J Supercomput (2014) 70:1120–1141
DOI 10.1007/s11227-014-1160-7

Robust architecture for distributed intelligence in an IP-based mobile wide-area surveillance system

Mikko Nieminen · Nikolay Tcholtchev · Ina Schieferdecker · Tomi Räty

Published online: 25 March 2014

Abstract Modern IP-based wide-area surveillance systems often build on networks of multi-modal, intelligent and mobile sensor units. Detection of complex events is performed on intelligent sensors and fusing input in the sensor units or centralized control room components. The domain of surveillance and public safety creates requirement for robustness and fault-tolerance. This article will present an automated intelligence architecture for mobile surveillance, which provides capabilities for combining onboard event detection in sensor units, centralized decision making on the server side, and automated exploitation of mobile surveillance unit positioning data. This architecture must be very reliable to provide services in the face of challenges such as natural disasters and fire, potentially damaging the infrastructure of the surveillance system. To increase its reliability and robustness, we study the introduction of a self-healing system into the architecture and examine the combined system's operation in three case studies.

Keywords Intelligent surveillance · Distributed surveillance · Decision making · Positioning · Self-healing

M. Nieminen (✉) · T. Räty
VTT Technical Research Centre of Finland, Oulu, Finland
e-mail: mikko.nieminen@vtt.fi

T. Räty
e-mail: tomi.raty@vtt.fi

N. Tcholtchev · I. Schieferdecker
Fraunhofer FOKUS, Berlin, Germany
e-mail: nikolay.tcholtchev@fokus.fraunhofer.de

I. Schieferdecker
e-mail: ina.schieferdecker@fraunhofer.de

1 Introduction

In recent years, surveillance systems for public safety have evolved into distributed and intelligent sensor networks [1]. The key building blocks of many current generation surveillance systems consist of (1) a network of sensors, potentially multi-modal and mobile, (2) an IP-based transport network which often allows wireless connectivity, and (3) centralized components providing functionalities such as decision making and monitoring. The increasing amount of sensors introduces more need and possibilities for distributed intelligence, while the use of wireless sensor networks facilitates the introduction of mobile and intelligent surveillance units on the field. One of the challenges within current generation surveillance systems is to establish an efficient distributed architecture design facilitating information acquisition via analysis [1]. As more systems employ wireless surveillance units, the need for providing real-time location-specific information should also be addressed [1].

The surveillance domain carries high demand for reliability and robustness [2], to maintain real-time performance in incidents [3]. In current IP-based surveillance systems, handling network failures is critical for achieving the required level of reliability [4]. The use of wireless and mobile devices further increases the need to tackle unexpected failures automatically, e.g., using self-configuration principles [1].

In this article, our aim is to specify a robust self-healing software architecture applicable to different mobile surveillance scenarios. This architecture should take advantage of distributed intelligence and positioning data of multiple mobile surveillance systems, to realize automated event analysis. Furthermore, the architecture should facilitate automated self-healing methods to survive failures, increasing the reliability and robustness of the system.

To best exploit the possibilities brought in by modern surveillance technology in the context of mobile wide-area surveillance, the architecture should realize the following key features:

- Use of intelligent sensors on the mobile surveillance units to perform distributed event detection
- Use of centralized intelligence to detect complex events from multiple input sources and automatically co-ordinate actions of the mobile units
- Exploitation of positioning information of the mobile units and detected events in the decision-making process for increased efficiency.

The rest of the article is arranged as follows. In Sect. 2, we study prior surveillance system designs related to sensor unit co-operation and sensor unit mobility. In Sect. 3, we propose a software architecture for intelligent mobile surveillance, and introduce an autonomic self-healing solution to this architecture in Sect. 4. We describe three case studies to investigate the use of self-healing in the surveillance system context in Sect. 5. In Sect. 6, we present our conclusions with topics for future work.

2 Related works

VTT Technical Research Centre of Finland has previously presented the Single Location Surveillance Point (SLSP) system for IP-based intelligent surveillance, supporting

multi-modal sensor fusion and automated reasoning. The SLSP system was designed to be deployed in a single location such as a large room or hallway. In the SLSP use case, statically positioned sensors are directly connected to a centralized decision-making server. Sensor mobility, positioning data and distributed decision making on the sensor side are not addressed in the SLSP design [5].

Correlation and co-operation of multiple sensor units are presented in multiple published surveillance system designs. In Hou et al. [6], data from a network of video cameras are correlated on a centralized server to, e.g., track objects and detect unauthorized moving events. A distributed approach to sensor unit co-operation is presented in Hou et al. [7], where video camera units collaborate autonomously to send commands and alerts to nearby units and personal mobile devices. The aforementioned designs and scenarios deal with statically deployed surveillance units, and do not consider sensor unit mobility. Several published surveillance designs mention exploitation of positioning information of mobile units [8–16]. Applications of positioning information include visualization of units on the field [12], autonomous navigation and coordinating groups of units [8,9,13,14] as well as surveillance coverage optimization [10]. While not directly concerning sensor unit mobility, multiple surveillance designs make use of location data for detection of events and/or objects [6,15–18]. The designs in [11,16] combine object detection and tracking with coordination of multiple mobile agents by utilizing groups of unmanned aerial vehicles (UAV) [11] and ground-based robots [16].

3 The SPY–VTT architecture for mobile intelligent surveillance

We present a software architecture for a distributed, IP-based surveillance of a physical area, enabling the use of both distributed and centrally coordinated intelligence, as well as the exploitation of surveillance unit mobility. The architecture, depicted in Fig. 1, consists of an arbitrary number of mobile units equipped with intelligent sensors, as well as a centralized control room, which reacts to alerts and manages the mobile units. The architecture is a part of a larger software framework called the Surveillance improved sYstem (SPY). The subsystem described in this article will be referred to as SPY–VTT.

The mobile units communicate with the control room side using wireless IP-based networks. Control room components may be connected either wirelessly, or using Ethernet, either on a local or a distributed network. Decision-making and alerting communication between the components is performed using data messages exchanged over transmission control protocol over internet protocol (TCP/IP). The components of the architecture are further detailed in the following subsection.

3.1 SPY–VTT architecture components

The mobile sensor unit (MSU) represents software on-board of a mobile unit deployed in the area under surveillance. The software is connected to one or more intelligent sensors, which provide high-level event data or metadata, such as “suspicious object detected in video”, instead of, or in addition to raw sensor data. Potential sensor types

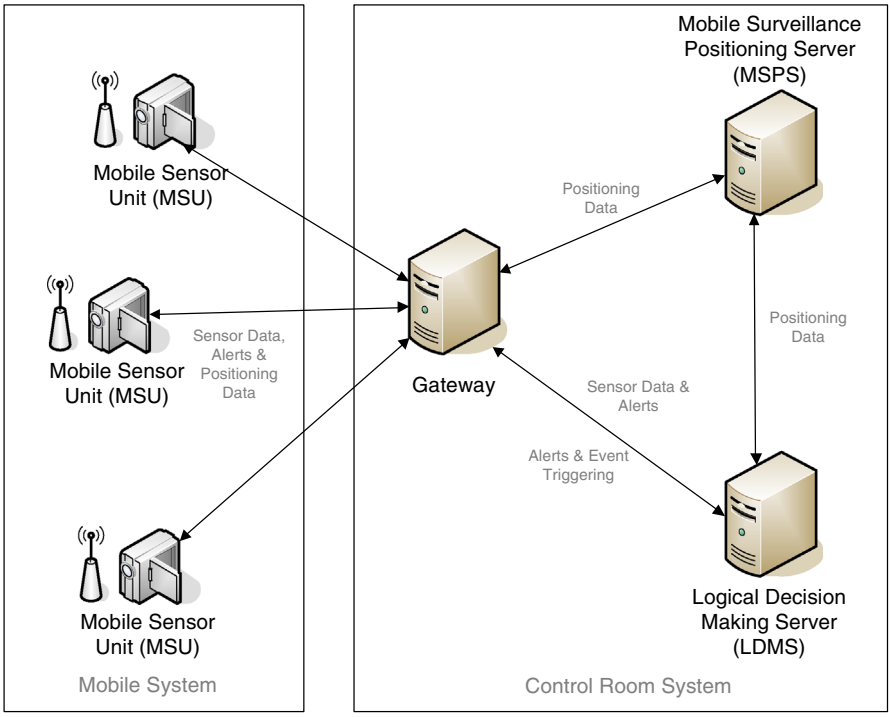

Fig. 1 SPY–VTT architecture for mobile intelligent surveillance

include, e.g., intelligent video cameras and audio sensors. Types and amount of sensors depend on the configuration of the MSU and the surveillance scenario in which the system is deployed. A global positioning satellite (GPS) sensor is required on each MSU for periodically reporting the unit's position to the control room.

