a parking sensor. In this case, each ROI defines the part of the image associated with a specific parking space ID.

Every smart camera frames a different scene. A static configuration in setup phase leads to problems of scalability and do not respond to possible environmental changes over time. Remote configuration allows administrators to set key parameters for the image processing algorithm running on each device. In ScanTraffic the following parameters can be set: the number of the regions of interest (ROI) to be monitored, the coordinates of the points defining such regions, and identifiers used to tag the collected data (e.g., the ID of a parking space). Other parameters may be set, e.g., to tune the algorithm sensitivity in particular light conditions. ROI configuration is performed by administrators through a remote graphical interface allowing to draw ROIs directly on the image received by the sensor (see Fig. 3). Parameters are saved in non-volatile memory, therefore no reconfiguration is needed when batteries are changed.

Fig. 4. A view of the Pisa International Airport land-side. We deployed flow sensors in the main intersections, and parking sensors in both the outdoor parking lot (on the right) and the indoor parking lot (on the left).

3.4 Communication Manager

The three service modules have different communication requirements. Specifically, both the remote configuration and the code-update services require high bandwidth (for transferring images and firmware, respectively), whereas the monitoring service requires timeliness and power management. The Communication Manager is in charge of handling this conflicting situation.

In ScanTraffic we defined two possible communication settings, corresponding to two possible states for the WSN: the running state and the maintenance state. During the running state, only the monitoring module is active and the WSN is configured to use GTSs and a superframe with inactive part. That permits the timely and power-aware communication described in Section 3.1. During the maintenance state the monitoring service is suspended and the WSN uses a superframe without inactive part, i.e., nodes are always active and they can send and receive packets continuously. Therefore, the remote configuration and code-update modules can quickly transfer large amounts of data (i.e., images and firmware).

We also designed a third state, called the off-duty state. It is used to temporarily "switch-off" the WSN when the ITS does not need its services (for example during the night). In this state all nodes are in sleep mode. The switch-off signal must specify the time length of the off-duty interval. Once this time elapses, the nodes automatically wake up. When specifying the time length, administrators should take into account device clock deviation: a lower value should be chosen if late wake-ups must be avoided. The resulting final state machine (FSM) is depicted in Fig. 1b. State transitions are caused by control messages sent from upper layers to the coordinator.

4 Implementation and Deployment

We implemented a prototype of the system infrastructure proposed by Ipermob and deployed it in the land-side of the Pisa International Airport (depicted in

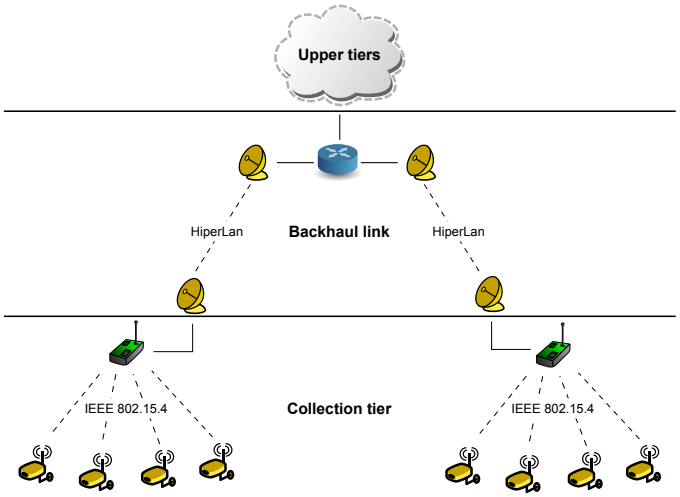


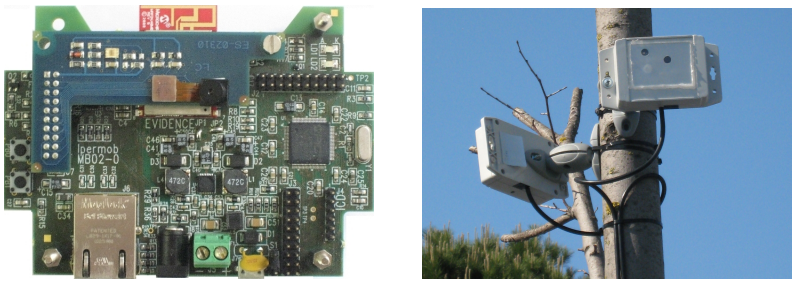
Fig. 5. Network architecture of the IPERMOB prototype: backhaul links connect WSN coordinators to upper tiers.

