Helicopter-Based Wildfire Monitoring System Software Architecture

Enric Pastor, Marc Sole, Juan Lopez, Pablo Royo and Cristina Barrado
ICARUS Research Group
Technical University of Catalonia
C/ Esteve Terradas, 7
08860 Castelldefels (Barcelona), Spain
enric@ac.upc.edu

Abstract—This work introduces a flexible and reusable architecture designed to facilitate the development of remote sensing applications. Based on it, we are developing a helicopter system, called Red-Eye, devoted to the detection, control and analysis of wild land forest fires in the Mediterranean area. The design of the proposed system is composed of five main components. Each component will work collaboratively to constitute a platform of high added value. The general architecture designed for wildfire monitoring is being tailored for two relevant objectives within the particular Mediterranean scenario: tactical day/night fire front evolution, and post-fire hot-spot detection.

The generalized integration of monitoring vehicles with the operation of other aerial resources (attack helicopter and airplanes) is an unsolved problem, both technically and methodologically. However, the operation of a monitoring helicopter during certain very well identified phases of the extinction process is highly plausible. High-level strategic monitoring is possible because no-conflicts will appear due to different flight levels; however, low-level tactical monitoring is a source of potentially dangerous in-flight traffic conflicts.

In addition to monitoring the fire front evolution, detection of remaining post-fire hot-spots located at the perimeter of fire is the main application that has been foreseen. Just after a fire front is contained or even during the days following the fire extinction, the monitoring tasks have to be maintained because of the danger of fire reactivation. The cost of monitoring with ground teams or aerial means it is very expensive and consumes resources needed in other fronts or possibly concurrent fires. However, a helicopter equipped with thermal cameras, can flight over the area and generate a map of hot spots with higher precision at a smaller cost. It is crucial in this application that the hot spots are reported immediately to avoid having ground brigades waiting for data for too long. Also, it is important to feedback the information in such a way it can be effectively consumed, not forcing ground brigades to walk around the forest without a clear operational scheme.

Based on a Service Oriented Architecture (SOA) [1] a heli-

borne hot-spot detection system is being developed. This paper describes the global architecture of the system, including the air segment, the ground control segment, and the interface with the squads operating on the fire area, etc. We also demonstrate how the available pre-defined modules in the SOA architecture have been reused to design this particular application, the additional subsystems required to implement specific hot-spot mission requirements, and the overall system / end-user interface.

TABLE OF CONTENTS

1 INTRODUCTION	1
2 INTRODUCTION TO THE RED-EYE SYSTEM.....	2
3 OPERATIONAL AND HMI OVERVIEW	5
4 SOFTWARE ARCHITECTURE.....	7
5 UNDERLAYING MAREA MIDDLEWARE	9
6 GROUND SEGMENT AND MANET OPERATION ..	14
7 CONCLUSIONS AND FUTURE WORK	16
ACKNOWLEDGEMENTS	16
REFERENCES	16
BIOGRAPHY.....	17

1. INTRODUCTION

Forest fires are an important problem for many countries. The economical loss is the most visible impact in the short term. The ecological damage and the impact on the wild life diversity and climate change are the most important lost in the long term. In some dramatic cases, we should add the loss of human lives directly caused by forest fire. Between 200.000–600.000 hectares (ha) are burnt every year in Europe; 80% corresponds to Mediterranean forests with high environmental value. In previous work we introduced a helicopter system, called Red-Eye, devoted to the detection, control and analysis of wild land forest fires in the Mediterranean area. The design of the proposed system is composed of a number of components, each one working collaboratively to constitute a platform of high added value. The system is designed for wildfire monitoring is being tailored for two relevant objectives within the particular Mediterranean scenario: tactical day/night fire front evolution, and post-fire hot-spot detection.

The Red-Eye system is a tactical hot-spot detection platform

¹ 978-1-4244-3888-4/10/\$25.00 ©2010 IEEE.

² IEEEAC Paper #1475, Version 1, Updated 30/10/2009.

designed to complement both the satellite based systems like those based on the MODIS (Moderate-resolution Imaging Spectroradiometer) sensor on-board the Terra satellite[2], [3], [4]; and the aircraft-based strategical monitoring platforms like the one managed by the USFD [5], [6], [7], [8].

This paper will introduce the service-based distributed architecture used to implement the Red-Eye system. Red-Eye is built around an embedded GIS (Geographic Information Systems) which provides geographically organized information integration, designed for capturing, storing, manipulating, analyzing and deploying all the information geographically referenced in order to facilitate system development. The system is build around a service-oriented architecture (SOA) allowing flexibility, reusability and scalability. The communication between all the services takes place using MAREA (Middleware Architecture for Remote Embedded Applications).

Most part of the graphical interface implementation has been developed using a specific open source library SharpMap, and it has been used C as programming language. The SharpMap library includes objects, methods and controls all necessary elements to develop embedded geographical information systems and map visualization. Orthophoto maps and other visual layers are managed by a GeoServer map service running in the same computer or in an attached one inside the same system.

Figure 1. Complexity of the wildfire scenario in the Mediterranean forest area.

The main elements composing the Red-Eye architecture are the positioning system GPS+INS, two interface screens (one to direct the pilot and a touch screen for the operator), a number of embedded computers and a combination of high resolution visual cameras and thermal cameras. All these elements will be on board the monitoring helicopter and the pilot will be guided by the system, which will drive the exploration of the areas while capturing images automatically. These images will be processed on-board to detect wildfire hot-spots and the results reported to the operator for final discrimination.

Red-Eye also includes a communication module to transmit selected information on detected hot-spots to squads operating on the ground. These squads are accessible through the creation a Mobile Ad-hoc Network between the Red-Eye system on-board the helicopters and mobile computer systems on the ground. Thanks to the middleware, those computer systems are seen as additional nodes in the distributed architecture; thus, information can be freely exchanged between those systems.

The paper is organized as follows. Section 2 generally describes the components of the Red-Eye system, its user interfaces, computation modules, sensors, communication elements, etc. Section 3 describes the operational mechanisms and HMI interfaces. The software service-based architecture of the Red-Eye system, including both the aerial and the ground segments are described in Section 4. The overall software architecture works through a service-based specific middleware called MAREA. Details of the capabilities and operation of MAREA are described in Section 5. Section 6 shows that the Red-Eye architecture not only operates on the air, with sensors and operational control on-board the system helicopter, but also on the ground with a networked set of squad information terminals operating as a MANET. Finally, Section 7 concludes the paper and outlines future research and development directions.

2. INTRODUCTION TO THE RED-EYE SYSTEM

The proposed Red-Eye system is designed to improve the overall awareness of the fire managers by providing tactical support to wildfire monitoring and control of ground squads [9]. Red-Eye is built around existing COTS technology that can be immediately deployed on the field on-board traditional helicopters at a reasonable cost. Red-Eye is designed to operate from a human on-board the helicopter, and its operation can be integrated in the standard operation of aerial resources. Information is gathered by the on-board sensors, processes and then displayed in such a way it can be immediately exploited by the operator in the helicopter or relayed to the ground controller and operative squads.