The MSUs contain decision-making capabilities for processing event input from sensors. In the currently envisioned scenarios, internal MSU decision making is intended primarily for two purposes: (1) Filtering out redundant or benign events reported by sensors for saving wireless network bandwidth and control room processing time, and (2) Fusing together multiple atomic inputs from different sensors into single complex events. The MSU receives messages and commands from the control room components, as incidents reported by MSUs are automatically reacted on. Commands may consist of, e.g., pan/tilt/zoom (PTZ) or recording commands to sensors, while messages may contain, e.g., instructions to move towards an area where additional MSUs are needed. The architecture also allows MSUs to share data and communicate with each other through the control room. MSUs can represent multiple different types of surveillance units, and multi-type MSUs can cooperate within the same scenario. Possible MSU types include, e.g., manned units such as police cars, deployable unmanned wireless camera units, or personal devices equipped by patrolling surveillance personnel.

The *Gateway* manages the TCP/IP message connections between SPY–VTT components. MSUs send data to the control room components or each other through the

Gateway, which also routes responses and other messages to the appropriate MSUs. Messages are directed to their appropriate receivers by the Gateway using control room components with unique identification strings of the components. MSUs do not have to possess information of the physical addresses of different control room components when using the Gateway. Conversely, the control room components do not have to maintain connections to each of the MSUs currently online in the system.

The mobile surveillance positioning server (MSPS) processes, stores and distributes positioning data of the MSUs deployed on the field. As MSUs report their position periodically to the control room software, the MSPS stores the information into a database. The server provides a querying interface for retrieving the current and past MSU positioning data by LDMS and other possible components from the MSPS database. The LDMS may query, e.g., for the positioning information of a specific MSU, or multiple MSUs to track positions of MSUs currently within a certain distance of an event.

The logical decision-making server (LDMS) provides decision-making capabilities for automatically utilizing data from the MSU network. It is based on the non-mobile LDMS design within the SLSP system [5]. The LDMS receives event data from the MSUs on the field in real-time as messages containing high-level event data. Analysis may consist of the following types of deductions: (1) determining the seriousness of a reported event thereby deciding on whether there is need for further actions, and (2) correlating similar reports from multiple MSUs into a greater complex event, and deciding on a response based on this correlation. Based on event analysis, LDMS automatically performs actions depending on the scenario and the perceived threat level of the event reported. The actions may consist of, e.g., issuing commands to controllable sensors, sending alert messages to be displayed for surveillance personnel, or providing directions to a unit for relocating to the area of the perceived threat. LDMS also utilizes MSU positioning information it queries from MSPS. In currently envisioned scenarios, the positioning data can be exploited by LDMS automatically in two ways: (1) In case of an event requiring assistance from additional MSUs, LDMS can automatically select the most suitable MSUs, e.g., those closest or most quickly movable, to assist the original reporter of the event. (2) If multiple similar events are reported within a close proximity, the LDMS may deduct that a larger area of disturbance can be found, and can instruct other MSUs to cover the area in question.

3.2 Advantages of the SPY–VTT architecture

The use of LDMS together with data originating from the MSUs enables the SPY–VTT architecture to take advantage of location-specific information of actors working on the field in a wide-area surveillance scenario. The system can take in consideration the current position of units and sensors on the field, instead of relying on a network of static sensors as depicted in [6,7]. A number of prior designs primarily focus on supporting a specific type of mobile or deployable surveillance unit such as ground-based robots [8,10,13,16], UAVs [9,11] or video camera units [15,17]. The SPY–VTT architecture does not limit the types of MSUs used, and also allows coordinating combinations of

different MSU types as required by the surveillance scenario and environment. The scale of the geographical area to cover is also not limited by SPY–VTT, although in real-life implementation factors such as unit communication range [11] need to be addressed. In centralized decision making within the SPY–VTT architecture, MSU positioning data can be combined with location-specific event and object detection data from the MSUs in a similar fashion to [6,11,16–18]. This makes the SPY–VTT system aware of the situation of all entities involved in surveillance, namely (1) units, (2) sensors and (3) events or objects. The gathered situational information can be used by LDMS for various types of automated reacting via a centralized decision-making process, while previous designs have concentrated on, e.g., visualization [12] and navigation assistance [8,9,13,14].

The centralized decision-making approach makes it possible for the architecture to collect all the data produced by distributed entities utilized in surveillance as well as the events these multiple units have identified. This facilitates a broader situational awareness, which in turn enables processing and recognizing complex scenarios with access to all necessary information from the field. As a whole, the SPY–VTT architecture efficiently exploits the potential resources available in a distributed and mobile surveillance environment, by realizing the combination of key features in mobile surveillance: (1) distributed intelligence in surveillance units, (2) centralized decision making for handling complex events that require situational awareness from multiple sources together with coordination of multiple actors on the field, as well as (3) exploiting the positioning information of various surveillance entities on the field. This combination of distributed, centralized and mobility-aware operations enables the architecture to support various mobile surveillance scenarios, from locating unwanted events to coordinating response efforts among multiple geographically distributed sensors and personnel. Thus, the SPY–VTT architecture provides a comprehensive and adaptable solution for modern intelligent surveillance in mobile environments.

4 Introduction of a self-healing solution into the architecture

In this section, the need for a self-healing solution to integrate with the SPY–VTT architecture is analyzed. That results in a strong requirement which is fulfilled by the combination of the SPY–VTT architecture with an existing solution from the domain of autonomic computing/networking.

A traditional approach to resilience in systems engineering is to carefully dimension the system parameters, and add additional backup components, such that hardware and software problems do not immediately make the system deteriorate. A modern approach is to enable the self-healing of a system by introducing software-based solutions that can increase the robustness with respect to a wide range of software and even hardware problems. Next, we investigate how this approach can improve our proposed surveillance architecture. At the end of this section, the advantages of the proposed approach when it comes to ensuring the robustness of SPY–VTT are highlighted.

4.1 Self-X functions and autonomic control loops

During the past decade there was a significant push within the community towards the investigation and adoption of self-X features within the system architectures, e.g., self-configuration and self-healing. This included the notion of autonomic computing as introduced in a number of fundamental papers elucidating and defining key aspects and concepts [19–21]. In addition, concepts from the autonomic computing field are applied and investigated in various scopes within the IT and telecommunications world, such as for example in the scope of cloud computing and efficient virtual resources management [22,23].