Fig. 4). The prototype serves as a proof-of-concept for the entire system, including visual WSNs managed by SCANTRAFFIC. The goal of such visual WSNs is to collect information about parking lot occupancy and traffic flow. For this purpose, we used the two embedded vision algorithms described in [10]: one algorithm counts cars passing in a road section; the other algorithm detects the occupancy status of a set of parking spaces. For the sake of brevity, we use the name "flow sensor" to denote a smart camera running the former algorithm and the name "parking sensor" to denote a smart camera running the latter. As described in [10], those algorithms achieve an overall detection rate of 95% with a false alarm rate of 0.1%.

Each visual WSN is connected to the rest of the system via a backhaul link. A special node in the WSN, the coordinator, acts as a gateway between the WSN and the upper layers. In the IPERMOB prototype the backhaul link is a HiperLAN link (see Fig. 5) and the coordinator uses the UDP/IP protocol to communicate with upper tiers.

4.1 Hardware Description

The smart cameras used in the prototype were designed within the IPERMOB project. Specifically, two different boards were developed. The first board (Fig. 6) is designed to be particularly low-cost and low-power. It is equipped with an 80 MHz PIC32 microcontroller with built-in 128 KB of RAM and 512 KB of ROM. It has no extra hardware for driving the camera (e.g., a frame buffer), therefore it cannot achieve a high frame rate. Nevertheless, it is still suitable to act as a parking sensor, because, in such application, the inter-arrival time of the target events (i.e., cars occupying or leaving a parking space) is expected to



(a) (b)

Fig. 6. The microcontroller-based smart camera used in the Ipermob project. The board (a) is equipped with a Pic32 microcontroller, an IEEE 802.15.4 transceiver, a CMOS camera, and an Ethernet port used for debugging purposes. Since the device is battery-powered and communicates wirelessly, it can be easily installed almost everywhere (b).

be high. In the development of the second board, power constraints have been slightly relaxed in order to explore a solution with more flexibility and processing capabilities. The second board is an FPGA-based device equipped with 32 MB of RAM. The FPGA hosts a soft-core microprocessor, thus allowing to reuse the code developed for the other board. The FPGA also drives the camera to its maximum frame (i.e., 30 fps), although the soft-core runs at only 40 MHz. This feature, together with a relatively high amount of RAM, makes the board suitable for monitoring traffic flow.

4.2 Software Platform

Each node runs Erika [1], a real-time multi-tasking operating system especially designed for time-constrained embedded applications. Erika's priority-based scheduling allows to run the computationally intensive vision algorithms concurrently with ScanTraffic without compromising its functionality. Moreover, Erika's implementation of the IEEE 802.15.4 standard supports the beacon-mode and the GTS feature required by the monitoring module of ScanTraffic, i.e., Mirtes. Support for flow and parking sensors was easily added to the monitoring module by customizing the Mirtes virtual tables (cf. 3.1) with the two sensor types.

ScanTraffic memory requirements for each possible combination of hardware platform and node type are shown in Table 1. Sensor node RAM requirements have a variable component depending on the chosen image resolution, e.g., in case of 160×120 frames, 19 KB of additional RAM is accounted.

4.3 Testbed Planning and Implementation

The deployment of the system in a complex urban scenario required an accurate planning, with the following objectives: *i*) minimize the number of sensing

Table 1. Memory requirements for each possible combination of hardware platform and node type.

Device	Node Type	RAM [KB]	ROM [KB]
PIC32	Coordinator	26.7	150.1
PIC32	Parking Sensor	24.5 + Image Size	111.6
FPGA	Flow Sensor	16.0 + Image Size	80.0

devices, keeping the number of monitored parking spaces as large as possible, *ii*) minimize the number of WSN coordinators, i.e., nodes with broadband connection to the control center; *iii*) reducing installation time; *iv*) enhancing the reuse of existing poles and supports; *v*) maximizing system performance, carefully choosing the positions of the sensors, avoiding non-line-of-sight communication. We deployed a total of 21 smart cameras, specifically: 14 parking sensors monitoring 83 parking spaces, and 7 flow sensors monitoring 8 traffic lanes.