Current technology used to collect information on the operational situation on the ground prevents that the fire fighter coordinator to choose the right decisions on time. An aerial platform capable of over flying the area of a forest fire for long-enough periods of time and with communication capacity to link the gathered information with the firefighter squads operating from the ground would be an extremely valuable information gathering asset. The design of the RedEye system pretends to offer the following advantages:

- Payload can acquire coherent data even with bad weather or environmental conditions, like smoke.
- Can increase fire manager access to ubiquitous information and help to take decisions.
- It can also prevent dangerous situations for on-ground operators that can evolve very quickly.

- It can provide on-the-fly communication links between fire managers and the ground teams, especially in mountain or non-covered areas.

The above challenges are not critical to manage from a helicopter with the required technology on board. The project Red Eye proposes to develop a helicopter-based platform that will fly over the area of surveillance as long as needed and with the capacity to operate and to adapt to multiple demanding data collection scenarios thanks to its embedded computer systems.

The design of the proposed Red Eye system is composed of five main components (see Figure 2). Each component will work collaboratively to constitute a platform of high added value:

- Manned helicopter equipped with a sensor platform and embedded control system.
- Airborne command and control station.
- Mobile ground command and control station.
- Ground squad information terminals.
- Air/ground communication infrastructure.

A helicopter and its on-board sensor and computer system.

The fundamental elements of the system are the set of sensors (visible cameras, infrared and multi-spectral sensors). Their information will be collected and partially processed by the on-board computer system. Selected information will be partially transferred to ground for analysis. The overall system will be managed by an on-board operator.

Airborne command and control station.

On-board the helicopter and connected to the sensors, a command station will be operated by an operator in charge of the data-gathering process in order to analyze data and take the appropriate decisions. The controller will be able to decide which hot-spot points are more relevant and assign ground squads to deal with them.

A mobile ground command and control station.

The fire manager working on the ground in order to direct the overall operations can get real-time information from the aerial segment. The system also allows transferring some of the decisions to the ground command, like determining if certain hot-spots are relevant or which ground teams should be directed to supervise a certain hot-spot.

Ground squad information terminals.

A set of PDA clients carried by fire brigades will be integrated into a wireless network formed by the UAS and the MGCS. These PDAs will receive and present information from the MGCS (which could include messages, pictures, videos, maps, alarms, commands). They can also transmit data from a set of attached sensors to the MGCS, which includes: GPS position of the brigade, pictures, video, messages, am-

bient temperature and humidity, etc. The position of all information terminals is integrated in the mission interfaces offered by the MGCS to the fire operations coordinator.

The communication infrastructure.

It is mixture of communication connections that must guarantee the continuous contact of the helicopter with the ground command and control station. In addition, a secondary communication infrastructure transforms the Red Eye helicopter into a communication router between grounds squads and the ground command and control.

On-board Red Eye .

A prototype of the Red Eye system is being built around a helicopter operated by the Catalonia government. The selected helicopter is the Eurocopter AS350. Two units of this helicopter model (a B2 and a B3) are being operated for fire monitoring and search and rescue operations), and have been mostly selected by their availability. Figure 3 depicts a couple of images of the AS350-B2.

The air segment of the Red Eye system is composed of a set of cameras, computing and communications devices (see Figure 4). Sensing is implemented by a set of fixed mounted cameras that are calibrated to explore down to the flight direction of the helicopter. Three cameras are simultaneously operated, each one gathering data with a different objective:

FLIR A320:

- Thermal Imaging.
- Resolution: 16 bit 320x240 pixels.
- Frame rate: 9Hz.
- Connection: Ethernet 100Mbps.

Lumenera LE11059

- High resolution image.
- Resolution: 4000 X 2656 pixels.
- Frame rate: 3 fps at 4000x2656; 8.5 fps at 4000x640.
- Light sensitivity: 0.1 lux
- Connection: Ethernet 100Mbps

Lumenera LE350C_DN

- Medium resolution standard image/near infrared.
- Resolution: 2048x1536 pixels.
- Frame rate: 10 fps at 2048x1536; 40 fps at 1024x768.
- Connection: Ethernet 100Mbps.

Images will be acquired by an on-board computer system that will store the images on a system hard disk. Visible images are stored for reference, while the thermal image is processed to identify potential hot-spots. Initial filters separate those images that do not contain significant thermal information from those that contain potential hot-spots. If a thermal signature is detected then hot-spots are identified inside the image and

Figure 2. Overall architecture of the Red-Eye system: air and ground segments.

Figure 3. Eurocopter AS350 operated by the Departament d'Interior, Generalitat de Catalunya for Fire Monitoring and SAR operations.

correlated to other images taken immediately after the potential hot-spot image. Potential hot-spot images are stored and offered to the on-board operator.

The equipment on the helicopter also contains an IMU plus GPS that provides precise navigation and real-time orientation of the platform. This information will be stored correlated to the image acquisition process. Images will be only analyzed and geo-allocated if significant thermal signatures are detected. The image processing is not done in real time, but queued to be executed as soon as possible with the available CPU capacity.

Computation capabilities are implemented through fan-less embedded computer VIA EPIA ME6000G. Its main characteristics are:

- CPU: 600MHz Fanless VIA Eden Processor, selected in order to be able to operate without additional cooling systems..
- VGA: Integrated VIA UniChrome AGP graphics with MPEG-2 decoding acceleration. Two display capacity to provide monitor for both Red Eye controller and pilot.
- 10/100 Mbps Ethernet to have connectivity with the on board router subsystem.
- Two serial COM ports for IMU/GPS communication.

Internal communications between cameras and the computation module is implemented through an on-board Router-BOARD RB532A. Its main characteristics are:

- CPU: MIPS32 4Kc based 266MHz embedded processor.
- Memory: 64MB DDR onboard memory chip.
- 3 x 10/100Mbps Ethernet.
- 2 x MiniPCI for Wifi and WiMax links.

Figure 4. Airborne segment in the Red Eye system: components and 3D schematic view of the prototype integration.

The Red Eye system will continuously transmit to the ground the basic information on the position of the helicopter and the pictures that have been taken (only the position). If thermal signatures are detected, the application will identify those pictures and schedule them to be transferred to the ground if requested. All other pictures will be stored for post-mission analysis.

Screens and interfaces are also present in order to implement a control application and to provide visual cues to the helicopter pilot to guarantee that the flight path corresponds as much as possible to the one that better fits the area exploration underway. The main control screen is designed to offer access to the monitoring procedures, the monitoring flight-paths, the acquired images, the position of the ground teams, etc. This screen offer both touch screen interface and a joystick for precise pointer positioning.

A second small screen is placed at the pilot position providing him immediate guidance in terms of correct altitude, heading, lateral displacements, and initial and final points with respect the monitoring flight paths. This screen is just informative and its information will depend on the operational mode that has been selected by the controller.