The core idea towards the adoption of self-X behaviors is constituted by the introduction of agents inside the devices, which should have the capabilities to monitor and manage a set of resources with respect to different potential problems. The basic architecture for such an autonomic entity is illustrated in Fig. 2. This architecture identifies the steps within a generic control loop required for self-management as monitoring, analyze, plan, and execute, and it is widely known as the MAPE model. In that scope, a generic autonomic manager/element is responsible for a set of managed resources (element) and acquires information regarding their operation using different monitoring functions/probes, which are denoted as *Sensors* in Fig. 2. The monitoring aspect is illustrated by the *Monitor* step of the autonomic manager/element. Further, the monitored information is passed to the *Analyze* step that is responsible for correlating the different observations and understanding the situation of the managed resources/element. Given that the analyze step of the control loop has detected a problem in the operation of the managed resources, corresponding action(s) need(s) to be planned, which is the task of the *Plan* step in Fig. 2. Finally, the *Execute* step is in charge of executing the planned action(s) on the Managed Resources over a set of *Effector* interfaces. All the above described control loop phases may be differently implemented and may use various (information) models, to ensure smooth operation

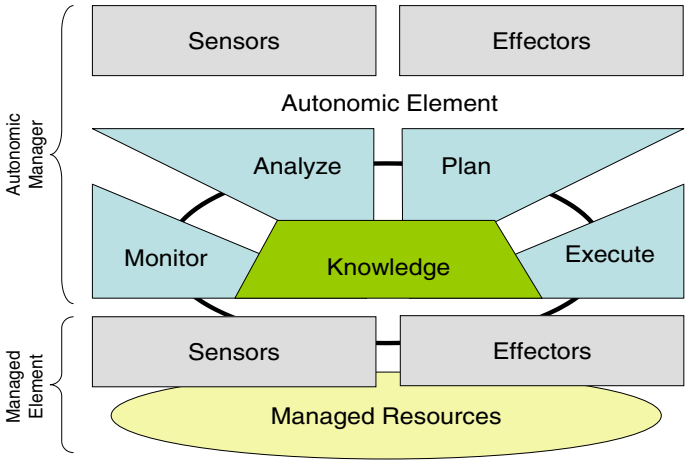

Fig. 2 The IBM-MAPE control loop [19] for autonomic computing

of the managed resources/element. These various models would contain vital information for the correct operation of the single steps and would in that way constitute a *Knowledge* base as depicted in the middle of Fig. 2. At the end, the autonomic manager/element itself might constitute a managed resource/element for an upper layer autonomic manager/element and, therefore, offers its own sensors and effectors interfaces. Based on this or similar [24,25] control loop structures, different self-X features are implementable.

4.2 The UAFAReS architecture for self-healing and its role within the SPY–VTT system

The unified architecture for autonomic fault-management, resilience and survivability (UAFAReS) [26] was developed in the course of different international projects. UAFAReS aims at specifying components and interfaces for the implementation of self-healing features within the involved devices. Related research in the area of self-healing network infrastructures is given by [27]. Trehan [27] investigates various mathematical tools for the fast reconfiguration and reestablishment of connectivity in self-healing networks. In addition, Sarma and Trehan [28] focus on different design principles for network infrastructures, which can be proven to enable the efficient restoration of end-to-end connectivity in the face of node and/or link failures. The work presented in [29] aims at introducing a terminology and a reference model for self-healing on top of diverse radio access technologies (RAT). Finally, Yuan et al. [30] investigates the implementation of self-healing mechanisms in wireless sensor networks. In comparison to these ideas, UAFAReS aims at realizing the processes of autonomic self-healing in a way that they can be easily configured to tackle different problems in the network devices and in the network as a whole. The basic idea is based on the insight that traditional fault-management processes as known from the telecommunication domain can be automated towards the realization of fault-removal and correspondingly self-healing in a distributed system. Traditionally, the telecommunication management network (TMN) standard [31] and related understanding views fault-management as consisting of fault-detection—*"detect the presence of a fault"*, fault-isolation—*"find the root cause(s) for the observed erroneous state"*, and fault-removal—*"remove the identified root cause(s)"*. In that context UAFAReS aims at realizing self-healing aspects by enabling the automation of these processes, and identifying additional processes which emerge in the course of that automation. Moreover, components are introduced that enable the immediate reaction to a faulty condition with the aim to sustain some basic system functionality. That way UAFAReS enables the realization of survivability aspects as known in telecommunication protocols such as SONET/SDH [32]. In addition, UAFAReS aims at providing hooks and APIs for implementing proactive resilient behaviors that aim at avoiding failures if such a trend is detected.

The UAFAReS components to be deployed inside a node within the SPY–VTT surveillance system are presented in Fig. 3. By a node we mean either an MSU on the field, the Gateway, the MSPS, the LDMS or other supporting machines, e.g., database servers. In addition, the overall process that is collaboratively executed by the UAFAReS components across multiple nodes is depicted in Fig. 4. Basically, within

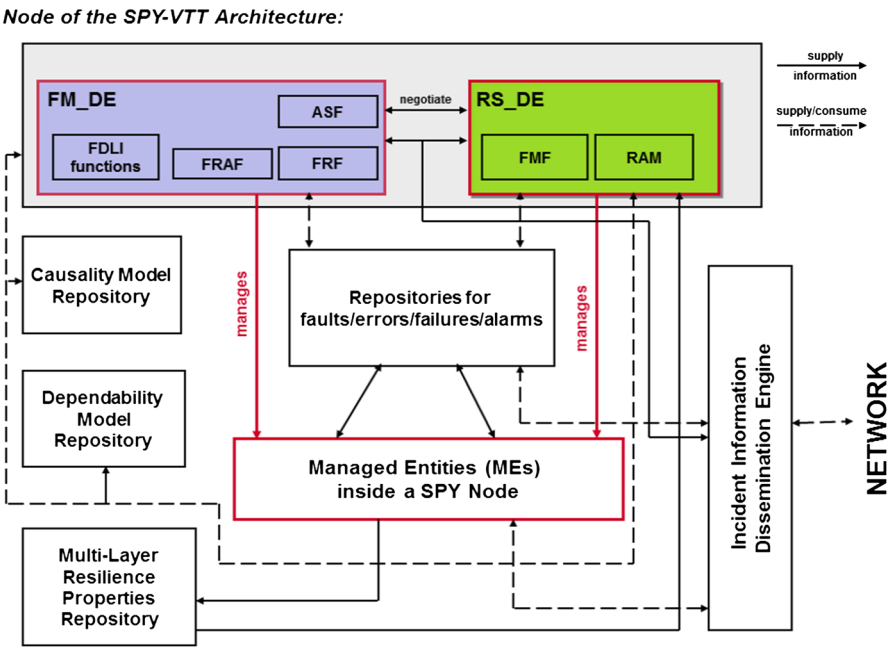

Fig. 3 The SPY–VTT–UAFAReS architecture based on [26]

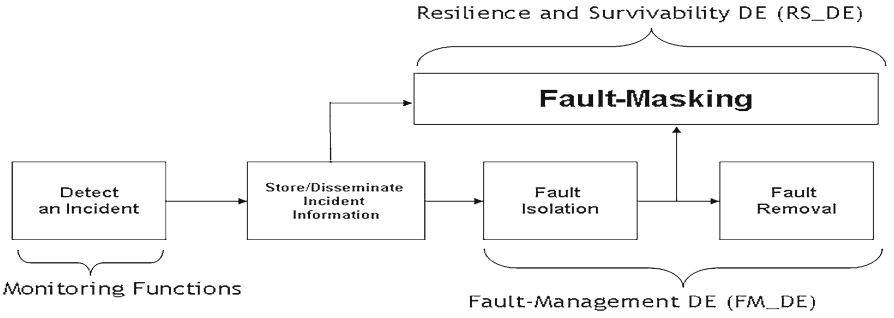

Fig. 4 The self-healing process according to [26]

a node, UAFAReS consists of two autonomic agents, decision elements (DEs) [25], which are responsible for realizing the self-healing function in tandem. In that context, the UAFAReS instances in the SPY–VTT nodes are responsible for collaboratively handling incidents related to the overall set of resources and functional entities in each SPY–VTT node. This overall set of resources and functional entities on node level is summarized as managed entities (MEs) inside a SPY Node on Fig. 3. To effectively handle incidents w.r.t these MEs, one DE is responsible for an immediate fault-masking reaction to a detected incident—this is the resilience and survivability DE (RS_DE), while the other fault-management DE (FM_DE) is in charge of identifying the root causes of the observed faulty conditions and removing these root causes in the long-term operation of the system (see Figs. 3, 4). That is, the overall self-healing process, as

illustrated in Fig. 4, consists of two parallel tasks: (1) one responsible for an immediate fault-masking reaction, and (2) another task responsible for isolating the root causes for the experienced faulty conditions and removing them in the long term.