The deployment of smart cameras was greatly simplified and sped up by the availability of remote configuration and code-update. The actual deployment (i.e., excluding the planning phase) was carried out by two people and it took less than three days. Most of the time was spent to physical install the sensors to pre-existing poles or trees. The configuration phase (including camera orientation) took just half a day. We programmed every smart camera with a special deployment firmware which constantly acquires the current camera view and sends it to a laptop connected to the Ethernet interface, thus allowing to properly orient the camera to capture the desired scene. Once the installation was completed, we remotely uploaded the operational firmware (flow or parking sensor) to each smart camera and we configured the algorithm using the remote configuration service. We could have used the configuration service also to remotely acquire the current camera view, thus avoiding the need for the procedure just described. However, this approach would actually have been slower, because the limited bandwidth does not permit a fast image retrieval, thus considerably slowing down the camera orientation setup.

5 Validation

The IPERMOB prototype served as a test-bed to validate SCANTRAFFIC design and implementation. SCANTRAFFIC real-time features (inherited from MIRTES) proved to be essential for the correctness of the information produced by the WSNs. Jitter-free periodic queries guarantee that the number of counted vehicles is related to a known time interval equal to the query period (tunable at runtime). It is therefore possible to precisely compute the traffic flow with a specific time resolution. The synchronous sensing enforced by SCANTRAFFIC permits to build a time-coherent status of the whole parking lot from the single parking spaces.

Table 2. Power consumption of both devices for different configurations.

Device	Node Type	Image Size	Frame Rate [fps]	$\mathbf{P}_{\max}$ [mW]	$\mathbf{P}_{\text{idle}}$ [mW]
Pic32	Parking Sensor	160×120	1	450	6
		320×240	1	462	6
FPGA	Flow Sensor	160×120	30	1230	25
		320×240	20	1230	25

The prototype deployment showed that visual WSNs running ScanTraffic can be quickly installed in a urban scenario reusing existing poles and supports, without the need for costly and time-consuming engineering works. Indeed, one unique characteristic of ScanTraffic visual WSNs is that they can be easily deployed on-request where needed, and subsequently removed. Such temporary installation can be used to provide ITS services during big events (e.g., concerts, festivals, etc.) or to collect data for planning infrastructure improvements.

5.1 Power Consumption

Currently Erika does not support the idle state for our boards and we did not implement it, since Ipermob main goal is to prove the feasibility of using visual WSNs in ITSs, taking energy related issues for a possible follow-up. However, we measured the power consumption of each board in idle state (P_{idle}) and at full load (P_{max}), i.e., with the radio transmitting and both the camera and the algorithm running. Results are shown in Table 2. For the parking lot monitoring application, it is reasonable to assume that, with a monitoring period of one minute, the Pic32-based board powered by 4 AA batteries, i.e., with a total voltage (V_{bat}) of 6 V, will last for about 3 weeks. Indeed, the camera and the algorithm can run just once a minute, therefore the board will be on for less than 2 seconds every minute (considering a frame rate of 1 fps, the start-up time, and the computation time). If we use a high beacon order, e.g., 10, and a low superframe order, e.g., 4, the radio is on for just 1 second every minute. The resulting duty cycle (α) is equal to 5% and, supposing a battery capacity (C_{bat}) of 2500 mAh the node life is:

$$\frac{C_{bat} \cdot V_{bat}}{\alpha \cdot P_{max} + (1-\alpha) \cdot P_{idle}} = \frac{2500 \cdot 6}{.05 \cdot 462 + .95 \cdot 6} \left[\frac{\text{mAh} \cdot \text{V}}{\text{mW}}\right] \simeq 520.8\,[\text{h}] \qquad (1)$$

On the contrary, the FPGA board, used for the vehicular flow monitoring, will last for just 11 hours. The reason is not only the higher power consumption, but primarily the duty cycle of 100% required by the flow monitoring. Indeed, entering the sleep mode would cause to miss some vehicles.