3. OPERATIONAL AND HMI OVERVIEW

The designed command and control systems software on board the helicopter and on the ground are almost identical. The main difference between them is the latency required to get the image information. On board the helicopter images can be reviewed as soon as acquired (although hot-spot detection will require certain computation time). On the ground, images can be seen under request and the latency of the transmission will be much longer and will depend on the distance from the helicopter to the ground control.

The operation of the command and control software is divided in five different phases, namely:

- System configuration.
- Definition of exploration areas.
- Hot-spot exploration.
- Hot-spot confirmation.
- Ground squad task assignment.

System configuration.

The system configuration screen allows configuring the system in order to define the preferred exploration altitudes and speeds according to the desired resolution in each phase of the operation (exploration/confirmation) according to the available combination of camera/optics.

Information indicating the location of the ground control station, landing sites, identification of the available ground squads can be also specified.

Finally, restricted areas can also be specified to indicate population areas not to be directly over flown and/or especially dangerous obstacles like towers or electric lines.

Definition of exploration areas.

The areas to be explored need to be specified by the operator by means of a closed polygon. Each polygon implies, at least, a separated exploration by the helicopter.

The Red Eye system automatically offers an exploration solution defined by a number of trapezoidal areas to be scanned through linear tracks over flown by the helicopter at the specified distance to the ground. The particular altitude to be flown in each pass will depend on the variations of the terrain elevation (available through a Digital Elevation Model of the area).

Hot-spot exploration.

Once the exploration scans have been properly defined, the scanning process itself can be started. The system automatically generates Initial Scan Points (ISP) and Final Scan Points (FSP) to guide the task of the pilot. The ISP allows the pilot

Figure 5. Hot spot detection during the area scanning process. Images can be analyzed as soon as available in order to confirm/discard the detection.

to properly align the vehicle with the exploration track, while the FSP provides room for the turning maneuver and the realignment with the following exploration track.

During the each exploration track the Red Eye cameras are automatically activated in order to gather images at the required frame rate (that will depend on the camera/optics, the selected altitude and the desired frame overlapping between consecutive frames). Each frame acquisition is indicated on the screen as a green dot. Frame acquisition stops once the FSP is surpassed and re-stars as soon as the helicopter gets into the next ISP.

While the exploration is being completed thermal images are being analyzed in order to determine if they have a significant thermal signature. In that case the corresponding dots may turn into red or yellow, depending on the intensity of the thermal signature. Additionally, the source of the hot spot is identified inside the image and then geo-referenced according to the information provided by the IMU and GPS at the time of the acquisition. The analyzed information is offered to the controller in order to confirm if it is a real hot-spot or maybe stray heat reflections from some element on the ground (rocks, human-made devices, etc.). Thermal images can be over imposed on top of high-definition images in order to facilitate the identification of the hot-spot.

Explorations are selected at users discretion and can be repeated if necessary and/or interleaved with the confirmation phase.

Hot-spot confirmation.

After an exploration is being completed some detected hot-spots may be still undecided. The Red Eye system offers support to design a trajectory that overflies all undecided hot-spots in order to analyze them at a lower altitude and/or during a hovering flight. Having a closer look on the potential hot-spot should help deciding its relevance.

Hot-spots are processed sequentially following the sequence suggested by the application. Confirmation sequences can be repeated after the ground squads have refreshed the area. In that way, detected hot-spots can be revisited to determine if the extinguishing task have been successful.

Ground squad task assignment.

Given the set of confirmed hot-spots, the controller can assign them to the available ground squads working on the area. Ground squads will receive all information available on the hot-spot like its position, thermal and visible images and any available detection information.

Once the squad has identified the hot spot and worked on it,

they will communicate the controller through the application that the spot has been processed. During this process, the controller will be able to follow the progression of the teams on its screen.

4. SOFTWARE ARCHITECTURE

The active elements in the overall Red-Eye architecture have been already described in Section 2. The proposed software architecture will be composed of five main components. Each component will work collaboratively to constitute a platform of high added value to the people in charge of surveillance and to help to make their task more accurately.

Red-Eye Architecture Components

The components of the proposed Red-Eye solution are:

- Sensor platforms and embedded control system on-board the helicopter.
- Mobile ground control station.
- Ground squad information terminal
- Air/ground communication infrastructure.

Figure 6 depicts the general application scenario in which a Red-Eye helicopter monitors an active fire being directly linked to the ground control station. In parallel to this direct air-ground communication, an additional Mobile Ad-hoc Network (MANET) connects all ground squad terminals in a highly dynamic structure. Two different types of nodes can be found, personal terminals operated by firefighters on the move and vehicle mounted terminals that offer improved link strength. Complementing this infrastructure, both the Red-Eye helicopter and the ground control station can become members of this MANET network. Specially relevant is the role of the aerial vehicle because it may provide communication to separated squad that may be out of range through ground links.

Sensor platforms and embedded control—The helicopter platform and sensor set is selected according to the targeted “Mediterranean” area and cost objectives. A medium altitude tactic helicopter is preferred for cost availability. Basically two cameras need to be installed on-board: a high definition visual camera and a thermal camera configured for temperature ranges according to the precise mission to be implemented. Multi-spectrum thermal sensors may also be desirable but its availability, cost and possible integration in a tactic helicopter is uncertain.

In addition to the INS/GPS-navigation subsystem, the avionics in the helicopter should include hardware support to execute all services required to properly manage the monitoring mission: sensor management, data storage, communications, image processing, flight-plan management and mission management. All those services are internally networked through the MAREA middleware running on top an Ethernet switch or router.

All data acquired by the sensor package will be internally stored and partially processed in real-time on board the helicopter. Selected telemetry and sensor information will be scheduled to be transferred to the ground control for more detailed analysis. All data can be later on transferred to the ground when the helicopter gets back on the ground or during flying periods in which no other high-priority data is scheduled to be transferred.

Mobile ground control station—The ground teams working on the surveillance area should have equipment available to directly track the helicopter operation in real time. Additionally, immediate “fire”-related information should be made available to this control station so that fire operations coordinators can use it regardless of further distribution of the information to other control centers.

This is the goal of the Mobile ground control station (MGCS). The objective is to be able to control the system close to the area of actuation, relaying on line-of-sight communications (due to the cost and complexity of satellite communications). Helicopter location and status, actual state of the mission operation and the initially obtained surveillance data will be received there, although other data sources in addition to the Red-Eye itself may be available. The MGCS will offer a basic graphical GIS interface for quasi real-time management of data and for mission re-tasking according to the fire evolution. Hot-spot management as well as fire-front detection and evolution could be completed here including a true orthonormalization of the overall information.