Fault-detection within the SPY–VTT–UAFAReS architecture is performed either by a dedicated monitoring system (such as the one presented in [33]) or by monitoring tools that can instantiate the APIs of the UAFAReS framework. These APIs provide the possibility of pushing incident information into the UAFAReS faults/errors/failures/alarms repositories inside a node. The incident information is further disseminated to other nodes over the incident dissemination engine, and forwarded to both local decision elements—RS_DE and FM_DE. That way incident information across the distributed system is made available to all FM_DEs and RS_DEs operating inside the nodes.

The resilience and survivability DE (RS_DE) includes a fault-masking functions (FMF) block and a risk assessment function (RAM). FMF offer hooks for implementing immediate reactions to reported incidents, while RAM uses monitoring information and raises alarms, in case a failure is anticipated. In that case, the FMF should get active and mitigate the potential faulty condition. Thereby, the FMF implement an *event-condition-action* type of policies, which are executed by a policy engine. When responding to alarms and incidents the FMF processes may refer to the multi-layer resilience properties repository, to obtain information related to the intrinsic resilience properties of the underlying entities. For example, there might be a situation which can be resolved quickly by a protocol or application without the need for an immediate intervention. However, if the entity does not succeed after a period of time, then the FMF should react. Such related information, e.g., timeouts, is stored in the aforementioned multi-layer resilience properties repository.

The fault-management DE (FM_DE) [34] is responsible for implementing the automation of fault-detection, fault-isolation, and finally fault-removal in the long-term operation of the SPY–VTT system. Incidents that get reported to the FM_DE are correlated by the so-called fault-diagnosis/localization/isolation (FDLI) functions, to identify a root cause. Thereby, the FDLI functions use a causality/fault-propagation model that is stored in the belonging repository. For traversing such a fault-propagation, we use a Markov Chain based algorithm that we specified and analyzed in our previous work [35]. Thereby, we systematically infer the probability for particular root causes to be the explanation of the experienced symptoms by combining insights from the areas of probability theory, linear algebra and data structures' design. Once a set of root causes has been identified, it is passed to the fault-removal functions (FRF) that tend to remove them and improve the operational conditions of the SPY–VTT system. These FRF once again implement an *event-condition-action* type of policies. Thereby, they employ scripts or tools which were preconfigured on the device and are executed as a result of the policies evaluation determining how to react to the identified root causes. These scripts might consume information regarding the dependencies among functional entities stored in a dependability model repository. To avoid conflicting actions executed in parallel by the FM_DE and the RS_DE, the action synchronization functions (ASF) are put in place. This component should be referred by both DEs to synchronize and eventually allow or disallow their tentative actions. In the course of that, the ASF use a utility function-based approach [35–37] to synchronize

the tentative actions and their potential impacts on the system. Within this approach, an optimization problem is solved for the set of tentative actions to select the best actions that do not contradict and improve the overall health of the system. Finally, in case of an executed action, the fault-removal assessment functions (FRAF) check whether the root causes have been indeed removed, and if not escalate the situation to the SPY human operator. The FRAF operate once again based on an engine realizing *if-then-action* type of policies.

The UAFAReS architecture has been developed and implemented over the course of three years and applied to different case studies from the domain of telecommunications. Further information on the algorithms implemented in the above described components, as well as on the case studies, can be found in [35–39].

4.3 Advantages of the UAFAReS framework for ensuring the robustness of the SPY–VTT architecture

UAFAReS was selected as the appropriate self-healing framework due to various advantages over other available systems. First of all, UAFAReS is based on a generic model for autonomic networking named GANA (generic autonomic networking architecture) [25,40], which is currently being standardized at ETSI [41,42]. That means that UAFAReS would be easy to integrate with other autonomic entities based on the GANA reference model. In addition, UAFAReS comes with a stable implementation, which was worked out in the scope of the EU EFIPSANS project [43]. Finally, UAFAReS offers generic configuration interfaces that can be used to configure it as to address various types of problems within the SPY–VTT architecture. This provides a serious advantage compared to similar (but limited in their scope) frameworks such as UniFAFF[1] [44], Omega [45], and D_2R_2+DR[2] [46]. Thereby, Omega is limited to 3G networks. UniFAFF focuses on clean-slate networks, such as the Autonomic Network Architecture [47], and D_2R_2+DR provides a generic conceptual framework for Resilience, which is instantiated on case study basis.

5 Case studies

This section presents three case studies, based on discussions with officials and other actors in surveillance for public safety, with the goal to illustrate the interplay between the SPY–VTT mobile surveillance architecture and the UAFAReS architecture for autonomic self-healing. Thereby, the case studies show how the robustness of the surveillance system is increased, which in turn allows for better managing complex emergency situations and can support lifesaving operations. At the end, a summary of the case study and a discussion of the limitations of our approach are provided.

[1] Unified Framework for Implementing Autonomic Fault-Management and Failure-Detection for Self-Managing Networks.

[2] Defend, Detect, Remediate, Recover, plus Diagnose and Refine.

5.1 Overcoming gateway unavailability through autonomic self-healing

The surveillance scenario for our first case study is illustrated in Fig. 5. In addition to control room components, the scenario consists of five patrolling MSUs, labeled MSU A-E. MSUs A–C represent road vehicles, while MSUs D and E are UAVs equipped with cameras to provide an aerial view on the situation. In the scenario, MSU A detects an alarming event and reports it to the control room, which determines that additional MSU support is needed. Based on MSU positioning data, nearby patrolling units are instructed to assist. In this case, the road vehicle MSU C as well as UAVs D and E are requested to move to the place of the event. We presume that all MSUs are equipped with the software components of the SPY–VTT–UAFAReS architecture.

Operation of the SPY–VTT components involved in the scenario is detailed within the following sequence diagrams. The sequence diagram in Fig. 6 depicts MSU registration and continuous operations running at pre-defined intervals while the system is operational.

- Upon initialization, each MSU sends a registration message to the LDMS for logging into the SPY–VTT system (1).
- Once active, the MSU continuously updates its GPS positioning data to the MSPS with pre-defined intervals (2).
- LDMS continuously sends keep-alive messages to each active MSU to ensure their availability (3).

The steps specific to the case study scenario, performed after MSU registration, are depicted in Fig. 7.

