Chapter 2
Deployment and Management of Cooperating Objects

2.1 Overview

Despite the vast amount of past and on-going research in network embedded systems and pervasive computing, real-world deployments of systems of Cooperating Objects are largely still limited to research prototypes. Managing, controlling, and verifying the cooperation and coordination among heterogeneous objects indeed represents a major challenge when the system is deployed in the field. In this respect, Cooperating Objects unfortunately inherit issues germane to some of their constituent technologies, e.g., the lack of visibility into the operation of sensor network systems. This is caused by some of their distinctive characteristics:

- Cooperating Objects are deeply embedded within the real world to perceive and control the environment through sensors and actuators. The dynamics of real-world environments negatively affect the system operation, e.g., because of the unpredictable behaviour of the wireless medium when radio communication is used;
- Cooperating Objects are required to go far beyond the simple interactions found in early deployments of the technologies they build upon. For instance, unlike most sensor network deployments that essentially revolve around pure data collection, cooperating objects are required to form highly dynamic distributed systems with complex interactions;
- Cooperating Objects are often subject to severe resource constraints, e.g., in terms of computation, communication, and available energy. Constrained resources complicate the programming activity, leading to error-prone software, and make it difficult to identify and remedy the causes of such failures.

Contributors of this chapter include: Nils Aschenbruck, Jan Bauer, Armando Walter Colombo, Christoph Fuchs, Philipp Maria Glatz, Stamatis Karnouskos, Paulo Leitão, Marco Mendes, Luca Mottola, Amy L. Murphy, Gian Pietro Picco, and Thiemo Voigt.

S. Karnouskos et al., *Applications and Markets for Cooperating Objects*,
SpringerBriefs in Cooperating Objects,
DOI: 10.1007/978-3-642-45401-1_2, © The Author(s) 2014

The characteristics and issues above entail that the development of systems of Cooperating Objects, their deployment in real-world settings, as well as their management during the system lifetime represent challenging tasks. First steps have been undertaken to address some of these challenges [1], yet these are typically isolated ad-hoc approaches that target only one specific facet of Cooperating Objects. As the current practice is not sustainable in the long-term and already often proved to be insufficient even for small-scale deployments, a widespread adoption of cooperating objects requires a more systematic approach to their deployment and management. Most importantly, not just one specific technology needs to be taken into account, but the focus needs to progressively shift towards the cooperation of heterogeneous platforms.

In this chapter, we discuss how the above issues are being tackled in the deployment and management of real-world systems of Cooperating Objects. We do so by touching upon diverse application scenarios and requirements, cast in a number of real settings:

- Section 2.2 illustrates efforts in employing Cooperating Objects for monitoring railway bridges, pointing out the challenges in data fidelity and distributed processing that need to be overcome for these systems to be practically effective;
- The application of Cooperating Objects in industry automation systems is the subject of Sect. 2.3, where the use of service-oriented architectures is suggested as a way to overcome real-world integration issues;
- Deployment of Cooperating Objects in harsh settings is discussed in Sect. 2.4 for a case of light-weight bird tracking, where weight of the hardware platform and connectivity issues represent the major obstacles to overcome;
- Section 2.5 reports on the use of Cooperating Objects in public safety scenarios, involving diverse computing platforms with distinctly different capabilities and the additional complexity due to mobile settings;
- Finally, Sect. 2.6 illustrates deployments of Cooperating Objects in operational road tunnels, highlighting the challenges stemming from their integration in control systems and with industry strength equipment.

Overall, the rest of the chapter exemplifies the issues at stake in deploying and managing systems of Cooperating Objects. Remarkably, practical and effective solutions in these areas are key requirements for eventual market adoption.

2.2 Monitoring Railway Bridges

2.2.1 Overview

An area where Cooperating Object technology holds great potential is the monitoring of civil structures. A challenging scenario in this domain is given by existing bridges. Particularly, we investigated the application of Cooperating Object technology for monitoring railway bridges in Stockholm, Sweden.

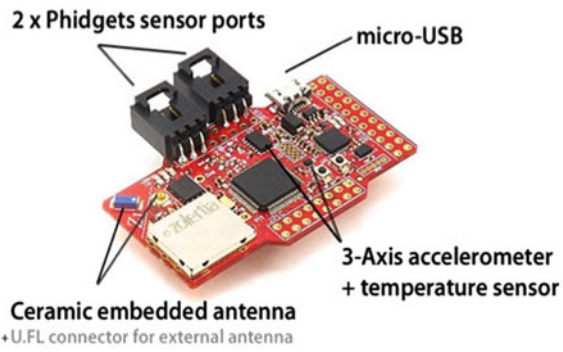

Fig. 2.1 Zolertia Z1 node

The gradual deterioration and failure of existing structures indeed requires the need for Structural Health Monitoring (SHM) systems to develop a means to monitor the health of structures. Dozens of sensing, processing and monitoring mechanisms have been implemented and widely deployed with wired sensors. On the other hand, the complexity and high installation costs of traditional wired SHM systems have posed the need for replacement with more flexible technology, such as Cooperating Objects.

To counteract memory and energy limitations, thus prolonging the lifetime of battery-operated systems, we designed low-power and memory efficient data processing algorithms. We used in-place radix-2 integer Fast Fourier Transform (FFT). Our implementation increases the memory efficiency by more than 40 % and saves processor power consumption over the traditional floating-point implementations.

A standard-deviation-based peak picking algorithm is next applied to measure the natural frequency of the structure. The algorithms together with Contiki, a lightweight open source operating system for networked embedded systems, are loaded on Z1 Zolertia sensor nodes, shown in Fig. 2.1. Analogue Device's ADXL345 digital accelerometers are used to collect vibration data, to validate the algorithms using supported beam structures.

2.2.2 Application Description/Usage Scenarios

The process of implementing a damage characterization and detection method for engineering structures is referred to as SHM. Although it had been quite a while since the science of SHM was introduced, its use was confined to mechanical structures like airplanes, ships, and machinery. It had never been applied to civil engineering structures until its significance was noticed in the frequent deterioration and collapse of large and prestigious structures. These issues emerge in particular for bridges, whose possible failure may have significant costs, both in economical and social terms.

Fig. 2.2 The catastrophic failure of I-35W Bridge in Minneapolis, Minnesota after collapse on August 1, 2007 (*left*); the Point Pleasant Bridge collapse (*right*)

For example, the catastrophic failure of the I-35W Bridge in Minneapolis, Minnesota (Fig. 2.2 left) and of the Point Pleasant Bridge (Fig. 2.2 right) were among the episodes that alerted the need to devise some means to tell the status of structures before anything could happen. Consequently, a continuous health monitoring of structures is important and a mechanism should be developed by which efficient and accurate information could be obtained.

Researchers, hence, gave special attention to this discipline and proposed their own customized solutions in the last couple of decades, which eventually gave birth to the science of SHM. SHM is thus one of the multidisciplinary fields that integrates the contribution of researchers from mechanical, electrical, civil and architecture engineering. Due to the easy access, the wide availability and reliability of wired systems, many solutions have been implemented using wired sensor networks. However, high installation cost, the need for specially trained professionals for set up and maintenance, and their bulky nature made the research community to divert its attention towards more flexible technologies, of which Cooperating Objects are an example.

Nevertheless, systems of Cooperating Objects deployed for SHM also present significant technical challenges. For example, it is essential to improve the energy efficiency as the energy budget is usually extremely limited, and yet sensed data in SHM applications comes in high volumes that are expensive to transmit wirelessly. On the other hand, memory and computing limitations also reduce the nature and amount of local processing that can be possibly performed aboard the devices to save on wireless transmissions.

Based on the above considerations, we aimed at understanding to what extent existing data processing algorithm can run on Cooperating Object devices in the face of computing and memory limitations, studying the trade-off in terms of quality of the output versus resource consumption. We then designed and implemented customized algorithms to better fit the characteristic constraints of Cooperating Object devices.