Ground squad information terminal—A set of PDA clients carried by fire brigades will be integrated into a wireless network formed by the Red Eye helicopter and the MGCS. These PDAs will receive and present information from the MGCS, which could include messages, pictures, videos, maps, alarms, commands. They can also transmit data from a set of attached sensors to the MGCS, which includes: GPS position of the brigade, pictures, video, messages, ambient temperature and humidity, etc. The position of all information terminals is integrated in the mission interfaces offered by the MGCS to the fire operations coordinator.

Air/ground communication infrastructure—The availability of a reliable communication connection between the Red-Eye helicopter and the MGCS, and eventually with the Squad Information Terminals is critical. However, the existence of a mixture of communication connections should not negatively interfere with the development of the various mission services involved in the application. The MAREA middleware will properly manage these connections according to the priority of the information to exchange, the available bandwidth, and the cost of each connection.

The wildfire monitoring system will initially rely on a variety of commercially communication subsystems; namely the embedded radio modem and a wifi/wimax module for high

Figure 6. General view of the participating modules in the SOA-Based Red-Eye architecture.

bandwidth data transmission. A GPRS/UMTS OEM system and a spread spectrum commercial radio modem will be incorporated in the future. Eventually, a commercial satellite link can be added depending on the budget availability.

Transmission rules will limit full resolution image exchange to the wifi/wimax link. Telemetry and Red Eye control will have the highest priority among all other information flows. Mission control will manage the acquired sensor information, those tagged as highly relevant will be scheduled for transmission either in full format or in reduced resolution. In case the available bandwidth is not sufficient, even for reduced resolution images, images will be analyzed in order to extract meta information like hot-spot location and/or position of fire fronts. Then, this meta-information can be exchanged even with the lowest bandwidth data links.

Helicopter-segment services

Figure 7 describes the main services that will compose the Red-Eye service architecture on-board the helicopter. As it can be seen, services are distributed in two sub-networks, one of them dedicated to data acquisition and management while the second one is dedicated to data exploitation and user HMI interfaces.

The data acquisition subnetwork contains specific services to manage the image acquisition of both the thermal and the high resolution visual camera. Both services will provide

streams of synchronized images, each one tagged with the local clock of the computers managing both IP cameras. Image acquisition will be pre-planned when executing area or perimeter scans, while individual images are requested when specific hot-spot points are under analysis.

In parallel to the image acquisition services a navigation service, linked to a INS/GPS unit, will provide a continuous stream of computed positions and angular information. Both camera services will attach this position information, in addition to the time stamp, so that other services can process and orto-normalize the images for further analysis. This navigation stream will also be used by other services within the Red-Eye architecture to manage, for example, the vehicle navigation.

Finally, this image stream is directly stored in a data-base service that acts as main data storage, but at the same time as processing queue to provide the hot-spot analysis services to exploit the available thermal images as fast as the available on-board computation permits, or to transmit information to the ground segment as fast as the available communication links permit.

The data exploitation and user HMI interfaces network contains the on-board controller interface that should provide all the HMI capabilities previously described in Section 3. This service will orchestrate the overall system operation by using a set of supporting services.

Figure 7. Overview of the services implementing the aerial segment of the Red-Eye system.

The Mission Management Service will organize all automated elements within the operation of the Red-Eye system, i.e. request the computation of the necessary scanning patterns, organize the collected information for each area under exploration and for each scan in each area, the activation of the sensors at the start and during the scanning, etc.

The Hot-spot Analysis Service will process the thermal images acquired in order to detect heat. In a first phase a crude selection criteria will be used. If traces of heat are detected, a more detail analysis will be performed, identifying the specific location in the image in which the hot-spot is located, and possibly correlating this information with adjacent images. Once a particular hot-spot is detected, the thermal image and its high-resolution visual counterpart will be ortho-normalized and merged. All the combination will be properly stored in the supporting storage service associated to all the remaining scanning information. Notifications and or alarms will be sent to the services tracking the operation on the ground so that the images can be scheduled to be transmitted.

A Geo-server Database Service provides terrain reference to properly perform the ortho-normalization tasks, but also to provide information for the safe navigation of the helicopter over the terrain. The objective is to have enough information so that the scanning operations are scheduled at the appropriate/safe separation from the ground.

The overall information gathered by the aerial segment of the Red-Eye system can be transmitted to the ground control or even to the individual squads operating on the ground. This is achieved through the Communication Gateway Service, a subsystem capable of selecting which one of the existing com-

munication links is the available at any time (taking into account communication shortage due to loss of line-of-sight), and routing the information scheduled to be sent to the ground accordingly to its priority. Moreover, this gateway is capable of connecting itself to the MANET on the ground, so that it can participate as an additional MANET node for achieving connectivity between ground squads that will be isolated otherwise (thus permitting a sort of over-the-horizon capability).

Finally, the Pilot Support Interface and the Red-Eye Operator Interface will implement the overall HMI interaction with the Red-Eye pilot and the operator. The first will offer references to the pilot so that he can fly the required tracks and scanning maneuvers; while the second will offer the operator the required interfaces to design the required scanning operations and the analysis capabilities required to identify the most critical hot-spots in real-time.

Ground-segment services

Figure 8 describes the main services that will compose the Red-Eye service architecture on the ground. Again, services are distributed in two sub-networks, one of them dedicated to communications while the second one is dedicated to data analysis and user HMI interfaces.

5. UNDERLAYING MAREA MIDDLEWARE

Middleware-based software systems consist of a network of cooperating components, in our case the services, which implement the business logic of the application and an integrating middleware layer that abstracts the execution environment and implements common functionalities and communication channels. In this view, the services are semantic units that behave as producers of data and as consumers of data coming

Figure 8. Overview of the services implementing the ground segment of the Red-Eye system.

from other services. The localization of the other services is not important because the middleware manages their discovery. The middleware also handles all the transfer chores: message addressing, data marshaling and demarshaling (so subscriber services can be on different platforms than the publisher service), delivery, flow control, retries, etc. Any service can be a publisher, subscriber, or both simultaneously. This publish-subscribe model virtually eliminates complex network programming for distributed applications.

In our architecture (MAREA) [10], the middleware takes the form of a component called service container. The services are executed and managed by a service container that is unique in each node of the distributed system network. The service container can manage several services and provides common functionalities (network access, local message delivery, name resolution and caching, etc.) to the services it contains. The service container supports mechanisms that go beyond the basic publish-subscribe model. The key benefit is that services are entirely decoupled. Very little design time has to be spent on how to handle their mutual interactions. In particular, the services never need information about the other participating services, including their existence or locations. The container automatically handles all aspects of message delivery, without requiring any intervention from the service, including: determining who should receive the messages, where recipients are located and what happens if messages cannot be delivered.