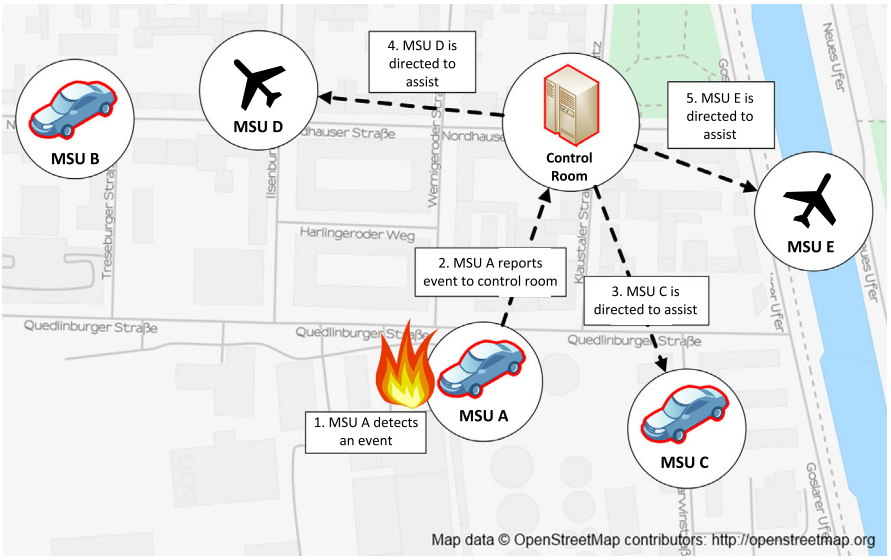

Fig. 5 First case study illustration

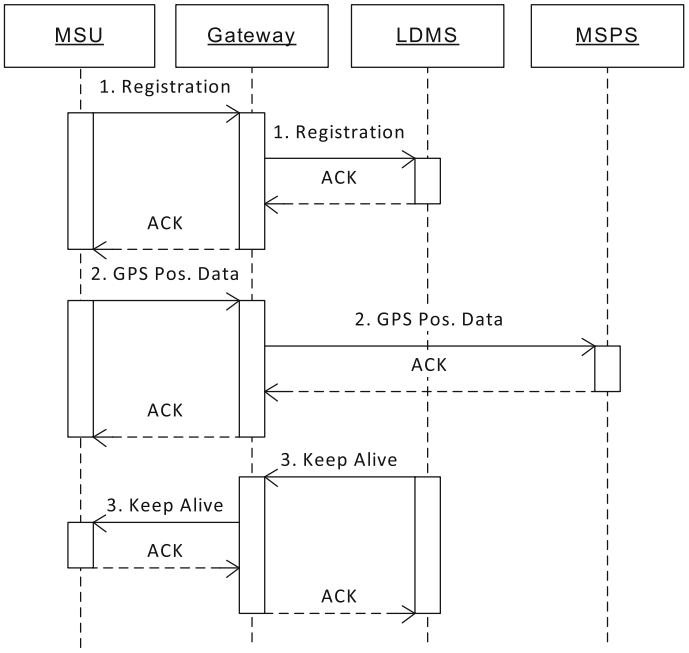

Fig. 6 Case study sequence diagram for continuous operations

- MSU A sends a message to the LDMS, reporting a sensor event determined to be a potential incident, in this case a fire (1).
- LDMS decides to instruct other MSUs to provide assistance. To choose the optimal assisting MSUs, LDMS uses the MSPS to retrieve the positions of MSUs within distance X to the coordinates MSU A, previously reported (2).
- LDMS sends out instructions to relocate to the event location to MSU C, which is determined to be in the vicinity of the reported event (3).
- LDMS informs MSU A of backup which has been sent to the area of the reported event (4).

Assuming a situation in which the burning facility is also the one hosting a base station, the MSUs will experience a loss of network connectivity to the Gateway and correspondingly to the LDMS and MSPS components. This would severely hamper the functioning of the SPY network and might even cause the loss of human life, since the rescue force would miss valuable information. The SPY–VTT–UAFAReS components must react in an attempt to restore the flow of messages and enable the transmission of events to relevant decision makers.

The reaction of the SPY–VTT–UAFAReS components consists of the following steps:

- Once the SPY–VTT software on the MSUs detects the lack of network connectivity to the Gateway, it reports this incident to the local repositories for faults/errors/failures/alarms.

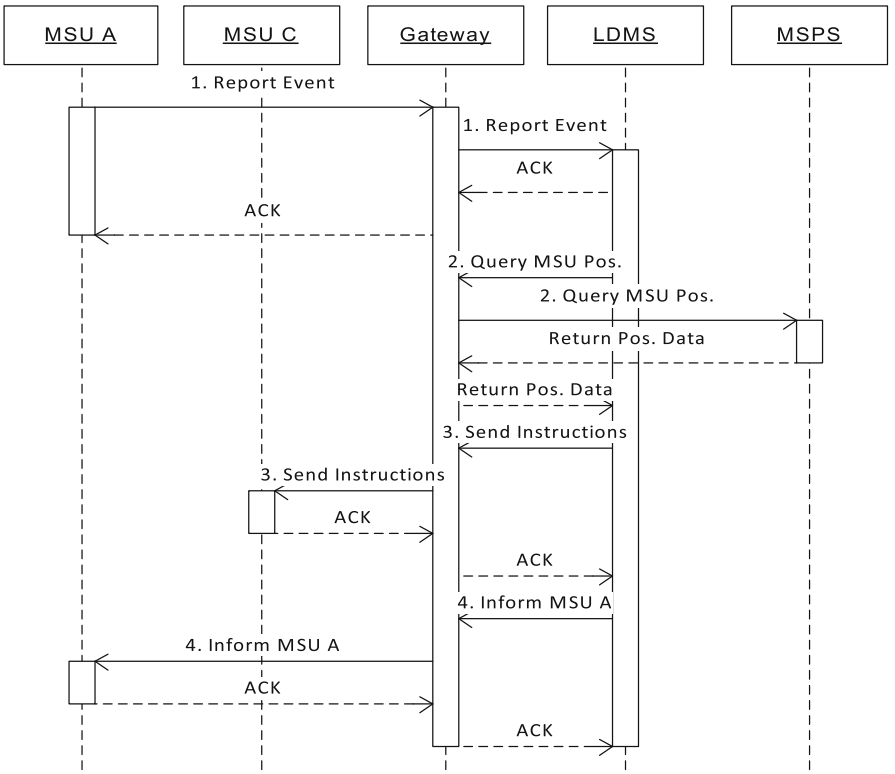

Fig. 7 Case study main sequence diagram

- The incident description is forwarded to the local fault-management decision element (FM_DE) and to the local Resilience & Survivability Decision Element (RS_DE).
- In addition, the incident description is submitted to the local Incident Dissemination Engine (IDE) of UAFAReS with the goal to convey it to the largest possible number of MSU nodes.
- The IDE broadcasts the incident description over the radio link and disseminates in that way to the other involved MSUs. Hence, after some time all the involved MSUs (A, C, D, and E) will have informed each other about their lack of connectivity to the Gateway.
- Based on this set of incident descriptions, the fault-isolation (FDLI) functions inside the FM_DE of each MSU conclude that there is a severe failure of the corresponding base station.
- To overcome this failure, the RS_DEs of each MSU are triggered to issue a proper reaction based on their embedded Fault-Masking Functions (FMF).
- The FMF of all road vehicles are preconfigured as to influence the network stack on the belonging MSU to stop sending information to the mobile network (LTE or UMTS), but instead to just broadcast it to other MSUs over the radio link.
- The FMF on all UAVs are preconfigured as to switch to a store-carry-forward (SCF) mode as discussed in [48,49]. In such a mode, the UAVs would receive and store

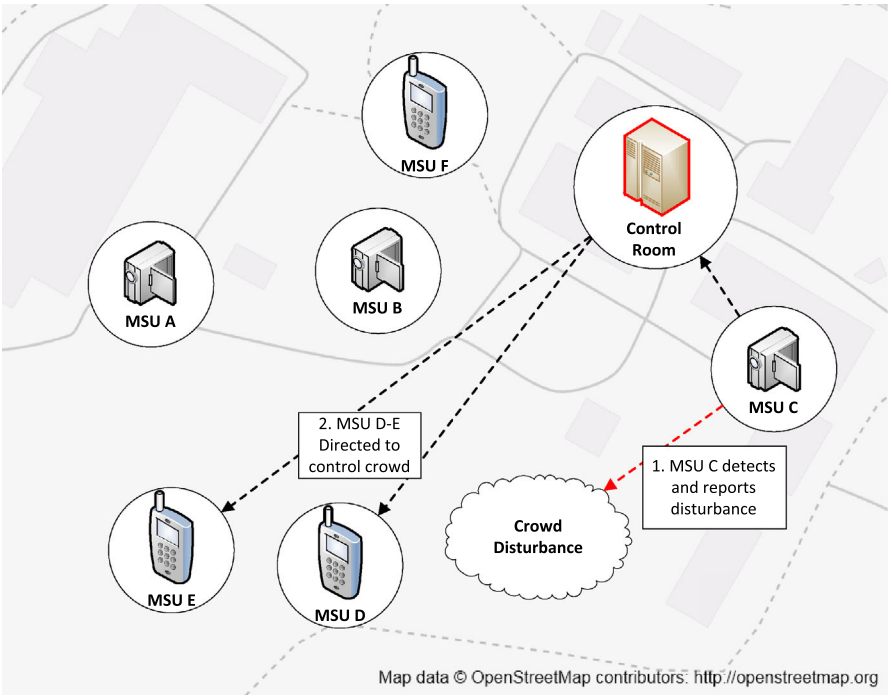

Fig. 8 Second case study illustration

a number of packets sent out by the road vehicles, carry those packets to the next functioning base station, and off-load them to the network towards the Gateway.