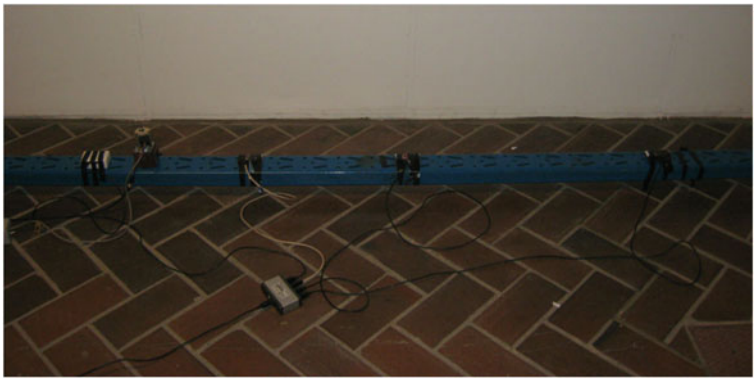

Fig. 2.3 Main section of the test beam, with wired monitoring system attached

Fig. 2.4 Wooden support of the test beam. A Zolertia Z1 node is also visible, attached at the end of the beam

2.2.3 Key Results and Lessons Learned

Our customized FFT and peak-picking algorithm implementations serve as a foundation to get the study of more complex algorithms started on a sound basis. To that end, we validated the performance of our implementations against a wired vibration monitoring system deployed in a university engineering lab.

We used a simple steel beam, shown in Fig. 2.3, supported by wooden blocks at the two ends. The length of the beam was 3.5 m and the wooden supports, shown in Fig. 2.4 at the end points add a total of 18 cm. The whole span was further divided into five sub-parts of each 66.4 cm long. By placing our sensor nodes on these sub-parts, we collected measurements for real-time and offline analysis.

For the wired system, HBM MGC Plus data acquisition system (DAQ) was used. The system includes a Si-FlexTM MEMS sensor to be firmly attached to the beam with a heavy electromagnet attachment, a multichannel ADC and a Catman DAQ software installed on a laptop computer.

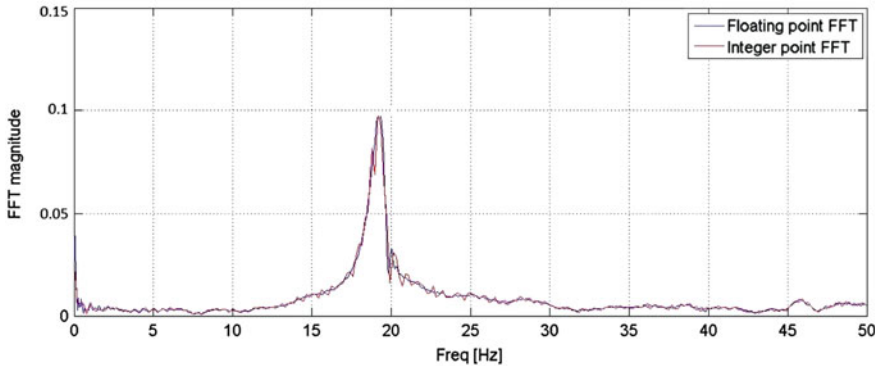

Fig. 2.5 Comparison between Matlab's floating-point FFT and FFT custom implementation for Zolertia Z1

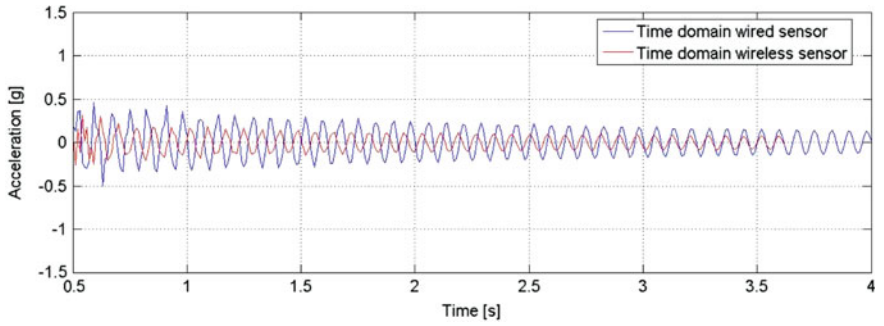

Fig. 2.6 Time domain data from wired and wireless sensors

After collecting time domain data with the wireless sensors, we apply our FFT implementation and compare the result with the floating-point FFT implementation in Matlab. Figure 2.5 shows the two next to each other. Our implementation gives a good approximation of the floating-point FFT on top of saving memory space and reducing energy consumption. Given the low resolution data from the ADXL35 accelerometer, the most important thing is noting the existence of the resonant peak frequencies. These points are what domain experts need to know to extract the important information to feature the behaviour of a bridge.

On the other hand, a time domain plot of the data from the wired and wireless systems is given in Fig. 2.6. As seen from the figure, the time domain data plot from the wired system is relatively of better quality than the one from the wireless system, although the data obtained from the wired and wireless look similar to some extent, which indicates that both are measuring same vibration data from the bridge.

A closer look into the measurements exposes some of the flaws in the wireless sensing system. At low amplitudes, the wireless system introduces noise due to the low resolution (10 bits per sample) of the ADC used in contrast with the 24-bits high

resolution one used in the wired system. Here, the 14-bits difference in the fidelity of both data is considered as one of the trade-offs in using wireless sensor system. More specifically, the difference for the signal quality can be attributed to two factors:

- *Noise level*: the Si-Flex accelerometer has lower noise level (300 ngrms/Hz) than that of the ADXL345 accelerometer ($\leq$1.5LSB rms); hence less amount of noise is introduced into the measurements of the wired system.
- *ADC resolution*: the MGC plus DAQ system has a 24-bit ADC which is of much higher resolution than the ADC found in the ADXL345 accelerometer (10 bits of resolution for a g-range of $\pm$2 g); the more the bit resolution of the ADC, the more the quantization level of the converter and eventually, this yields digital samples of higher fidelity.

In summary, we argue that while the software side may already be ready for real-world deployments, on the hardware side we still require better fidelity sensing devices able to get closer to the quality of mainstream wired systems. The provision of such hardware may open market opportunities for this domain, which would instead remain untapped in the current situation.

2.3 Cooperative Industrial Automation Systems

2.3.1 Overview

The future factory is a complex system of systems, where sophisticated and dynamic systems interact with each other in order to achieve the goals at system-wide but also at local level. To realize this, timely monitoring and control as well as open communication and collaboration in a cross-layer fashion are key issues. Modern approaches such as the service-oriented architecture (SOA) paradigm when applied holistically can lead to the desired result [2].

Promising futuristic approaches followed within the EU research projects SOCRADES (http://www.socrades.eu) and IMC-AESOP (http://www.imc-aesop.eu) adopt the "collaborative automation" paradigm where the aim is to develop tools and methods to achieve flexible, reconfigurable, scalable, interoperable network-enabled collaboration between decentralised and distributed embedded systems (Cooperating Objects). In particular, the SOCRADES technical approach [3–6] realized a service-oriented ecosystem, where networked systems are composed by smart embedded devices interacting with both physical and organizational environment, pursuing well-defined system goals. IMC-AESOP empowered by the advanced of cutting edge technologies and concepts [7], pushes the interaction and collaboration capabilities of Cooperating Objects even further by providing an insight how the future industrial automation systems [8] would interact and how their applications would benefit.

In a service-oriented industrial enterprise, the communication and coordination of activities is done by the engineering associated to the service requesting and offer. Cooperating Objects that are integrated in this environment can make usage of the service-oriented mechanism as a mean for cooperation between different autonomous parts of the system. The current application illustrates the deployment and management of service-enabled Cooperating Objects of a pallet-based assembly system (horizontal cooperation), as well as their integration into the production and enterprise resource levels (vertical cooperation). A major benefit of this approach is the modular system deployment and the usage of service-orientation to establish cooperative acts, as well as the integration of heterogeneous resources and information coming from different layers of the enterprise.

2.3.2 Application Description/Usage Scenarios

The factory of the future will depend on the services for realizing sophisticated functionalities [2, 5]. Services are basis of a mechanism by which needs and capabilities are brought together, and is a promising way of enabling interoperable interactions among the different cooperating entities. For Cooperating Objects, the principle of service-orientation can be seen as a mean for realizing cooperation. A Cooperating Object represents its actions and resources as a set of services that can be used by other parties e.g., other Cooperating Objects.

As an example, a service-enabled Cooperating Object (as depicted in Figs. 2.7 and 2.8) could be a mediator of a conveyor segment; hence it has the ability to read the sensors and control the actuators of the conveyor, to make it possible to transport pallets from its input to its output. This forms the internal objective of the Cooperating Object, but as it operates in a wider context it has also to respect external/global objectives of the system. The objective and available condition can be offered as a service to the outside (service: transport pallets), so that possible another entity (e.g., a pallet) could request it e.g., *"Please transport me from point A to point B"*. However to complete the service and also to respect global system objectives, the conveyor must interact with the availability service from the next transport unit or workstation connected to its output. This can be seen as the form of collaboration and automatic rearrangement of services in this system.