Middleware capacities

More concretely, the main functionalities that provide the service container are:

- **Service management:** The container is the responsible of starting and stopping the services it contains. It is also on charge of watching their correct operation and notifying the rest of containers about changes in the services behavior.
- **Name management:** The services are addressed by name, and the Service Container discovers the real location in the network of the named service. This feature abstracts the programmer from knowing where the service resides or how to communicate with it. In case of service malfunctioning, it is also the container responsibility to notify the other containers in the domain. In this way, the containers are able to clear and update their caches. From the name management point of view, the Service Container acts as a proxy cache for the services it contains.
- **Network management and abstraction:** The services do not access the network directly. All their communication is carried by the service container. This abstracts the network access, allowing the middleware to be deployed in different networks. Moreover, the container hides the bookkeeping related with the management of UDP/TCP ports and multicast groups.
- **Resource Management:** Given that each network distributed node has a unique container, and that all the services in that node are layered on top of it, the container is the right place to centralize the management of the shared resources of the node: memory, CPU time, input/output devices that are accessed in exclusive mode, etc. In some cases, for example when the node is a low-end microcontroller, the service container can act as a minimal operating system.

From the point of view of a distributed application there are basically three models for information communication: Point-to-Point, Client-Server and Data Distribution System. Our middleware is designed following the Data Distribution

System communication model because it better adjusts to the application characteristics of an aerospace mission while the other models are too simple for our purposes.

The Point-to-Point communication model is the simplest model for a distributed application. In this model the distributed elements must implement the communications one by one, specifying the target of every communication and assuming communication loses or service redundancies inside the object code. This model has been extensively used in telephony applications, using protocol stacks to reduce error message treatment. As the application complexity increases, it requires higher and higher bandwidths and more complex developments, because the connection pipes increase exponentially.

In the case of the Client-Server model, the distributed application must create differentiated objects for data providers and data consumers. The paradigm of Client-Server has been extensively used in the Internet and also in database accessing applications because it adjusts easily to network requirements and it is simple and intuitive to develop applications. In fact, Client-Server model is the base of network file systems, database management systems, and Internet services (e-mail, browser, etc.). Most middleware approaches, like SUN RPC, CORBA or Web Services are also designed under this paradigm. The Client-Server model has been extensively used with good results but mainly when information is centralized. In this case, the server centralizes the accesses to data and minimizes the possible race conditions of data modifications. Some of their weaknesses are poor performance, possibility of bottlenecks and poor robustness because of the single point of failure.

The Data Distribution Service model arose as a solution of most novel distributed applications today. It promotes a publish/subscribe model for sending and receiving data, events, and commands among the nodes. Nodes that are producing information publish that information and other nodes can subscribe to them. The middleware layer takes care of delivering the information to all subscribers that declare an interest in that topic. The Data Distribution Services model has been shown as a very good solution for many-to-many communication frameworks. They are also very efficient for distributing time-critical information.

For the specific characteristics of an aerospace mission, which may have lots of systems which may interact many-to-many, the proposed solution is based in this paradigm. Many Data Distribution Services frameworks have been already developed, each one contributing with new primitives for such open communication scenario. In our proposal we implement only the communication primitives required by a minimalistic distributed embedded system in order to keep it simple and soft real-time compliant. The mission and payload control has been used as a motivating example and guiding application. Next sections describe the proposed communication

primitives, which have been classified in four types: Variables, Events, Remote Invocation and File Transmission.

Variables

As variables, we mean the transmission of structured, and generally short, information from a service to one or more additional services of the distributed system. This information is sent at regular intervals or each time a substantial change in its value occurs. This relative caducity of the information allows to send it in a best-effort way. The system should be able to tolerate the loss of one or more of these data transmissions. This communication primitive follows the publication-subscription paradigm.

A service can provide zero or more variables. Each of them is composed of a basic type (boolean, integer, floating point real, character string, etc.) or by a composition (vector, struct or union) of basic types. From the point of view of the allowed data types in a variable our middleware is similar to a C-like language.

By means of its service container, a service announces the availability of its variables. This way, other services present in the distributed embedded system can subscribe themselves to one or more of these variables. From the moment a service subscribes to a variable, the provider service is responsible for sending it with the accorded quality of service characteristics. In any case, the services using this communication primitive should tolerate the loss of some variable values. If this situation goes on, the service container will warn of this timeout circumstance to the affected services.

The provider service can specify the variable validity as a quality of service parameter. When a variable value is lost, the subscribed services can receive previous values as long as they are still valid. In addition, the middleware has a mechanism that guarantees an initial exact value for the services that need it.

The service container maps this sort of communication over UDP packets in broadcast or multicast mode, when the underlying network allows it. This sort of transmission allows optimizing the bandwidth use because one packet sent can arrive to multiple nodes in the distributed embedded system.

Events

Events are similar to variables in the sense that both work following the publication-subscription paradigm. The main difference is that events, opposite to variables, guarantee the reception of the sent information to all the subscribed services. The utility of events is to inform of punctual and important facts to all the services that care about. Some examples can be error alarms or warnings, indication of arrival at specific points of the mission, start of some pre-programmed actions like taking a photo, etc. Events can contain associated information (error codes, current position, etc.) or have meaning

by themselves. When they carry additional information, data is coded the same way as in the variable communication primitive.

In the case of events, an important fact that has to be taken into account, apart from correct reception of the message by all subscribers, is that latency due to reception and processing may be critical. A typical solution to this problem is the reservation of time slots in both the processor and the network

The publisher and subscriber services interact with other services always using the service container like in the variable case. Finally, this communication primitive is mapped by the service container over TCP or over UDP using a mechanism to acknowledge and resend lost packets. This specific retransmission mechanism in the application layer is more efficient for event messages than the generic case provided by the TCP stack.

Remote invocation

Most middleware implements some way of distributed computing based on the remote procedure call paradigm, for example ONC RPC, CORBA or Web Services. However, for some distributed embedded domains the data publication-subscription or global data space paradigm seem more appropriate. For this reason we provide a first-class set of publish-subscribe communication primitives.

Nevertheless, remote invocation is an intuitive way to model some sort of interactions between services. Some examples can be the activation and deactivation of actuators, or calling a service for some form of calculation. Thus, in addition to variables and events, the services can expose a set of functions that other services can invoke or call remotely. The functions exposed by a service can have an arbitrary number of parameters and optionally a return value.

This communication primitive implements two-way point-to-point communication between two services; one acts as a client and the other as server. The client service is always the initiator or the communication and the server service location is abstracted by the middleware. However, a difference from previous communication primitives is that the client service is not continuously subscribed nor connected to the server service. Their relation is occasional and delimited by the time the invocation is executed.

During middleware initialization, the services check that all the functions they need to properly accomplish their task are provided by one or more services available in the network. Redundancy and fault-tolerance are managed by the middleware, that can also redirect remote calls to server services statically or dynamically.

Static allocations of the client-server relationships are useful in critical services where resources like CPU time or network bandwidth are pre-allocated. On the other hand, runtime in-

formation can be used to redirect calls to non-critical services to the server where best performance is expected. For this, load balancing techniques are used.

Upon service failure, if another service is implementing the same functionality, the middleware will detect the situation and redirect requests to the redundant service. This allows the system to continue its mission, although perhaps in a degraded mode. If no service provides the requested function the middleware will warn the system to take the programmed emergency procedure.