Steered by the SPY–VTT–UAFAReS components in each MSU, data flow is maintained from the MSUs to the Gateway and correspondingly to the LDMS and MSPS. Certainly, it is impossible to transmit continuous data in such a situation. However, in that context, the SCF method would facilitate the reporting of non-continuous data, such as new events and images, to the control room, thereby allowing relevant decision makers to better take action.

5.2 Adaptive surveillance for coping with network congestion

The second case study is illustrated in Fig. 8. In this case study, two types of MSUs are utilized to monitor a public event with large crowds: deployable static camera units with PTZ controls and image analysis capabilities (MSU A–C), as well as personal mobile devices on security personnel patrolling the area of interest (MSU D–F). MSU initialization and position reporting are identical to the first case study and Fig. 7. Furthermore, we presume that all involved nodes are equipped with the components of the SPY–VTT–UAFAReS framework. This includes the MSUs as well as the components of the control room illustrated in Fig. 8.

In this scenario, a disturbance is detected within the large crowds present in the area by image analysis performed on MSU B, and reported to the control room. The immediate reaction of the LDMS would be to alert the nearest security personnel patrolling the area, in this case MSU D and MSU E, to attempt and control the crowd in the area of the incident. However, given the large crowd and correspondingly the large number of mobile devices in that area, it is possible that the belonging LDMS messages fail to reach the MSU D and MSU E because of the resulting network congestion. This congestion could be caused by, e.g., overloaded UMTS/LTE base stations. The network congestion would especially get visible at the Gateway, where the LDMS messages are routed to the relevant MSUs. Hence, the communication stack on the Gateway will experience the failure to deliver the LDMS messages to MSU D and MSU E. In such a situation, the following steps will be executed by the components of the SPY–VTT–UAFAReS framework, to remediate the challenge:

- Once the communication stack on the Gateway detects the lack of reachability to MSU E and MSU D, it reports that to the UAFAReS repositories on the Gateway.
- Subsequently, the incident description is forwarded to the FM_DE and RS_DE as well as to the IDE, all of them within the Gateway.
- The IDE disseminates the incident information to all reachable components of the SPY–VTT architecture such that corresponding error messages can be shown on the monitors of the surveillance personnel.
- Still, the LDMS requires additional information regarding the crowd disturbance, to automatically improve its analysis of the events and aid the responsible persons in handling the complex situation.
- Therefore, the RS_DE on the Gateway would activate its FMF, which would implement a behavior so as to trigger the camera MSU that initially reported the disturbance, i.e., MSU C in Fig. 8, to switch its zooming capabilities and to continue reporting detailed zoomed images to the LDMS. That way, the quality and detail of the information provided to the LDMS would be increased, despite the network congestion problems in the involved communication stacks and links.

This second case study demonstrates how the combination of the SPY–VTT architecture with a framework for autonomic self-healing has the potential to facilitate the provisioning of the best possible data to the LDMS, despite challenges due to network congestion. This is especially crucial in emergency situations where the overall system would strive to supply the LDMS with high-quality information, to facilitate better data analysis in the LDMS and aid the corresponding decision makers, thereby supporting lifesaving operations.

5.3 Self-healing within the IT infrastructure of the control room system

The third case study is related to the IT infrastructure running in the control room system and the use of database technology in that context, as illustrated in Fig. 9. Referring back to the operating sequences depicted in Figs. 6 and 7, one can observe that there are regular interactions and queries directed to the MSPS, to handle the positions of different MSUs. All these communications to and from the MSPS imply

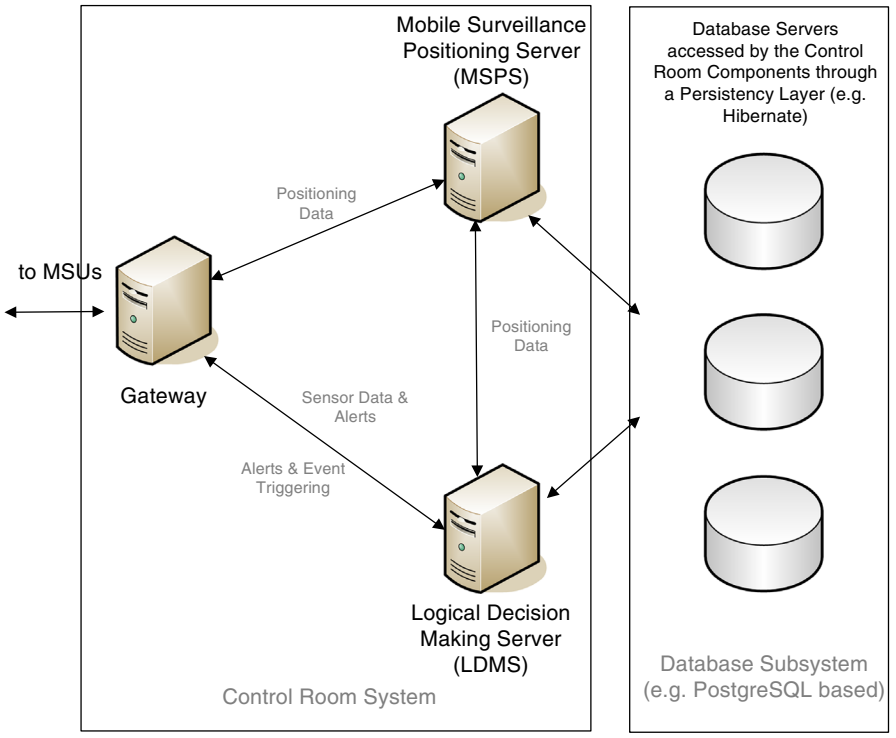

Fig. 9 SPY–VTT control room architecture with database subsystem

regular updates and queries issued on the database running in the background. We presume an implementation in which the MSPS uses a Hibernate [50] persistence layer to communicate with a PostgreSQL [51] database (see Fig. 9). Our experiences with such a setting have shown that a version mismatch[3] between the Hibernate database driver and the PostgreSQL database can lead to a problem in the closing of database connections. This problem might in turn sporadically lead to failures while opening new database connections, and correspondingly to problems querying and updating MSU positions within the MSPS. Assuming that the MSPS developers and operators are not aware of the version mismatch, they would only observe that database connections are sometimes not released resulting in issues while handling GPS data. To remediate this problem, the MSPS operators would use the SPY–VTT–UAFAReS autonomic self-healing capabilities and features. That is, a lightweight monitoring daemon would be instrumented on the MSPS node, to observe the number of open database connections. This daemon would interplay with the SPY–VTT–UAFAReS components of the MSPS node in the following way:

[3] It might be a difficult task to identify the version mismatch as responsible for the problem of database connection closing, since in general the entire set of database features function, and the failure to close database connections appears non-deterministically. We needed a couple of months and a large number of tests to diagnose such a problem in a prototype of ours.

- The database connections monitoring daemon would periodically check the number of open connections
- In case a pre-defined number of open connections are being approached, the daemon would create an alarm description, which would be pushed to the SPY–VTT–UAFAReS repositories on the MSPS node.
- The MSPS-node-IDE would in turn disseminate the alarm description to the involved machines and would correspondingly notify the SPY–VTT–UAFAReS components on the machine hosting the database.
- Subsequently, the alarm description would be forwarded to the FMF of the RS_DE on the machine hosting the database. These FMF are preconfigured as to react by automatically restarting the PostgreSQL database and that way releasing the large number of open connections.
- A restart is also required for the database communicating module of the MSPS, to release the connections which were opened on the MSPS node. Therefore, the FMF of the belonging RS_DE would automatically restart the MSPS server and trigger the release of all database connections.