The approach for creating complex, flexible and reconfigurable production systems is that these systems are composed of modular, reusable entities that expose their production capabilities as a set of services. This composition approach applies to most levels of the factory floor; simple devices compose complex devices or machines, which in turn are composed to build cells or lines of a production system and so on. The same applies to concept of service-oriented production systems and composing complex services from simpler services.

The application scenario that is realized to demonstrate the integration of service-enabled Cooperating Objects is based on a customized Prodatec/FlexLink DAS 30—Dynamic Assembly System. The DAS 30 system is a modular factory

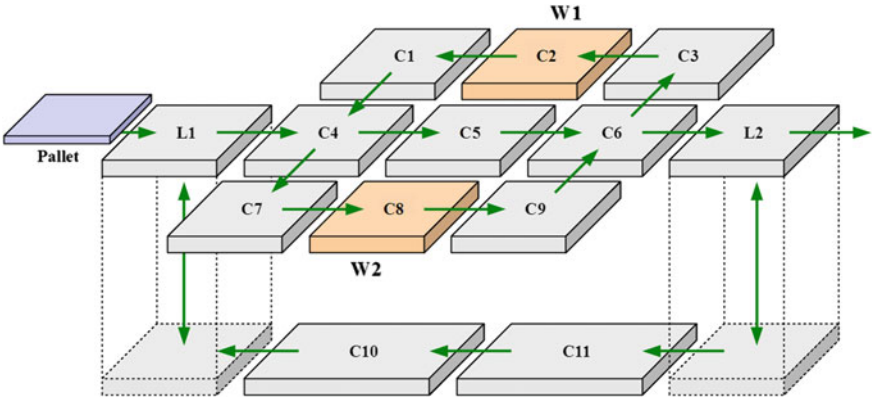

Fig. 2.7 Modular composition of the assembly system

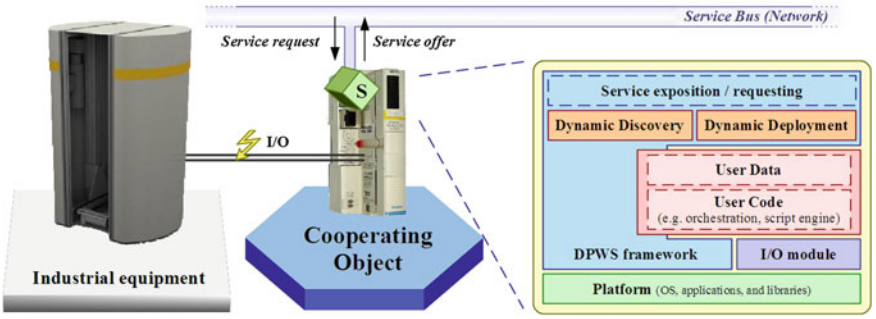

Fig. 2.8 Service-enabled Cooperating Object for industrial automation

concept platform for light assembly, inspection, test, repairing and packing applications. Figure 2.7 shows a representation of the modular composition of the system, using mechanical conveyor modules (C1–C11), lifters (L1 and L2) and workstations (W1 and W2).

The service-enabled Cooperating Objects are the host for most of the services exposed in the system and also responsible for the cooperation and control activities (Fig. 2.8). These devices have two main interfaces: (i) mediating the connection to the shop-floor industrial equipment via I/O (e.g., lifter) and (ii) managing the access to the service bus by exposing and requesting services. The web service infrastructure is based on the SOA4D implementation of DPWS (Device Profile Web Services) (forge.soa4d.org). The Cooperating Object is configurable with the dynamic deployment features available via the SOA4D stack. Once on-line, the Cooperating Object can be discovered (dynamic discovery as defined in the DPWS protocol) and provided services can be requested; similarly it can also request services whenever it needs to.

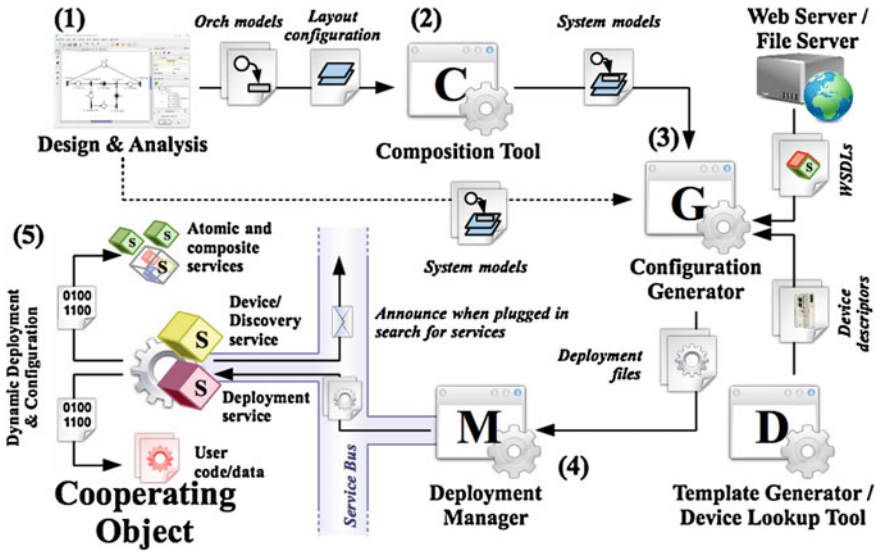

Fig. 2.9 Configuration and deployment tools for service-enabled Cooperating Objects

For the customization of the Cooperating Object in terms of what it should actually do, there is the container for the user code and data. These can be any type of program, for example a reasoning engine with a knowledge base, a script engine that can interpret uploaded scripts or a model-based orchestration engine for reading a given work-plan made of services (an orchestration) and execute it. The last option is used in this work as reference for the system deployment and management. The tool chain needed for composing systems, creating configuration files and deploying those files to Cooperating Objects is depicted in Fig. 2.9, which illustrates the complete engineering sequence from the design of the components or the system, passing by the composition and followed by the deployment to the devices.

As depicted in Fig. 2.9, the sequence starts with the design of component workflow/model (*step 1*) once per device type. Composition of the instance models can be done for one or more system model(s) (*step 2*) followed by the generation of configuration files (*step 3*). This process generates basically two files: (i) a service class descriptor, containing the referenced port types and a model representation, and (ii) and a device descriptor, containing the device and hosted service information, including all discovery hints needed by the execution engine (later on to resolve the referenced component services). The Cooperating Object must be running and ready to receive the configuration (*step 4*). Then the deployment manager id invoked in order to download the descriptor files for a specific system model to the target device. This step is repeated until all models are deployed. A new device is generated if there is server information in the deployment files. The new device can then be used by any client. Once a target has received the configuration and configured correctly, the execution start automatically (*step 5*) by first making announcement and discover

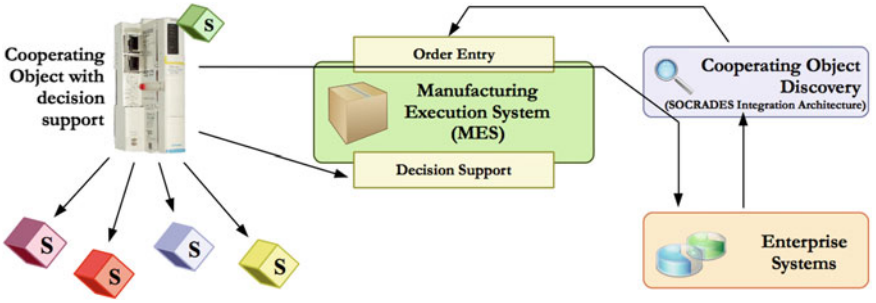

Fig. 2.10 Manufacturing execution system overview

of needed services (dynamic discovery) and proceeding afterwards to the operation defined by the orchestration engine.

As can be seen, the current approach is production agnostic, since it does not contain any information concerning products. This is intended for the separation of concerns, i.e., production and resource management is handled by other Cooperating Objects or services in the system. For production management and information in adjacent to the automation tasks, production orders are integrated via service-orientation from the enterprise resource planning (ERP) system directly on the shop floor. The detailed production steps are stored in the Manufacturing Execution System (MES). This MES interacts with the Cooperating Objects via an orchestration engine and also is in contact with the ERP system as shown in Fig. 2.10.