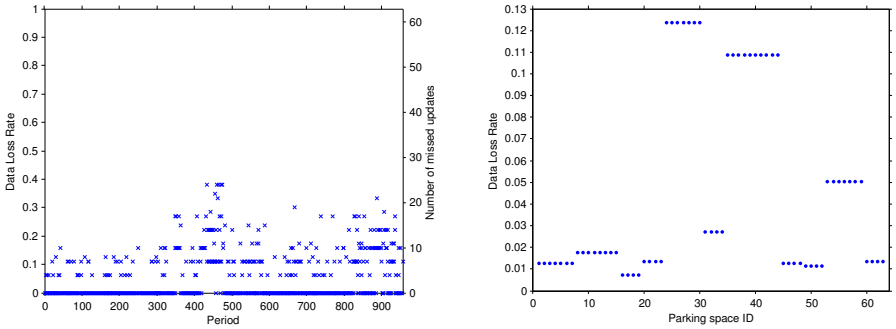


Fig. 7. ScanTraffic prototype data transmission reliability: (a) evolution of the data loss rate during the 5-hour-long validation experiment, and (b) average data loss rate per parking space.

5.2 Data Transmission Reliability

From a communication point of view, ScanTraffic must operate in a harsh environment. In public places as the Pisa International Airport the ISM 2.4 GHz band is much crowded. Moreover cars, as well as other metal objects, highly reflect electromagnetic waves and produce multipath interference whose modeling is difficult because of the rapid changes in the scenario. The adopted data protection can reduce the probability of data loss, but cannot completely avoid it. However, as long as data loss is moderate, it is not a problem, because Ipermob target applications can tolerate missing data. Indeed, their metrics of interest are aggregate values.

To evaluate the data transmission reliability of the deployed prototype, we set a monitoring period of 20 seconds and, for every period, we counted the number of missing parking space and flow updates. The experiment lasted for more than 5 hours, i.e., for almost 1000 periods. Fig. 7 shows the results of such experiment performed on the biggest visual WSN deployed within the prototype, having 11 sensors monitoring a total of 63 parking spaces. Therefore, in each period, 63 updates were expected. Fig. 7a shows the percentage of missed updates for every period. On average, just 4.5% of the data is lost (i.e., the occupancy status of 2-3 parking spaces is not updated). Fig. 7b shows the loss rate per parking space. Parking spaces monitored by the same sensor have the same loss rate, because the sensor groups all the updates together in one message. For most of the sensors only 2-3% of the messages are lost, with the exception of 3 sensors whose loss rate is up to 12%. Those sensors are responsible for the data loss peaks shown in Fig. 7a. Indeed they are the farthest from the coordinator and therefore their link quality can easily drop under the minimum acceptable level. The easiest solution to fix this problem is to add another coordinator nearer to them, thus creating a new WSN. Indeed, the considered area, spanning for more than $3\,000\,\mathrm{m}^2$, is probably too large to be covered by just one coordinator.

6 Conclusions and Future Work

In this paper we presented ScanTraffic, a system to collect traffic information using visual WSNs. Specifically the implemented prototype acquires information about parking space occupancy status and vehicle flow. We claim that, at least for parking monitoring, the proposed system is better then existing solutions based on scalar sensors, specifically inductive sensors. First of all, as discussed in Section 5, our system can be used for temporary installations thus addressing new application scenarios such as surveys for infrastructure planning and ITS support for special events. Our system is interesting for permanent installations as well. Indeed, the battery depletion problem (discussed in Section 5.1) can be solved using small solar panels (in our calculations an area of $0.2\,m^2$ is sufficient) and still the total cost of the system will be competitive with that of solutions based on inductive sensors (which are cheaper per se, but require expensive engineering works to be embedded in the asphalt). Finally, our system has greater potentiality than existing solutions. Thanks to the smart camera versatility we discussed in Section 1, our system can support new functions (e.g., vehicle classification) as well as new applications (e.g., surveillance), without any physical intervention.

ScanTraffic shows how a proper network/software design is necessary to fully exploit smart cameras features. We built ScanTraffic starting from two previous works of ours, i.e., Mirtes and the embedded vision algorithms. We successfully integrated the computation-intensive vision algorithm with Mirtes without compromising its real-time features and we added error correction techniques to Mirtes in order to address the communication issues posed by real-world scenarios. However we quickly realized that to achieve the aforementioned advantages over traditional scalar WSNs, additional services were needed, namely remote configuration and code-update. The resulting system has been described in the paper, proving that it is suitable for real-world deployment.