In some cases, both event and function primitives can be applied to realize a functionality. In this case, in our current implementation, events seem faster than their function equivalent.

This communication primitive is generally mapped by the service container over TCP, but UDP plus retransmission at the middleware level can be also used. However, this primitive is never mapped over broadcast or multicast given that is always a point-to-point communication.

File-based Transmission

In our preliminary prototypes, it has been discovered that some requirements of the mission are not fulfilled by the proposed communication primitives. In some cases, there is the need to transfer with safety continuous media. This continuous media includes generated photography images, configuration files or services program code to be uploaded to the service containers.

Some modifications could be done to the previous primitives to accept this sort of information transfer. But finally, a specific primitive has been developed to treat this case, given the huge performance benefits that can be attained. This primitive is used basically to transfer long file-structured information from a node to another and is implemented mapped directly over TCP.

Internal Implementation

MAREA has been designed with interoperability and flexibility in mind. Their internals have been distributed in several layers allowing the addition of new components such as transports and codifications. MAREA has its components distributed in four layers: Protocol, Encoding, Presentation and Transport. This distribution loosely resembles the PEPT architecture from Sun[11].

The transport layer is the lowest level layer on the MAREA stack. It is on charge of sending and receiving data from the underlying network and to interface to the upper Encoding layer. The MAREA middleware is message based so the transport layer is especially well suited for datagram based networking (i.e UDP). In the case of stream based communications, the transport layer implementation has two options:

Figure 9. MAREA Internal Implementation. The figure shows the internal delivery of data between components on the same local network.

a) one stream can be opened for each message, and after its sending, the stream is closed. b) as the stream initialization usually require a long handshaking phase it is better for performance to reuse the established connections, waiting for new messages to be send to the same destination through a unique stream. In its current status, MAREA offers an UDPTransport, TCPTransport and PersistentTCPTransport. They implement basic UDP and TCP transports for MAREA messages.

In the case of PersistentTCPTransport as the first connection with other node is established it is cached in a table indexed by node. All the following messages between the nodes that have established the connection are sent through this connection. For this transport, a specific transport has been designed to dissect the stream into messages. Basically, each message is preceded by a header that indicates the length of the next message. This way, the receiving transport layer knows when a message is complete and it can pass it to the upper Encoding layer.

MAREA transport layer is not limited to IP based communication. Although the most common connection between MAREA containers is IP over Ethernet or Wi-Fi other simpler networks are possible, for example serial RF modems. Each of these connections are represented in MAREA by an ITransport interface. This interface provides access to the underlying device and allows to send and receive data and to get link status information (i.e. link quality, expected bandwidth and latency). This modular approach allows to easily add new datalinks to the system.

The encoding layer is on charge of transforming MAREA messages into byte arrays and vice versa. Data received in the transport layer is cut into datagrams and sent to the upper encoding layer. Here, the received bytes are translated into a message that can be processed by the upper layers. In a similar way, when the Service container needs to communicate with other Service containers it forwards a Message to the encoding layer that transforms it into a coded byte ar-

ray. MAREA provides several coders and decoders to allow easy interoperability and devices/network adaptability. Depending on the selected encoding, the balance between codification latency and network bandwidth needed to transmit the encoded message can be modified. Currently, MAREA supports one optimized binary encoder and one text-readable XML encoder.

Finally, the Protocol layer is on charge of scheduling the message depending on its priority and after that, process it. Control messages managing the discovery, publication and subscription of services are managed on this layer, while data messages are feed to the subscribed services. All the services provide a common interface to get notifications and data from the container and other services.

The service interface contains operations to manage the following operations:

- **Start():** This operation is invoked for all the initial services at container startup. A service should respond to this call by allocating its needed resources and start performing its functionality.
- **Stop():** A service is asked to finish by receiving this call. Container issues this call at system shutdown, reconfiguration or during problem containment.
- **Run():** Usually, a service requires a thread to perform its job. The Run() function contains the code to be executed by this main thread.
- **VariableChanged():** When a subscribed Variable primitive gets new data, the container calls this function in the subscribed services. The function acts as a callback returning the originating service and the sent data.
- **EventFired():** This is the equivalent callback but for Event primitives.
- **FunctionCall():** Finally, Function primitives are notified by issuing this call in the producer service. The service executes the indicated operation and returns the results to the container, which will send them to the calling service.

The service container will issue this operations on the different services, as they are initiated, stopped or they receive new data from others services. In addition, the service container will contact with other containers in the local network to discover new services and manage remote publications and subscriptions. More details about the protocol between containers are given in the next subsection.

Figure 9 shows three different views of a MAREA service container receiving and processing a message through its different layers. In the leftmost figure, we can see two service containers deployed over the same aircraft internal network. More specifically, the Mission Control service notifies the Video Camera service to take an aerial photo by using a Variable communication primitive.

In the central figure, we focus on what happens inside the service container and its layers. First, we found the Transport layer where the different transports keep the communication with other service containers. Next, we found the Encoding layer where the raw data received in a transport is decoded to be scheduled and processed in the upper Protocol layer.

In the rightmost figure, the Mission Control notifies the Video Camera by using a Variable primitive so the `VariableChanged()` callback is called for all the subscribed services in the container. Then, each service will react to the input according its functionality.

6. GROUND SEGMENT AND MANET OPERATION

Adhoc networks are packet-based multi-node networks that are gaining popularity in other cooperative environments because they do not need a central point of organization (like an access point in Wi-Fi networks) and they can be self-configured as nodes appear and disappear. If we add dynamism and mobility we reach the concept of MANET (Mobile Adhoc Network) [12], [13], [14].

MANETs can extend a network more than the range of a single node range because their inherent multi-hopping capability allows the use of intermediate nodes to propagate packets to its destination.

Sensi re: Support Terminal for Ground Squads

One of the environments in which the technology has influenced less is the forest firefighting techniques. In this scenario, it usually work small groups of firemen squads, organized all around a fire truck. The fire truck commonly carries a GPS receiver and a radio unit to be in contact with a coordinating base station.

The Red-Eye system offers the ground squads a PDA system that can receive the set of hot-spots they have been tasked by the controller. The PDA set feeds back the position of the squad given its attached GPS. The PDA set can also be

instrumented in order to feedback other information like temperature, humidity level and wind speed and direction. These data samples are designed to be sporadic because they need the proper placement of the sensors. However, the GPS position will be continuously transmitted. The main advantage of using a MANET infrastructure is that information can get to the squads either directly from the Red-Eye helicopter or relayed from some other ground squad, thus minimizing the “out of coverage” situations. Also, having a continuous monitoring of the ground squads will improve their efficiency as well as prevent dangerous situations in case of major changes in the evolution of the wildfire.

These limited means generates some grave problems: difficulty for knowing the location and state of the fire brigades and lack of updated information for the decision taking. This situation is make worse by the fact that communication is difficult in these conditions. Brigades usually work in zones of difficult access and which lack adequate communication infrastructures. They mostly use voice communication over radio, and then is difficult to transmit concretely fire state or positions over the terrain.