As mentioned, the depicted system utilizes collaboration among multiple layers i.e., from the devices in the shop-floor, to the MES and the ERP. However to make this possible two issues need to be resolved i.e., (i) dynamic discovery of Cooperating Objects and their services that do not reside on the same physical location or network segment, and (ii) seamless interaction among these Cooperating Objects. To resolve this, we have developed a middleware named SIA (SOCRADES Integration Architecture) [9] developed explicitly with "device-to-business" integration in mind [5]. The middleware offers auxiliary services in interacting with devices and systems. As a complementary functionality a network application (named LDU) is created, that provides discovery of devices and services via DPWS (Device Profile for Web Services) on the local network and connection to the enterprise system. The application is cross-platform ready (prototype is implemented in Java) and hence can be automatically instantiated by a Cooperating Objector even run manually in the local network by the factory operator (by just clicking on it in his web browser). Different versions of the application can add-up functionalities, e.g., proxy also specific enterprise services at the local shop-floor, where they can be discovered and used by the Cooperating Objects as well as other devices and services. The middleware provide a means for connecting and managing devices from different physical and network premises.

The system operates autonomously with information shared among the Cooperating Objects which may spawn various levels e.g., local sensors, MES, enterprise

systems and services. For instance once the workflow is started, the orchestration engine requires a decision to proceed further. To identify the pallet to be handled, the engine gets the RFID number of the associated RFID reader using the matching service. This pallet ID is used to get the next service from the MES. The returned service allows the orchestration engine to proceed. In turn, the pallet is moved on to the determined facility, e.g., workstation 2. Reaching the destination, the accompanied production unit is called to execute a service for the given pallet ID. It is possible to let the system produce the units automatically and allow completing an order without human interaction. This scenario focusing on cooperation happening horizontally at "device" level as well as vertically i.e., among devices and systems/services is part of ongoing efforts to show the added-value benefit that can be materialized with new disruptive technologies and concepts in the domain of industrial automation.

2.3.3 Key Results and Lessons Learned

Taking the granularity of intelligence to the device level allows intelligent system behaviour to be obtained by automatically composing configurations of devices that introduce incremental fractions of the required intelligence. This approach favours adaptability and rapid reconfigurability, as re-programming of large monolithic systems is replaced by reconfiguring loosely coupled embedded units, that can then further enhance their functionalities via cooperation with other devices, systems and services.

The realized service-oriented solution demonstrated the viability to develop complex distributed and Cooperating Object enabled applications, based on service-orientation mechanisms that allow the horizontal and vertical integration. In fact, the service-oriented principles allow to overcome interoperability problems that usually appear in these environments due to the existence of heterogeneous software and hardware applications. A key result demonstrated is related to the flexibility and modularity exhibited by the service-oriented based approach to develop Cooperating Objects solutions. In fact, the on the fly adaptation to unexpected disturbances or even to process changes (in these cases requiring off-line adaptation) is a crucial factor for the success of this kind of solutions.

From a functional perspective, the focus is on managing the vastly increased number of intelligent devices and mastering the associated complexity. From a runtime infrastructure viewpoint, the focus is on a new breed of very flexible cooperative real-time networked embedded devices (wired/wireless) that are fault-tolerant, reconfigurable, safe and secure. Especially auto-configuration management is a new challenge that is addressed through basic plug-and-play and plug-and-run mechanisms.

From technological and infrastructural viewpoints, the use of the Service-Oriented Architecture (SOA) paradigm implemented through web service technologies enables the adoption of a unifying technology for all levels of the enterprise, from sensors and actuators to enterprise business processes. This means that low cost devices may

communicate directly to higher-level systems and enhance their own functionality, which may lead to more adaptive and lightweight approaches.

As already probably noticed, only minimal assumptions about the concrete production line are present in the whole system design. It is important to recall here that the operational behaviour of the devices is self-controlled and guided by internal/external events that also may correspond to service calls. Collaboration among the stakeholders is made possible with dynamic discovery and seamless interaction among them. Additionally more sophisticated approaches can be realized based on orchestration of the existing Cooperating Objects in other Cooperating Objects as well as with the support of the infrastructure [5]. We have also to point out that this approach is not limited to the specific depicted example case, but other domains and systems may also benefit from it, leading to a continuously evolvable infrastructure [2] that may adapt to the business needs and hence provide a competitive advantage.

2.4 Light-Weight Bird Tracking Sensor Nodes

2.4.1 Overview

There are mainly two methods in use for tracking the behaviour of birds. Birds are either tracked by means of applying a sensing unit or some sort of marker onto individual birds or they are manually tracked by people spotting flocks of birds or maybe individuals of rare species. Unfortunately, both ways are tedious ways for approaching the problem, in particular, due to the manual labour involved for recollecting units and the uncertainty in observations when it comes to gathering continuous traces. Also, these methods are not always sufficient from an application point of view. For instance, it is difficult to track the influence of man-made structures on the behaviour of birds that way, which is often necessary information for biologists studying bird species. The UvA Bird Tracking System (BiTS) aims at overcoming such issues. A reconfigurable bird-tracking platform has been deployed that allows long-term application of harvesting-enhanced GPS-enabled sensor nodes with wireless readout. The possibility to download birds' data through permanently installed base stations feeding into a database with a 'virtual lab' web front-end allows collecting long-term high quality traces of birds.

Though the system is readily available and working, implications stemming from the challenging application context make up a number of interesting research questions that need to be addressed. Designing a structure that is to be deployed on birds puts severe constraints on weight, size and durability which in turn translates into limited performance (i.e., limited data that can be collected or that can be downloaded by a base station). In a joint effort with UvA, The research aim is targeting research and evaluation of cooperative networking behaviour among sensor nodes on different birds.

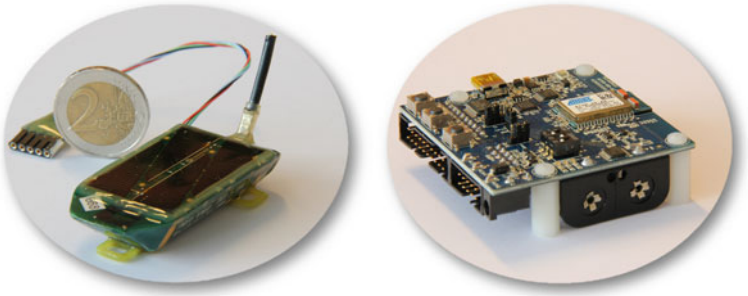

Fig. 2.11 UvA BiTS mote with programming interface exposed and a BlueBean development kit for radio evaluation networking are shown

Intelligent data dissemination needs to be applied in the context of intermittently connected networks to improve system performance. This translates into the need for accurately profiling and modelling the system as well as setting up realistic and highly scalable simulation environments that need to be tested with hands-on experience from real-world experiments. This way, applicability of novel protocols can be tested based upon existing traces and simulated behaviour can be checked by means of comparison of its characteristics with results from real-word experiments. Figure 2.11 shows an actual hardware prototype and part of a development kit for evaluating networking protocols. Outcomes from using a simulation environment will be discussed later.

2.4.2 Application Description/Usage Scenarios

As shown in Fig. 2.12, the Uva BiTS architecture comprises multiple bird tracking applications and combines their information in a database. Readily completed applications include a case study on Common Buzzard flight activities at an airport, migration behaviour of seagulls and their interaction with wind farms. Exemplary results from tracing flight activities at an airport are shown in Fig. 2.13.

For bird tracking applications at airports, options like manual bird spotting, using marking or sensing devices without radio communication are out of the question due to hard real-time constraints. Designing a bird monitor, tracking e.g., flocks of geese flying low and close to an airport, requires real-time information for avoiding potentially dangerous bird strikes. Employing a monitor by means of the UvA BiTS platform might allow to not necessarily track flocks of birds but also to possibly recognize encounters with base stations deployed around the airport.