We believe that our proposed architecture is general enough to be reused in other applications, not necessarily related to the ITS domain. To this end, we are considering alternative communication patterns addressing different application requirements and overcoming the star topology limitation posed by Mirtes. We have already implemented [8] a set of protocols following the web service approach on embedded systems. This solution relies on the 6LoWPAN standards and can support complex topologies like the mesh. Control over the sensing activity and access to the collected data are possible through a REST interface provided by every sensor node in the network. We use the Constrained Application Protocol (CoAP) to implement such RESTful environment. The major drawback of this solution is the (current) lack of real-time support, making it unsuitable for real-time applications. Therefore we are also considering another solution based on cluster-tree topology. We plan to use one of the approaches proposed in literature [14,9] to implement a beacon-enabled cluster-tree network, and to extend Mirtes to support it. Compared to the previous solution, this implementation will be more complex and less robust, but it will support real-time applications.

Acknowledgments We would like to thank our shepherd Tommaso Melodia and the anonymous reviewers for their valuable feedback that helped us to improve this paper.

References

1. Erika Enterprise RTOS. `http://erika.tuxfamily.org/`
2. Alessandrelli, D., Pagano, P., Nastasi, C., Petracca, M., Dragoni, A.F.: Mirtes: middleware for real-time transactions in embedded systems. In: 3rd IEEE International Conference on Human System Interactions (HSI). pp. 586–593 (2010)
3. Arief, B., von Arnim, A.: TRACKSS approach to improving road safety through sensors collaboration on vehicle and in infrastructure. In: 68th IEEE Vehicular Technology Conference (VTC). pp. 1–5 (2008)
4. Barbagli, B., Bencini, L., Magrini, I., Manes, G., Manes, A.: An End To End WSN Based System For Real-Time Traffic Monitoring. In: 8th European Conference on Wireless Sensor Networks (EWSN) (2011)
5. Campbell, J., Gibbons, P.B., Nath, S., Pillai, P., Seshan, S., Sukthankar, R.: Irisnet: an internet-scale architecture for multimedia sensors. In: 13th annual ACM international conference on Multimedia. pp. 81–88 (2005)
6. Ceriotti, M., Corrà, M., Orazio, L.D., Doriguzzi, R., Facchin, D., Jesi, G.P., Lo Cigno, R., Mottola, L., Murphy, A.L., Pescalli, M., Picco, G.P., Pregnolato, D., Torghele, C.: Is There Light at the Ends of the Tunnel? Wireless Sensor Networks for Adaptive Lighting in Road Tunnels. In: 10th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN/SPOTS). pp. 187–198 (2011)
7. Chen, W., Chen, L., Chen, Z., Tu, S.: WITS: A wireless sensor network for intelligent transportation system. In: 1st International Multi-Symposiums on Computer and Computational Sciences (IMSCCS) (2006)
8. Gutierrez Mlot, E.D., Bocchino, S., Azzarà, A., Petracca, M., Pagano, P.: Web services transactions in 6LoWPAN networks. In: 12th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM) (2011)
9. Koubâa, A., Cunha, A., Alves, M., Tovar, E.: TDBS: a time division beacon scheduling mechanism for zigbee cluster-tree wireless sensor networks. Real-Time Syst. 40, 321–354 (2008)
10. Magrini, M., Moroni, D., Nastasi, C., Pagano, P., Petracca, M., Pieri, G., Salvadori, C., Salvetti, O.: Visual sensor networks for infomobility. Pattern Recognition and Image Analysis 21, 20–29 (2011)
11. Mathur, S., Jin, T., Kasturirangan, N., Chandrasekaran, J., Xue, W., Gruteser, M., Trappe, W.: Parknet: drive-by sensing of road-side parking statistics. In: 8th International Conference on Mobile Systems, Applications, and Services (MobiSys). pp. 123–136 (2010)
12. Soro, S., Heinzelman, W.: A Survey of Visual Sensor Networks. Advances in Multimedia 2009, 1–22 (2009)
13. Thiagarajan, A., Ravindranath, L., LaCurts, K., Madden, S., Balakrishnan, H., Toledo, S., Eriksson, J.: VTrack: accurate, energy-aware road traffic delay estimation using mobile phones. In: 7th ACM Conference on Embedded Networked Sensor Systems (SenSys). pp. 85–98 (2009)
14. Toscano, E., Lo Bello, L.: A multichannel approach to avoid beacon collisions in IEEE 802.15.4 cluster-tree industrial networks. In: 14th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA) (2009)