By executing these steps, the MSPS services would be only shortly unavailable and would then continue handling GPS data from the MSUs. Hence, by introducing autonomic self-healing features into the SPY–VTT architecture, it is possible to remediate emergent erroneous situations in the IT infrastructure of the control room system, e.g., due to the unawareness of the MSPS operator for a version mismatch between a Hibernate driver and a PostgreSQL database.

6 Summary and discussion of the case studies

The case studies presented here are based on experiences with the SPY–VTT architecture and on much experimentation with the reaction mechanisms of the UAFAReS framework. Thereby, the case study related to the database management (PostgreSQL) is based on experiences from managing positions of mobile nodes in a project from the Smart Cities context, while the field case studies are based on theoretical considerations and discussions with actors from the public safety domain.

The detailed specification of the UAFAReS reactions within the scope of the case studies allows for discussing on the advantages brought by the proposed combination, and by its limitations in increasing the resilience of a SPY–VTT deployment. The clear advantage brought by the combination of SPY–VTT and UAFAReS is that the operation of the surveillance system can continue in the face of particular problems which have been captured in the models (Fault-Removal policies, Fault-Mitigation policies, Causality Model, etc.), based on which UAFAReS operates in the SPY–VTT nodes. However, the point that particular situations must have been directly or indirectly anticipated, and thus captured in the operational models, constitutes also a major limitation of the UAFAReS framework and its resilience support for SPY–VTT. In case of problems which were not identified as potential challenges, the UAFAReS components will not be able to react correctly and the SPY–VTT framework will fail to deliver its services for the relevant decision makers. In addition, UAFAReS is only able to overcome problems as long as there is a theoretical possibility for (partially)

restoring the service through the removal of a fault (root cause) or through the usage of alternative resources, e.g., in the case of deploying the UAVs to store-carry-forward content to the closest functioning base stations. In situations where this is not possible, UAFAReS would not be able to implement a resilient behavior, which in turn would mean that SPY–VTT would not be able to continue its operation.

To summarize, the case studies clearly show how the combination of the presented technologies can have significant impact on public safety. In addition, it must be clear that the proposed combination would improve the resilience, but not resolve all potential threats to the presented mobile surveillance system. However, the clear positive impact on public safety in general, invites to continue the research and development activities relating to the proposed combination of a mobile surveillance system and an autonomic framework for self-healing.

7 Conclusions and future work

In this article, we have presented our research related to the realization of a robust architecture for distributed mobile surveillance, event detection and automated reacting based on location-specific situational awareness of the mobile surveillance environment. We define components that realize distributed monitoring of an area, and extract events which are reported to a set of centralized components and reacted upon based on automated decision making capable of complex event processing. However, in a case of a catastrophic event, the surveillance system itself could be impaired by the effects of the event in question. To remediate this issue, we combine our proposed surveillance network with a self-healing framework that emerged in years of research around the topic of autonomic computing/networking. This solution also operates in a distributed manner inside the nodes of the surveillance system and matches the nature and requirements of our distributed surveillance architecture.

Our case studies demonstrate the benefits of a distributed and intelligent mobile wide-area surveillance system, as well as the benefits of combining such architecture with a distributed framework for autonomic resilience and self-healing. With respect to the pure SPY–VTT architecture, we see how the mobile nodes can support each other based on distributed event detection and the intelligence of the control room components. However, the case studies show that the surveillance system itself can be vulnerable to effects around the alarming event it is meant to provide information about. In such cases, it is imperative for the SPY–VTT architecture to deploy self-X mechanisms to self-organize in the face of the posed challenge. The case studies illustrate how the distributed reaction mechanisms can lead to a change in the behavior strategy of the MSUs for delivery of information to the control room. The functionalities presented in the scenario may thus be vital and lead to saving human lives in catastrophic situations. Based on our case study examples, we conclude that the combined SPY–VTT–UAFAReS architecture can provide a robust and fault-tolerant solution for distributed and intelligent mobile surveillance. With flexible usage of different entities' positioning information, a combination of distributed and centralized decision making, as well as abilities for overcoming technical failures, the architecture is suitable for versatile critical surveillance scenarios in mobile environments.

In future research regarding the SPY–VTT architecture, we will further evaluate the design based on our findings regarding the importance of self-healing in increasing reliability in critical surveillance scenarios. We intend to focus on various aspects such as scalability and robustness based on long-term simulations and experiments.

References

1. Räty T (2010) Survey on contemporary remote surveillance systems for public safety. IEEE Trans Syst Man Cybern Part C Appl Rev 40(5):493–515. doi:10.1109/TSMCC.2010.2042446
2. Moniak G et al (2007) Robust and high data rate wireless link for security between a bus and a control centre. In: Proceedings of the IEEE vehicular technology conference, VTC-2007, pp 1426–1430. doi:10.1109/VETECF.2007.304
3. Shet VD, Harwood D, Davis LS (2005) VidMAP: Video monitoring of activity with Prolog. In: Procedings of the IEEE conference on advanced video and signal based surveillance, AVSS 2005, pp 224–229. doi:10.1109/AVSS.2005.1577271
4. Huang C-M et al (2007) Enabling robust network fault tolerance for IP-based surveillance systems. In: Proceedings of the IEEE conference on consumer communications and networking, CCNC 2007, pp 39–43. doi:10.1109/CCNC.2007.15
5. Nieminen M, Räty T, Lindholm M (2009) Multi-sensor logical decision making in the single location surveillance point system. In: Proceeding of the fourth international conference on systems, ICONS '09, pp 86–90. doi:10.1109/ICONS.2009.18
6. Hou P-L, Yen C-S, Wu C-M, Tsai K-C (2009) True event decision by cooperative sensor network. In: Proceedings of the mobile data management: systems, services and middleware, MDM '09, pp 377–378. doi:10.1109/MDM.2009.60
7. Castanedo F, Patricio MA, Garcia J, Molina JM (2006) Extending surveillance systems capabilities using BDI cooperative sensor agents. In: Proceeding of the 4th ACM international workshop on video surveillance and sensor networks, VSSN '06, pp 131–138. doi:10.1145/1178782.1178802
8. Capezio F, Sgorbissa A, Zaccaria R (2005) GPS-based localization for a surveillance UGV in outdoor areas. In: Proceedings of the 5th international workshop on robot motion and control, RoMoCo '05, pp 157–162. doi:10.1109/ROMOCO.2005.201417
9. Winkler S, Buschmann M, Kruger L, Schulz H-W, Vorsmann P (2005) Multiple sensor fusion for autonomous mini and micro aerial vehicle navigation. Proc IEEE TENCON 2005:1–6. doi:10.1109/TENCON.2005.300916
10. Renzaglia A, Doitsidis L, Martinelli A, Kosmatopoulos EB (2010) Adaptive-based, scalable design for autonomous multi-robot surveillance. In: Proceedings of the 49th IEEE conference on decision and control (CDC), pp 4618–4624. doi:10.1109/CDC.2010.5717844
11. Hu J, Xie L, Lum K-Y, Xu J (2013) Multiagent information fusion and cooperative control in target search. IEEE Trans Control Syst Technol 21(4):1223–1235. doi:10.1109/TCST.2012.2198650
12. Labbe P, Lamont L, Ge Y, Li L (2007) Creating a dynamic picture of network participant geospatial information in complex terrains. In: Proceedings of the 2nd international conference on internet monitoring and protection, ICIMP 2007, pp 39–45. doi:10.1109/ICIMP.2007.13
13. Guillet A, Lenain R, Thuilot B (2013) Off-road path tracking of a fleet of WMR with adaptive and predictive control. In: Proceedings of the IEEE/RSJ international conference on intelligent robots and systems, IROS 2013, pp 2855–2861. doi:10.1109/IROS.2013.6696760
14. Masár M (2013) A biologically inspired swarm robot coordination algorithm for exploration and surveillance. In: Proceedings of the IEEE 17th internatinal conference on intelligent engineering systems, INES 2013, pp 271–275. doi:10.1109/INES.2013.6632825
15. Tomioka Y, Kitazawa H (2013) Collaborative patrol planning of mobile surveillance cameras for perfect observation of moving objects. Proceedings of the IEEE international conference on multimedia and Expo, ICME 2013:1–6. doi:10.1109/ICME.2013.6607583