One of the main application symbolized in Fig. 2.12 is monitoring foraging and tracking migration of birds. Several applications have been implemented for foraging (e.g., tracking Oystercatcher in the Dutch Wadden Sea) and for studying cross-continental migration behaviour. For tracking foraging behaviour, it is particularly

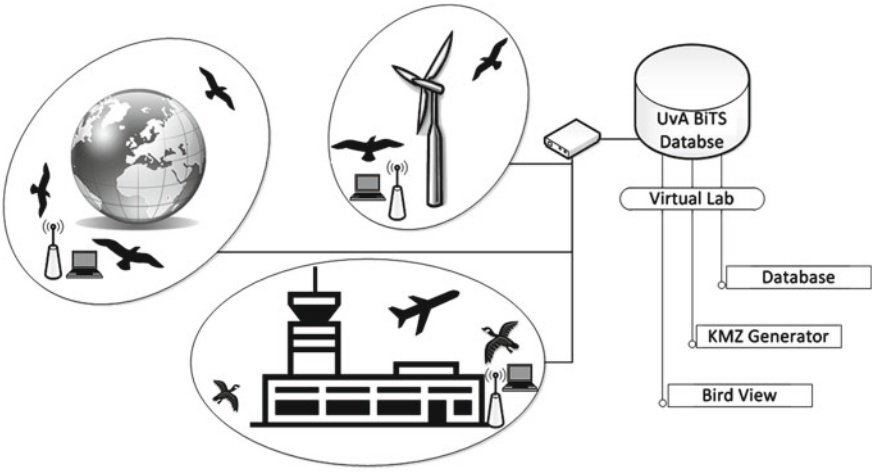

Fig. 2.12 Multiple applications are run side by side eventually sharing base stations

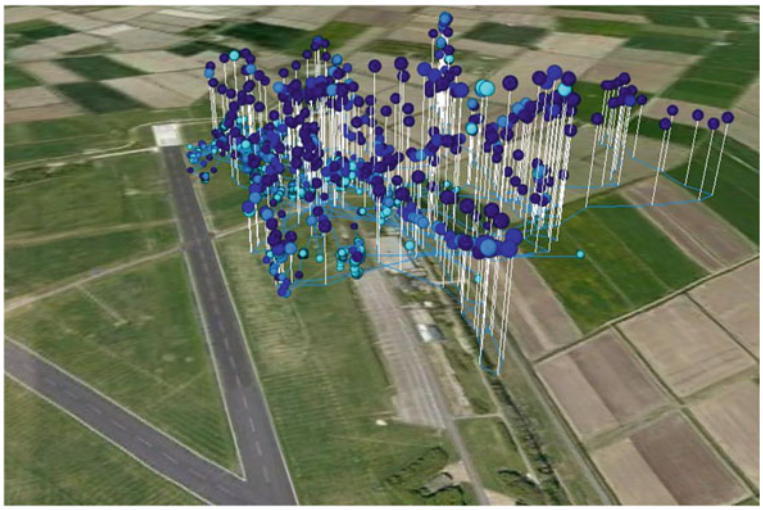

Fig. 2.13 GPS trace of Common Buzzard flying at an airport. Taken from [10]

beneficial to have a base station close to where motes have been deployed. For the case of tracking migration behaviour it usually gets more complicated. Memory load gets higher between consecutive readouts and cooperative readout among multiple base stations can be beneficial.

Last but not least, the interaction of birds with man-made structures is being monitored. Successful traces of seabird-windfarm interactions have been taken at Orford Ness close to the east coast of England. 25 Lesser Black-backed Gulls have been attached nodes of the UvA BiTS platform in 2010 and 2011. Interaction with

wind farms can be an important issue to be considered when it comes to conservation of birds' natural habitats. Plans had been made to set up windfarms at Brenner in western Austria being one of the main transit routes through the Alps. However, it also one of the main migration routes for birds crossing central Europe through the Alps. Information that could be collected with the BiTS could be invaluable for studying how bird migration routes are impacted from men-made structures, but also from long-term changes of climate conditions.

2.4.3 Key Results and Lessons Learned

System development is challenging because of issues with long-term application life cycles compared to short-lived state-of-the-art in the rather new field of harvesting-enabled sensors with hard-to-predict mobility. Setting up a bird-tracking application may take years of planning and testing of a deployment, because of yet-to-be validated yearly migration patterns of birds. However, history tells us that hardware components and protocols can be utilized in ever more efficient ways—a couple of years ago UvA BiTS might not have been considered feasible—which does not match the speed of possibly adapting the application at hand. Running the system for years, the information that is traced by birds becomes more and more valuable (i.e., long lasting traces allow for more expressive studies of changing bird behaviour) and therefore newly deployed platforms/applications must not conflict (i.e., be backwards compatible) with older applications that are still running. This mismatch between innovation in technology (of course the costumer wants to work with the highest possible performance for any new platform) and long application lifecycle demands for careful consideration at every new design step. Existing compatibility between applications and versions of platforms must not be violated and future compatibility must not be hindered either—a major demand in the domain of Cooperating Objects.

As being a key element for any wireless Cooperating Objects, communication standards and respective hardware components and protocol implementations deserve special consideration when it comes to compatibility in long-term deployments. In particular in the case of hard-to predict mobility, collecting and reprogramming sensors is not applicable means of avoiding having outdated instances of the platform in the field. Due to those issues being described above, SerialNet (providing versatile reconfiguration features) over ZigBee-compliant (with ZigBee being the only de-facto small-footprint wireless industry standard widely in use) ZigBit running on a far-spread AVR-based platform (enforcing long lasting support and further development of the system).

Currently, wireless communication capabilities are made use of for wireless readout functionality and eventual multi-hop communication for the non-intermittently-connected case. This means that first of all, direct (single hop) readout is performed, and furthermore, readout via relaying nodes (ZigBee router functionality) is possible as well as long as the end-to-end channel is connected and does not get blocked by other communication. This raises the question whether data good put could be

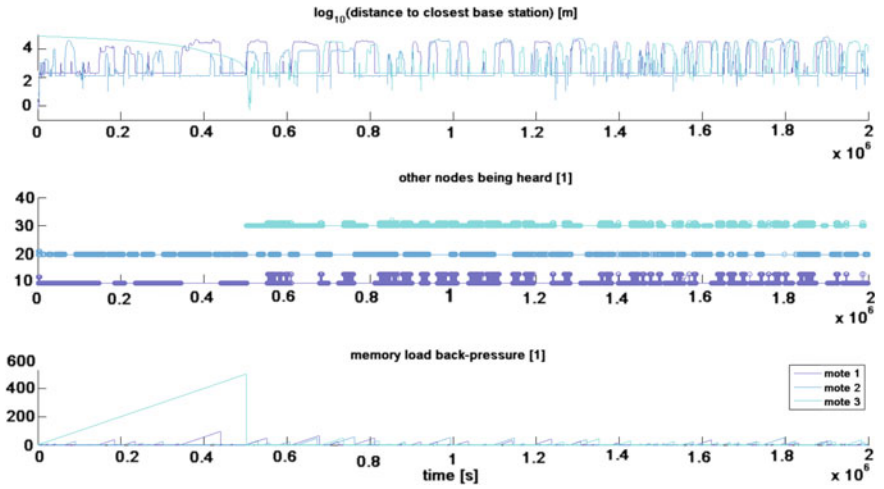

Fig. 2.14 Sample evaluation of three gulls' traces around a base station in Texel (The Netherlands)

increased by means of data muling or message ferrying as known from other sensor network applications encountering the intermittently connected (delay tolerant) case. Due to the dependable nature of the BiTS application, it is worth it to first study birds' mobility and interaction characteristics before deploying a novel networking approach that is possibly degrading existing deployments' performance. Figure 2.14 depicts a snapshot of a possible scenario rerunning traces that have actually been gathered in the wild and three motes' connectivity among each other and to a base station while simulating (using a modified Castalia) a wireless readout protocol similar to the one being used in the actual application.

Figure 2.14 shows the motes' distance to their closest base station in the uppermost plot. Next, other nodes or base stations that could be recognized are being indicated. Circles at 10/20/30 indicated base station contact, where circles at 10/20/30+i indicate contact to node 'i'. The lowermost plot compares the memory load of different motes given that GPS readings are taken at a constant rate and the mote with the higher memory load is allowed to offload data if multiple nodes are in base-station range at the same time. Three key observations can be made from looking at this snapshot of normalized information.

- There would possibly be ways of improving good put performance if the birds' movements would have been known in advance. Thus using delay tolerant networking approaches might be beneficial for the application. However, irregularity in the application scenario may drastically change the constraints which need most attention at a time, as can be seen with the peak load of memory at mote 3—not to mention the variability of available energy which has been omitted from the plots.
- However, different bird (and even different researcher) behaviour may vary and lead to different performance among different runs with one and the same protocol.