We propose to alleviate these drawbacks by the usage of a distributed system. For this application, each fireman will carry a personal digital assistant (PDA) device communicating with a central base station where all the information is centralized and keep a global vision of the situation. The PDA is a small and light device secured to the fireman arm. This way, the PDA does not disturb during the fireman work, and he can use the device with his free hand. The functionalities provided by the device are: receive and present information from the Base Station. This include, voice messages, images (picture or videos), text messages, maps and associated GPS coordinates. It can also send these sort of messages. Meanwhile the brigade is carrying the PDA, its attached sensors are gathering useful information to the base station, brigades realtime location, temperature and humidity. This way the base center can have an updated picture of the fire and the effectives.

All the PDAs and the base station conform a mobile ad-hoc network or MANET. Obviously, this network does not guaranties a perfect coverage however it offers flexibility and mobility while its coverage can be improved by using repeaters installed in the trucks or in the aerial means. The PDA includes a service container and several services that implement the client application: Client Application, Sensor Module, Storage Module and Map Database.

- Sensor module interact with the GPS receiver and the humidity and temperature sensors. It can broadcast this data at regular intervals using the variables *temperature*, *humidity* and *position*. In case of an extreme temperature detected it can send an *alarm* to the base station using the event primitive.
- Storage module is a common component used in most of the applications following this architecture. It represents a

file system where other components can store and retrieve information in a hierarchical way.

- The Map Database is a higher level of abstraction component that allows the storage and retrieval of geo-referenced information, mainly maps.

All file-structured information, i.e. maps or images, that is need to be sent to several brigades can be sent using the file transfer primitive of the middleware. This primitive optimizes the bandwidth usage when multicast networks are used.

Communication Gateway

In air-ground communications it is very common to use different links to support communication at different ranges and to provide redundancy in the case a link is down. These point-to-point links usually do not support multicast and may have associated costs (economical or power-consumption) that restrict their usage to specific situations or very important transmissions. In this case the system should be intelligent enough to send the data through the most appropriate channel. This decision should be based on the type and length of the data to send, the current quality of the different links and the mission status.

At the same time, the communication should be as much uniform and transparent to the services as possible in order to allow the deployment of the same services in different missions over different network relays. MAREA has a specific system called Communication Gateway that it is on charge of managing air-to-air and air-to-ground communications. Their main purpose is to make the network transparent to the different services that use it. Our middleware assumes a high-speed Ethernet-like network that interconnects the different service containers. This local area network provides a very good bandwidth/cost ratio and very high reliability.

Communication Gateway requirements—The basic requirements that the Communication Gateway has to fulfill are:

- Support for several datalinks: The vehicle will have several datalinks that the services should use in command and transparent manner. New datalinks should be easily configured to be used.
- Network hardware and protocol independence The services should not be modified because using one network technology or another. The underlying network can use IP or not. For example, most RF modems provide a wireless RS-232 link; this means no data transmission integrity, no error correction and no retransmissions. MAREA Transport component implementation should resolve this transport-specific problematic while encapsulating and hiding these details to the rest of the system.
- Transparent handover between networks. When a communication session has been established between two remote service containers changing from a datalink to other should be transparent to the services involved. Imagine that we are using a high-throughput 802.11a WiFi connection to while the helicopter is near the ground station, as the distance increases,

signal will decrease and the Communication Gateway will change to RF modem communication. Established sessions will hand over from a datalink to the other.

- Prioritize traffic from different services. When the available bandwidth is not enough for all the required data flows, the Communication Gateway should prioritize the traffic from specific services. Other traffic can be discarded, postponed its transmission, or stored in persistent medium for later processing on ground. For, example command data should be prioritized always over telemetry data.
- Using simultaneously several networks for higher throughputs If a vehicle configuration provides several datalinks and in a specific point of the mission there is a high throughput necessity and they are available, the MAREA container should distribute the generated data through the different datalinks. Imagine that we have two RF modem on different bands and we want to download to ground a high-resolution aerial photo, by using the two RF modems simultaneously we will be able to finish the job in half the time.
- Security and encryption The MAREA protocol has some security checks, but in general it has not need to cope with malicious services, because the internal local network can be considered safe. When the communications are done external to this network and using a wireless channel the possibilities that this channel can be attacked are highly increased. Some effort is done in this area, although the current implementation of the current gateway lacks these features.

System architecture—The idea of the communication gateway is to add a new layer on the stack that will manage the multiple datalinks and decide which to use for each data flow. This process is dynamic as it depends on the links status. Link status can change from available to unavailable in any moment of the mission, on that case, the Communication Gateway will be on charge to transparently handover the communications from one link to other. One advantage of separating this functionality on an independent layer is that the service containers present on internal networks not needing handover and multiple network interface functionalities can remove completely this layer avoiding all the resources cost.

Communication Gateway layer is installed between the Protocol and Encoding layers, just before deciding which encoding use. It only processes output messages. Input messages are bypassed and their differentiation is performed at the scheduling done by the Protocol layer. Messages from high priority services are queued before.

An XML configuration file defines which transports to use and which policies to apply to each sort of packets. The packets are mainly classified by their producer and their consumer. In addition to this static information, the Communication Gateway receives dynamic information from the Transport layer, mainly the link status of all the transports enabled. The Transport components on the transport layer has been extended to be able to provide this information.

All information related to link status and interfaces are stored in a component called Network Manager. The Network Manager is in charge of relating addresses with its link status. With this both dynamic and static information the Selector component decides to which interface send each Message.

Services and service container layers use specific objects called TransportAddress to identify the recipient of a Message. This object not only carries the address of the underlying specific networks (for example its IP address) but also information about the encoding to use. This TransportAddress will be used by the lower layers to select the appropriate encoding and interface. Communication Gateway only needs to route the Message through the lower layers by changing the TransportAddress of the Message.

Figure 10. Communication Gateway architecture. The figure shows the layers of the MAREA middleware interacting with Communication Gateway.

A general view of the architecture is shown in Figure 10. After being processed by the Protocol layer, two packets arrive at the Communication Gateway. With the information gathered from the Transport layer and stored in the Network Manager about the two Link available the Selector decides which route to use. The XML configuration files defines that the Messages from Service 1 are encoded in binary and sent through network 1 (Wi-Fi), while the Messages from Service 2 are encoded also in binary, but sent through network 2 (RF). In addition to the Communication Gateway layer, a specific SerialTransport has been implemented to be able to use Radio Frequency Modems based on RS-232 interfaces, for example the Digi 2.4Ghz XStream module.

7. CONCLUSIONS AND FUTURE WORK

Having aerial resources working in a command and control over wildfires may improve the overall extinguishing process.

Improving the awareness through those monitoring systems may will maximize the efficiency of the available resources and at the same time improve the safety of the personnel working on the fire. However the complexity of integrating these monitoring systems together with the traditional aerial resources is a complex and unsolved task.