16. Reid R, Cann A, Meiklejohn C, Poli L, Boeing A, Braunl T (2013) Cooperative multi-robot navigation, exploration, mapping and object detection with ROS. In: Proceedings of the 2013 IEEE intelligent vehicles symposium, pp 1083–1088. doi:10.1109/IVS.2013.6629610
17. Ekpar F (2008) A framework for intelligent video surveillance. In: In: Proceedings of the IEEE 8th international conference on computer and information technology workshops, pp 421–426. doi:10.1109/CIT.2008.Workshops.112
18. Miyaki T, Yamasaki T, Aizawa K (2007) Multi-sensor fusion tracking using visual information and Wi-Fi location estimation. In: 1st international conference on distributed smart cameras, ICDSC '07, pp 275–282. doi:10.1109/ICDSC.2007.4357534
19. Autonomic Computing: an architectural blueprint for autonomic computing (2006) IBM White Paper
20. Strassner J, Agoulmine N, Lehtihet E (2006) FOCALE: A novel autonomic networking architecture. In: Latin American autonomic computing symposium (LAACS)
21. Dobson S et al (2006) A survey of autonomic communications. ACM Trans Auton Adapt Syst (TAAS) 1(2):223–259. doi:10.1145/1186778.1186782
22. Soares MA, Madeira ERM (2012) A multi-agent architecture for autonomic management of virtual networks. In: 2012 IEEE network operations and management symposium (NOMS), pp 1183–1186. doi:10.1109/NOMS.2012.6212047
23. Ardagna D, Panicucci B, Trubian M, Zhang L (2012) Energy-aware autonomic resource allocation in multitier virtualized environments. IEEE Trans Serv Comput 5(1):2–19. doi:10.1109/TSC.2010.42
24. Hellerstein J, Diao Y, Parekh S, Tilbury D (2004) Feedback control of computing systems. Wiley-IEEE Press, New York
25. Chaparadza R (2008) Requirements for a generic autonomic network architecture (GANA), suitable for standardizable autonomic behavior specifications for diverse networking environments. International Engineering Consortium (IEC) Annu Rev Commun 61
26. Tcholtchev N, Chaparadza R (2010) Autonomic fault-management and resilience from the perspective of the network operation personnel. In: Proceedings of the GLOBECOM workshops, 2010 IEEE, pp 469–474. doi:10.1109/GLOCOMW.2010.5700364
27. Trehan A (2013) Algorithms for self-healing networks. CoRR abs/1305.4675
28. Sarma AD, Trehan A (2012) Edge-preserving self-healing: keeping network backbones densely connected. InL Proceedings of the 2012 IEEE Confernce on INFOCOM Workshops, pp 226–231. doi:10.1109/INFCOMW.2012.6193496
29. Barco R, Lazaro P, Munoz P (2012) A unified framework for self-healing in wireless networks. IEEE Commun Mag 50(12):134–142. doi:10.1109/MCOM.2012.6384463
30. Yuan S, Qiu L, Gao S, Tong Y, Yang W (2012) Providing self-healing ability for wireless sensor node by using reconfigurable hardware. Sens Basel Switz 12(11):14570–14591. doi:10.3390/s121114570
31. The FCAPS management framework: ITU-T Rec. M. 3400
32. Suwala G, Swallow G (2004) SONET/SDH-like resilience for IP networks: a survey of traffic protection mechanisms. IEEE Netw 18(2):20–25. doi:10.1109/MNET.2004.1276607
33. Tcholtchev N and Petre R (2012) Design, implementation and evaluation of a framework for adaptive monitoring in IP networks. In: Proceedings of the 9th IEEE international conference and workshops on the engineering of autonomic & autonomous systems (EASe 2012)
34. Wodczak M, Tcholtchev N, Vidalenc B, Li Y (2013) Design and evaluation of techniques for resilience and survivability of the routing node. Int J Adapt Resil Auton Syst (IJARAS) 4:36–63. doi:10.4018/ijaras.2013100103
35. Tcholtchev N, Cavalcante AB, Chaparadza R (2010) Scalable Markov chain based algorithm for fault-isolation in autonomic networks. In: Proceedings of the IEEE global telecommunications conference (GLOBECOM 2010), pp 1–6. doi:10.1109/GLOCOM.2010.5683163
36. Kastrinogiannis T et al (2010) Addressing stability in future autonomic networking. In: Procedings of the 2nd ICST conference on mobile networks and management, MONAMI 2010, pp 50–61. doi:10.1007/978-3-642-21444-8_5
37. Tcholtchev N, Schieferdecker I (2014) Framework for ensuring runtime stability of control loops in multi-agent networked environments. In: Gavrilova, M, Tan, CJK (eds) Lecture notes in computer science, vol 8360, Subseries: Transactions on computational science
38. Tcholtchev N et al (2009) Addressing stability of control-loops in the context of the GANA architecture: synchronization of actions and policies. In: Proc IWSOS 2009. LNCS, vol 5918, pp 262–268. doi:10.1007/978-3-642-10865-5_28

39. Tcholtchev N et al (2009) Towards a unified architecture for resilience, survivability and autonomic fault-management for self-managing networks. In: 2nd international workshop on monitoring adaptation and beyond MONA+ Workshop, Stockholm, Sept 2009. LNCS, vol 6275, pp 335–344. doi:10.1007/978-3-642-16132-2_32
40. Chaparadza R et al (2013) SDN Enablers in the ETSI AFI GANA reference model for autonomic management & control (emerging standard), and virtualization impact. In: Proceedings of the 5th IEEE international workshop on management of emerging networks and services (MENS 2013), collocated with Globecom 2013
41. ETSI Autonomic Future Internet ISG: http://portal.etsi.org/portal/server.pt/community/AFI/344. Accessed 24 Jan 2014
42. Chaparadza R., Wodczak M, Ben Meriem T, De Lutiis P, Tcholtchev N, Ciavaglia, L (2013) Standardization of resilience & survivability, and autonomic fault-management, in evolving and future networks: an ongoing initiative recently launched in ETSI. In: Proceedings of the 9th international conference on design of reliable communication, networks (DRCN), pp 331–341
43. Uzsak F, Simon C (2011) Monitoring system for EFIPSANS networks. In: Proceedings of the 34th international conference on telecommunications and signal processing (TSP), pp 47–51. doi:10.1109/TSP.2011.6043776
44. Chaparadza R (2008) UniFAFF: A unified framework for implementing autonomic fault-management and failure-detection for self-managing networks. Int J Netw Manag 19(4):271–290. doi:10.1002/nem.702
45. Baliosian J, Matusikova K, Quinn K, Stadler R (2008) Policy-based self-healing for radio access networks. In: Proceedings of the IEEE network operations and management symposium (NOMS 2008), pp 1007–1010. doi:10.1109/NOMS.2008.4575269
46. Sterbenz JPG et al (2010) Resilience and survivability in communication networks: strategies, principles, and survey of disciplines. Comput Netw 54(8):1245–1265. doi:10.1016/j.comnet.2010.03.005
47. Bouabene G, Jelger C, Tschudin C, Schmid S, Keller A, May M (2010) The autonomic network architecture (ANA). IEEE J Sel Areas Commun 28(1):4–14. doi:10.1109/JSAC.2010.100102
48. Fall K (2003) A delay-tolerant network architecture for challenged Internets. In: SIGCOMM '03: Proc conf applications, technologies, architectures, and protocols for computer communications, ACM, pp 27–34. doi:10.1145/863955.863960
49. Sterbenz JPG et al (2010) Resilience and survivability in communication networks: strategies, principles, and survey of disciplines. Comput Netw 54(8):1245–1265. doi:10.1016/j.comnet.2010.03.005
50. Hibernate persistence framework: http://www.hibernate.org/. Accessed 15 June 2013
51. PostgreSQL database: http://www.postgresql.org/. Accessed 15 June 2013