Obviously, mote 3 was activated at a way different place than the other motes, though belonging to the same deployment. Furthermore, mote 2 has hardly any contact to other birds' motes, though they all have contact to the same base station on a regular basis.

- The main lesson being learned when inspecting real-world traces is that one is dealing with a highly dependable system, where it is difficult to tell in advance, what constraints will be put to their limits to what extent and how often. Therefore, extensive simulation is necessary before protocol changes or architectural modifications can be carefully incorporated into the system.

Despite the fact that numerous problems have successfully been overcome as results of many deployments have shown, there are still a couple of optimizations that are currently being researched. In particular energy and memory efficiency and capacity receive attention as does networking and data communication good put.

2.5 Public Safety Scenarios

2.5.1 Overview

Public safety organizations such as first responders, fire fighters, police, and military units need robust communication networks to cooperate and transmit different kind of sensor information. These networks have to be reliable even when a pre-deployed infrastructure has been destroyed. Wireless multi-hop networks (such as Mobile Ad-Hoc Networks (MANETs), Wireless Sensor Networks (WSNs), and Wireless Mesh Networks (WMNs)) promise to meet the requirements of (1) spontaneous deployment, (2) being independent of any kind of existing infrastructure, and (3) robustness in the sense of self-organization and self-healing by their very definition. These networks have been a topic in research for more than a decade now. Recently, real-world tests and deployments provided valuable insights concerning challenges and future research directions. There are different mesh and WSN testbeds (e.g., [11–13]) enabling the research community to run tests in static and mobile real-world networks. However, concerning Cooperating Objects in public safety scenarios, there are significantly different requirements:

- Spontaneous deployment,
- Mobility typical for public safety scenarios,
- Typical applications and traffic for public safety scenarios.

Due to these characteristics, deploying Cooperating Objects in public safety networks is a huge challenge. To overcome this challenge, we have developed *Bonn Sens* [14, 15] a prototype based on commercial off-the-shelf (COTS) hardware. The prototype comprises typical public safety applications and is spontaneously deployable. Furthermore, this prototype enables us to perform evaluations with real public safety end-users, e.g., by deploying the prototype in manoeuvres.

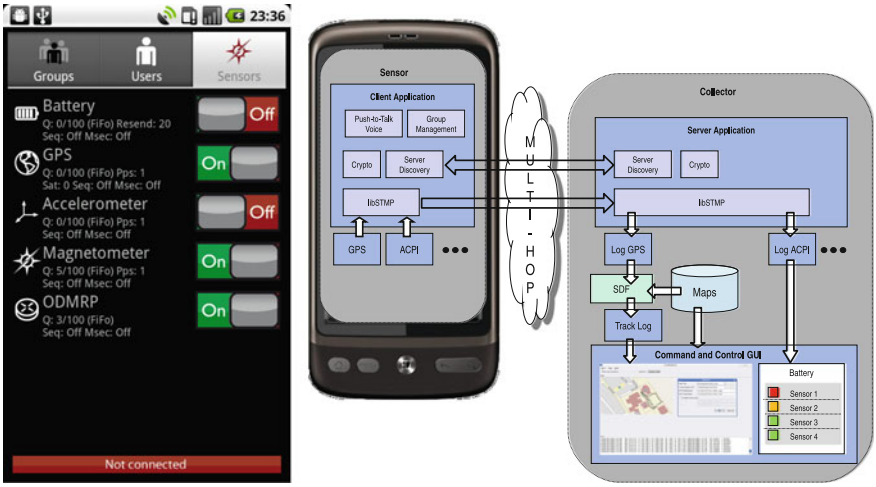

Fig. 2.15 BonnSens System

2.5.2 Application Description/Usage Scenarios

In public safety scenarios there are two common requirements for a command and control system: (1) push-to-talk voice communication and (2) map-based (blue-force) tracking. To communicate inside a team, squad, group, and platoon as well as in between, talk-group based voice communication is important. In addition, the central as well as local command and control points need to know where their units are.

A suitable Cooperating Objects architecture for this scenario consists of a distributed sensor and collector application for the transmission of sensor data over a wireless multi-hop network as shown in Fig. 2.15. The Cooperating Objects support the collection of sensor information (e.g., Global Positioning System (GPS), accelerometers, and magnetometer) via modular extensible plugins. The transmitted sensor data is stored in a database by the collector on the central side. Depending on the type of sensor, the data can be visualized using different types of Graphical User Interface (GUI). Some kind of data, such as positioning data, may be additionally processed by a sensor data fusion algorithm before being visualized on a map. The voice communication is realized using a peer-to-peer based voice application including a dynamic group management on the lightweight nodes.

The architecture consists of two components: (i) portable, lightweight sensing objects and (ii) fully-equipped collector objects. For the sensor objects standard COTS smartphones are used. Smartphones available on the market today, often come with integrated sensors like GPS, accelerometer, and compass. These devices are an ideal basis for the lightweight objects. However, smaller WSN mote like objects can in principle be used as well.

We have implemented a client sensing application for Android OS. Figure 2.15 shows a screenshot of our sensor data client application *BonnSens*. For the lightweight objects, we currently use the HTC Desire smartphone. For the fully-equipped collector objects, standard COTS laptops can be used. In the last deployments, we used a Dell Precision M4300 (Intel Core 2 Duo T7700 2,4 GHz and 4 GB RAM) running Ubuntu 10.04. All objects (lightweight and fully-equipped) can be used as relays depending on the topology. However, to gain more robust topologies and to safe energy at the lightweight objects, we add a mesh backbone. For the mesh, we tested two kinds of COTS mesh routers: (1) ASUS WL-500g Premium V1 (266 MHz ARM processor, 8 MB Flash memory, 32 MB RAM) and (2) ALIX 3D2 (500 MHz AMD Geode LX800, 1 GB Compact Flash memory, 256 MB RAM). In both routers we use WiFi-cards with Atheros chipsets (TP-LINK TL-WN660G). For an easy on-site deployment an infrastructure-independent power support is a requirement. Thus, we use motorbike-batteries with 12 V-20 Ah. Using these batteries, we can run the mesh backbone for more than 12 h without any infrastructure.

For the relaying we implemented a multicast routing approach, as both core applications, voice communication and blue-force tracking, imply multicast. The voice application is group-based. Thus, it can be efficiently realized by multicast-groups. In some public safety scenarios, there may be several fully-equipped objects with a demand for a visualization. As multicast routing protocol we chose the reactive On-Demand Multicast Routing Protocol (ODMRP) [16]. ODMRP is a mesh-based approach based on scoped flooding. A selected subset of nodes forwards the packets. We chose ODMRP as it showed promising results in public safety specific simulative performance evaluations [17]. Furthermore, it showed to behave quite reliable even under attacks like sinkholes, as the mesh structure provides robustness against the attraction of routes. We implemented ODMRP using the Click modular routing framework [18]. The Click user-space mode enables us to run ODMRP on the mesh routers as well as on Android phones.

To provide functionalities for sensors such as registering at a receiver, timestamping of sensor data, synchronization of all nodes, as well as providing sensor management functions, we specified and implemented the Sensor Data Transmission and Management Protocol (STMP) [19]. In order to realize a consistent implementation of STMP for lightweight as well as for fully-equipped objects, we modularized STMP by implementing it as a commonly usable library in C named *libSTMP*. Client applications may thus be programmed either using the device specific API (e.g., Android API for the deployment on smartphones) or using native C code (e.g., for the deployment on a laptop) both accessing the same STMP library.

To support tracking of a sufficient accuracy even in complex scenarios such as urban canyons, appropriate sensor data fusion algorithms have been integrated. As shown in [20] a standard Kalman filter [21] may suffer significantly from Out-of-Sequence (OoS) measurements. An Accumulated State Density (ASD) methodology [22] which allows to calculate the impact on all states within a given time window proved to be a valuable alternative.

2.5.3 Key Results and Lessons Learned

We present the lessons learned during our deployments. To get feedback on our approaches as early as possible, we started to deploy early versions of our architecture in manoeuvres. We deployed mesh backbones as well as portable, lightweight sensing nodes. The latter ones are smartphones carried by the units. To date, we have done spontaneous deployments in four manoeuvres in cooperation with the Johanniter Academy in Münster (Germany) on the manoeuvre ground of the institute of fire-fighting. During these manoeuvres, we had to learn several lessons that we will discuss now. By doing so, we will also describe challenges typical for public safety networks and systems.