The Red-Eye system is a partial solution to this task. Its objective is to provide hot-spot detection over areas in which other aerial resources has been temporally removed. It has been identified that this scenario may provide important benefits in terms of efficiency. Red-Eye will allow aerial and ground resources to be employed on other fire fronts by reducing the number of resources that need to be kept on the field once a fire front has been controlled.

Red-Eye works on top of a middleware specifically designed for avionics. MAREA provides a full distributed paradigm to the avionics designers. Each avionics semantic is encapsulated into a service, or piece of software which realizes a function. Data communication between services are always conducted through the middleware. The physical location of the services is transparent to the designers, being MAREA responsible of the service discovery, location and execution. The main targets in the MAREA development process have been the communications efficiency, the transparent service location and the automatic selection of the physical link for the communications. The communications efficiency is defined as the relation between the transfer speed and system cost. The paper presents also the second layer, the gateway. This layer has information from the lowest layer about all the available links and their status. The gateway, using a configuration file, decides the best physical channel. Service ubiquity is the final result for avionics designers.

ACKNOWLEDGMENTS

This work has been partially funded by Ministry of Science and Education of Spain under contract CICYT TIN 2007-63927.

This work has been co-financed by the European Organisation for the Safety of Air Navigation (EUROCONTROL) under its CARE INO III programme. The content of the work does not necessarily reflect the official position of EUROCONTROL on the matter.

REFERENCES

- [1] "Upnp forum," <http://www.upnp.org>. [Online]. Available: <http://www.upnp.org>
- [2] Y. Kaufman, C. Justice, L. Flynn, J. Kendall, E. Prins, L. Giglio, D. Ward, W. Menzel, and A. Setzer, "Potential global fire monitoring from eos-modis," *Journal of Geophysical Research*, vol. 103, no. 32, pp. 215–238, 1992.
- [3] L. Giglio, J. Descloitres, C. Justice, and Y. Kaufman,

“An enhanced contextual fire detection algorithm for MODIS,” *Photogrammetric Engineering and Remote Sensing*, vol. 87, pp. 273–282, 2003.

- [4] D. Roy, P. Lewis, and C. Justice, “Burned area mapping using multi-temporal moderate spatial resolution data - a bi-directional reflectance model-based expectation approach,” *Remote Sensing of Environment*, vol. 83, pp. 263–286, 2003.
- [5] “National infrared operations - historical perspective,” Remote Sensing Applications Center, August 2004.
- [6] “Designing and implementing a near real-time wildland fire information system,” Jet Propulsion Laboratory, California Institute of Technology, 1989.
- [7] J. D. Nichols, “Firefly system concept,” in *Proc. SPIE Vol. 1540, p. 202-206, Infrared Technology XVII, Bjorn F. Andresen; Marija S. Scholl; Irving J. Spiro; Eds.*, ser. Presented at the Society of Photo-Optical Instrumentation Engineers (SPIE) Conference, B. F. Andresen, M. Scholl, and I. J. Spiro, Eds., vol. 1540, Dec. 1991, pp. 202–206.
- [8] D. Wright, T. Yotsumata, and N. El-Sheimy, “Real time identification and location of forest fire hotspots from geo-referenced thermal images,” *International Archives of Photogrammetry Remote Sensing and Spatial Information Sciences*, vol. 35, no. 2, pp. 13–18, 2004, also AIAA Paper 89–0269, Jan. 1989.
- [9] E. Pastor, C. Barrado, P. Royo, J. Lopez, E. Santamaria, X. Prats, , and J. M. Batlle, “Red-eye: A helicopter-based architecture for tactical wildfire monitoring strategies,” in *IEEE Aerospace Conference*. Big Sky, Montana: IEEE, 2009.
- [10] J. Lopez, P. Royo, E. Pastor, C. Barrado, and E. Santamaria, “A middleware architecture for unmanned aircraft avionics,” in *ACM/IFIP/USEUNIX 8th Int. Middleware Conference*, Newport, California, Nov. 2007.
- [11] H. Carr, “PEPt. A minimal RPC architecture,” in *OTM 2003*, Italy, Nov 2003.
- [12] A. Neyem, S. Ochoa, and J. Pino, “Supporting mobile collaboration with service-oriented mobile unit groupware: Design, implementation, and use,” in *Groupware: Design, Implementation, and Use*, vol. LNCS 4154, 2006, pp. 228–245.
- [13] M. Tentori and J. Favela, “Activity-aware computing in mobile collaborative working environments,” in *Groupware: Design, Implementation, and Use*, vol. LNCS 4715, 2007, pp. 337–353.
- [14] A. Neyem, S. Ochoa, J. Pino, and M. Borges, “Supporting group decision making and coordination in urban disasters relief efforts,” *International Journal of Decision Systems*, vol. 16, no. 2, pp. 143–172, 2007.

BIOGRAPHY

Enric Pastor is a Computer Science Engineer and holds Ph.D. also in Computer Science from the Technical University of Catalonia (Spain). Currently, he is an associate professor at the Castelldefels School of Technology. His research background includes the development of CAD tools for the automatic design of synchronous and asynchronous circuits, and CAV tools for the formal verification of concurrent systems. He has published more than 30 journals and conference papers in these fields. Dr. Pastor is leading the ICARUS (Intelligent Communications and Avionics for Robust Unmanned Aerial Systems) Research Group, completely focused on the development of systems that support and facilitate practical applications of UAS technology. The ICARUS group is especially interested in developing wildland fire monitoring tools and strategies both using UAS technology and classical vehicles. The ICARUS group has published more than 25 research papers on UAS technology in the last three years.

Marc Sole holds a Ph. D. (2009) On Computer Sciences by the Technical University of Catalonia, where he is also an Assistant Professor in the Computer Sciences Faculty of Barcelona. His Ph.D. thesis was focused on automated formal verification of timed concurrent systems. Research interests include formal verification, process mining, data anonymization and UAS. He belongs to the ICARUS research group.

Juan Lopez is an assistant professor at the Technical University of Catalonia. His main interests are embedded software architectures and unmanned aircraft systems design. Currently, he is working in middleware based avionics architectures for UAS.

tions.

Pablo Royo received a degree in Telecommunication Engineering from the Technical University of Catalonia. Now, he is an assistant professor at the Technical University of Catalonia. His research interests include the Unmanned Aerial System design and their application in autonomous intelligent applica-

Ph.D. Thesis was focused on automatic extraction of low level parallelism on loops at compile time. She belongs to the ICARUS Research Group, which target is the research on Unmanned Aerial Systems, their architecture, their civil uses and their integration into the airspace.

Cristina Barrado holds a Ph. D. on Computer Sciences by the Technical University of Catalonia, where she is also an Associate Professor at the Castelldefels School of Technology, Spain. She has been teaching operating systems since 1990 and currently a course of Digital Avionics Systems. Her