As portable, lightweight sensing objects, smartphones are used in our system. Many smartphones on the market today have integrated sensors like GPS. Thus, these devices seem to be an ideal basis for the map based tracking part of our command and control system. However, just visualizing raw GPS positions of all lightweight objects is not sufficient for the requirements in public safety scenarios. An accuracy of 1m is typically requested. Such an accuracy is challenging on typical manoeuvre sites due to obstacles, etc. that may temporarily prevent the proper reception of a GPS signal. Thus, the raw positions have to be filtered and fused. Using standard filters such as a Kalman filter in the system yields to new challenges such as OoS measurements (cf. [20]). The filters need to be optimized for the usage in typical multi-hop networks.

For the measurements as well as for the units that want to use the command and control system, it is important that all objects do not run out of battery. If necessary, single batteries have to be exchanged. For doing so, it is important to know the objects that have battery problems. Thus, we learned that the most important sensor data to be aware of during a deployment is the battery power of the devices deployed. Furthermore, energy awareness in general is very important.

Some voice messages and sensor information transmitted may have higher importance than others. This becomes relevant especially when the data rate is limited due to sub-optimal signal propagation characteristics or resource constrained nodes. Furthermore, when messages are transmitted as broadcasts on the link layer, the basic rate is used. This may lead to additional rate limitation. Thus, data prioritization or, more general, communication and sensor management needs to be implemented. In our system, we implemented the Sensor Data Transmission and Management Protocol (STMP) [19] to take care of prioritization as well as communication and sensor management.

Especially in early deployments, it is important to save all data locally as well—just in case there is a problem with the network (e.g., limited data rates). This also allows for an easier debugging. However, when relying on local logs, non-synchronized clocks may be a challenge. Furthermore, approaches well-described in the literature and evaluated in simulations or labs, may not run very well in real deployments. For example, links may be extremely variable which yields suboptimal performance of some approaches.

2.6 Road Tunnel Monitoring and Control

2.6.1 Overview

State-of-the-art solutions for road tunnel lighting either use pre-set light levels based on date and time, or adjust the lights based on an open-loop regulator relying on an external sensor. Both solutions disregard the actual lighting conditions inside the tunnel, and may endanger drivers or consume more power than needed. The solution developed within the TRITon (*T*rentino *R*esearch and *I*nnovation for *T*unnel M*oni*toring, triton.disi.unitn.it) project deployed a WSN along the tunnel walls to measure the light intensity and report it to a controller, which closes the loop by setting the lamps to match the lighting levels mandated by law. Unlike conventional solutions, our system adapts to fine-grained light variations, both in space and time, and dynamically and optimally maintains the legislated light levels. This enables energy savings at the tunnel extremities, where sunlight enters, but it is also useful inside the tunnel to ensure the target light levels even when lamps burn out or are obscured by dirt. The overall architecture for adaptive lighting, of which the WSN is an integral element, was awarded a European patent in March 2012.

The system was developed with the goal of reducing the management costs of road tunnels and improving their safety. Our WSN-based control system has been installed since August 2010 in an *operational* tunnel on a high-traffic freeway, where it has been running without intervention. Our contributions range from hardware to software, with the former built on top of our TeenyLIME middleware. Based on measurements and calculations, the energy consumption in our tunnel is up to 50 % less than a solution with standard technologies.

2.6.2 Application Description/Usage Scenarios

The system as shown in Fig. 2.16 contains many components working in concert to monitor and control the loop. The principal element collecting the light values inside the tunnel is a WSN composed of approximately 90 nodes divided between the two carriageways of the tunnel. The sensed values are collected by four gateways, combined with the value of an external luminance sensor and sent to an industrial PLC. The PLC implements a centralized control logic to determine the lamp levels required to meet the legislated levels and sends commands to the individual lights to establish these levels. In addition, a SCADA subsystem is connected to the PLC and provides an interface to the human operator for visualization and manual control.

The WSN nodes are functionally equivalent to TelosB motes, equipped with an MSP430 microcontroller, a Chipcon 2420 radio chip, and an on-board inverted-F microstrip antenna. A custom sensor board is attached and contains 4 ISL29004 digital light (illuminance) sensors, and 1 TC1047A temperature sensor. Each node

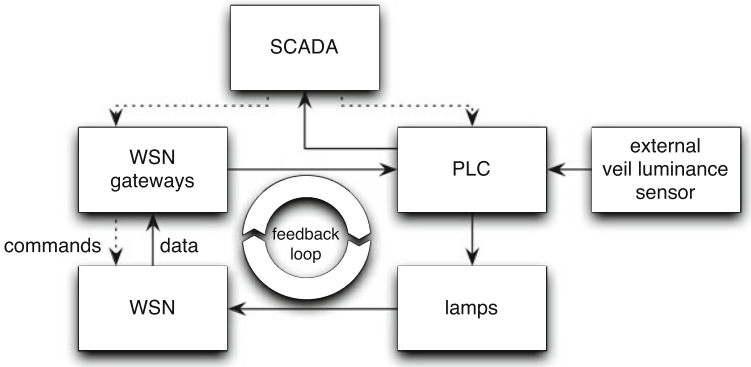

Fig. 2.16 Tunnel System Architecture

is powered by 4 Duracell Procell D-size batteries and placed inside an IP65 (water and fireproof) polycarbonate box with a transparent cover.

We also placed four Verdex-Pro embedded computers by Gumstix to serve as the gateway nodes collecting light samples. These are powered and connected via Ethernet to the PLC and SCADA.

The software installed on the motes runs on top of TinyOS [23]. However, unlike the vast majority of other deployments where application and system software sit directly on the operating system, we built on top of the TeenyLIME [24] middleware (teenylime.sourceforge.net).

This choice was motivated by the fact that TeenyLIME was used successfully by our group in another long-term, real-world deployment for structural health monitoring [25], where its higher-level abstractions were shown to reduce the overall code footprint, allowing one to pack more functionality on memory-restricted nodes.

The main software components are for sensing, data collection and data dissemination. Sensed light values are reported by each node every 30 s, however the reported value is computed over multiple samples from each of the four sensors, eliminating outliers and averaging the remaining samples. To collect the data over multiple hops, we implemented our own tree-based collection protocol that uses LQI to measure link quality. This is motivated by the experimental observation that the resulting trees are similar to those obtained with ETX-based protocols, but with much less overhead [26]. Our protocol also supports multiple sinks, with the best sink being identified implicitly by choosing as parent the neighbour with the smallest node-to-sink routing cost. By periodically refreshing the tree, the routing paths adjust to changes in the topology and all nodes will automatically recover in case a sink fails. Finally, we use a hop-by-hop recovery mechanism to ensure data reliability. The last component, data dissemination, is used to dynamically reconfigure system parameters such as the sampling frequency.

2.6.3 Key Results and Lessons Learned

Our experiences from more than two years of working in actual tunnels are reported next.

Our final deployment is on a high-traffic road, therefore carrying out experiments on this site would have been impractical due to the need to partially or totally block the road. Therefore, we obtained access to a shorter, lower-traffic tunnel that served as a testbed we could more easily access to run experiments. These tests were critical to test the stability of the system, evaluate the energy consumption, and establish critical application-level parameters. In this test scenario we were able to identify critical problems in both the hardware and software which could not have been detected in a testbed environment. For example, the mounting mechanism attaching the original sensor board to the mote was affected by vibrations caused by traffic in the tunnel. The sensor boards installed in the final system have a different mounting mechanism that does not suffer from this problem. While the tests were invaluable, there were still differences between the test tunnel and the final tunnel. Specifically, in the test tunnel, 44 nodes were spread along 132 m instead of the target 630 m. While we performed a few tests with the target node density to understand the behaviour of the routing protocol, we could not anticipate all issues that arose in the final test tunnel. For example, while our estimates for node lifetime were accurate, we saw an unexpected pattern of node death in the final tunnel.

We designed our system to support the needs of the adaptive lighting application. However, in the course of the project, we used the same system to test an innovative system for fire detection as well as to monitor carbon monoxide levels inside the tunnel. The former was done with minor modifications to the original system, e.g., increasing the sampling frequency and separating out the infrared component from the light sensors. By doing so, we were able to accurately track the location of a propane flame on the back of a firetruck as it moved through the tunnel. For CO detection, we designed a new sensor board with a CO sensor, then used the backbone of light sensors to transmit the CO data to the gateways.

As this system is installed in a real road tunnel, controlling lights that fundamentally affect the safety of the drivers, it is important that the system function at all times. While it is possible to signal the failure of a single node, indicating that it should be repaired, the system must continue to function. In case of significant failures, the system can fallback to using whole-tunnel pre-set light levels, but single node failures must be compensated. Therefore, the node density must be sufficient that a single node failure does not disconnect the tree. Further, we introduced multiple gateways in each tunnel in order to survive single gateway failures.

As designed, our system has node lifetimes above one year. However, additional studies done by our group have shown that we can increase lifetime by applying model-driven data acquisition, a technique that aims to reduce the amount of data reported by each node. At each node, a model predicts the sampled data; when the latter deviate from the current model, a new model is generated and sent to the data

sink. With a simple derivative-based prediction model that is easy to implement on the limited capacity WSN nodes, our experiments suggest that the system lifetime can be tripled [27].

2.7 Conclusions

As a result of the efforts described above and of the general state of the art, many of the research questions related to deployment and management of Cooperating Objects are now well understood, while the corresponding solutions are still slowly making it into mainstream practice.

The missing tile to the puzzle, which is going to boost the acceptance of Cooperating Objects technology and correspondingly open new market avenues, lies in the standardization and integration of the methodologies at stake, which still partly represent ad-hoc or isolated efforts. Establishing a sound basis in the deployment and management of Cooperating Objects will indeed lessen the burden to create products out of research prototypes.

References

1. Mottola L, Picco GP (2011) Programming wireless sensor networks: fundamental concepts and state of the art. ACM Comput Surv 43(3):19:1–19:51. doi:10.1145/1922649.1922656
2. Colombo AW, Karnouskos S, Mendes JM (2010) Factory of the future: a service-oriented system of modular, dynamic reconfigurable and collaborative systems. In: Benyoucef L, Grabot B (eds) Artificial intelligence techniques for networked manufacturing enterprises management. Springer, New York. ISBN 978-1-84996-118-9
3. Karnouskos S, Baecker O, de Souza LMS, Spiess P (2007) Integration of SOA-ready networked embedded devices in enterprise systems via a cross-layered web service infrastructure. In: Proceedings of IEEE conference on ETFA emerging technologies and factory automation, pp 293–300. doi:10.1109/EFTA.2007.4416781
4. Colombo AW, Karnouskos S (2009) Towards the factory of the future: a service-oriented cross-layer infrastructure. In: ICT shaping the world: a scientific view, ISBN: 9780470741306, European Telecommunications Standards Institute (ETSI), Wiley, Sophia-Antipolis
5. Karnouskos S, Savio D, Spiess P, Guinard D, Trifa V, Baecker O (2010) Real world service interaction with enterprise systems in dynamic manufacturing environments. In: Benyoucef L, Grabot B (eds) Artificial intelligence techniques for networked manufacturing enterprises management, ISBN 978-1-84996-118-9. Springer, London
6. Taisch M, Colombo AW, Karnouskos S, Cannata A (2009) Socrades roadmap: the future of soa-based factory automation. http://www.socrades.eu/Documents/objects/file1274836528.84
7. Karnouskos S, Colombo AW (2011) Architecting the next generation of service-based SCADA/DCS system of systems. In: 37th annual conference of the IEEE industrial electronics society (IECON 2011), Melbourne
8. Karnouskos S, Colombo AW, Bangemann T, Manninen K, Camp R, Tilly M, Stluka P, Jammes F, Delsing J, Eliasson J (2012) A SOA-based architecture for empowering future collaborative cloud-based industrial automation. In: 38th annual conference on the IEEE industrial electronics society (IECON 2012), Montréal, Canada
9. Spiess P, Karnouskos S, Guinard D, Savio D, Baecker O, Souza LMSd, Trifa V (2009) SOA-based integration of the internet of things in enterprise services. In: Proceedings of IEEE international conference on web services, (ICWS 2009), Los Angeles, pp 968–975. doi:10.1109/ICWS.2009.98

10. University of Amsterdam (2012) UvA bird tracking system. http://www.uva-bits.nl/other-projects/
11. Bicket J, Aguayo D, Biswas S, Morris R (2005) Architecture and evaluation of an unplanned 802.11b mesh network. In: Proceedings of the 11th ACM internernational conference on mobile computing and networking
12. Raychaudhuri D, Seskar I, Ott M, Ganu S, Ramachandran K, Kremo H, Siracusa R, Liu H, Singh M (2005) Overview of the ORBIT radio grid testbed for evaluation of next-generation wireless network protocols. In: Proceedings of the IEEE wireless communications and networking conference
13. Werner-Allen G, Swieskowski P, Welsh M (2005) Motelab: A wireless sensor network testbed. In: Proceedings of the fourth international conference on information processing in sensor networks (IPSN'05), special track on platform tools and design methods for network embedded sensors (SPOTS)
14. Aschenbruck N, Bauer J, Ernst R, Fuchs C, Kirchhoff J (2011) Demo abstract: a Mesh-based command and control sensing system for public safety scenarios. In: Proceedings of the 9th ACM conference on embedded networked sensor systems
15. Aschenbruck N, Bauer J, Ernst R, Fuchs C, Kirchhoff J (2011) Deploying a mesh-based command and control sensing system in a disaster area maneuver. In: Proceedings of the 9th ACM conference on embedded networked sensor systems
16. Lee SJ, Gerla M, Chiang CC (1999) On-demand multicast routing protocol. In: Proceedings of the IEEE wireless communications and networking conference (WCNC), pp 1298–1302
17. Aschenbruck N, Gerhards-Padilla E, Martini P (2009) Modeling mobility in disaster area scenarios. Elsevier Perform Eval Spec Issue Perform Eval Wireless Ad Hoc Sens Ubiquitous Netw 66(12):773–790
18. Kohler E, Morris R, Chen B, Jannotti J, Kaashoek FM (2000) The click modular router. ACM Trans Comput Syst 18(3):263–297
19. Aschenbruck N, Fuchs C (2011) STMP—sensor data transmission and management protocol. In: Proceedings of the 36th IEEE conference on local computer networks
20. Govaers F, Fuchs C, Aschenbruck N (2010) Evaluation of network effects on the Kalman filter and accumulated state density filter. In: Proceedings of the 2nd international workshop on cognitive information processing
21. Bar-Shalom Y, Li XR, Kirubarajan T (2001) Estimation with applications to tracking and navigation. Wiley, New York
22. Koch W (2009) On accumulated state densities with applications to out-of-sequence measurement processing. In: Proceedings of the 12th ISIF international conference on information fusion
23. Hill J, Szewczyk R, Woo A, Hollar S, Culler D, Pister K (2000) System architecture directions for networked sensors. In: Proceedings of the 9th international conference on architectural support for programming languages and operating systems, pp 93–104. doi:10.1145/378993.379006
24. Costa P, Mottola L, Murphy AL, Picco GP (2007) Programming wireless sensor networks with the TeenyLIME middleware. In: Proceedings of the 8th ACM/USENIX international middleware conference
25. Ceriotti M, Mottola L, Picco GP, Murphy AL, Guna S, Corrà M, Pozzi M, Zonta D, Zanon P (2009) Monitoring heritage buildings with wireless sensor networks: the Torre Aquila deployment. In: Proceedings of the 8th international conference on information processing in sensor networks (IPSN)
26. Mottola L, Picco G, Ceriotti M, Guna S, Murphy A (2010) Not all wireless sensor networks are created equal: a comparative study on tunnels. ACM Trans Sens Netw (TOSN) 7(2):149–162
27. Raza U, Camerra A, Murphy A, Palpanas T, Picco G (2012) What does model-driven data acquisition really achieve in wireless sensor networks? In: Proceedings of the 10th IEEE international conference on pervasive computing and communications (PerCom 2005), IEEE computer society

http://www.springer.com/978-3-642-45400-4

Applications and Markets for Cooperating Objects

Karnouskos, S.; Marrón, P.J.; Fortino, G.; Mottola, L.;
Martínez-de Dios, J.R.

2014, XIV, 120 p. 44 illus., 40 illus. in color., Softcover
ISBN: 978-3-642-45400-4