2014 International Conference on Unmanned Aircraft Systems (ICUAS)
May 27-30, 2014. Orlando, FL, USA

Smart Sensor Protocol – a new standard for UAV and payload integration

Rayner M. Pires, Arthur A. Chaves, Kalinka R. L. J. C. Branco Institute of Mathematics and Computer Science (ICMC)
University of São Paulo (USP)
São Carlos-SP, Brazil 13566–590
Email: {rayner, kalinka}@icmc.usp.br, arthurac@usp.br

***Abstract*—Due to the unmanned aerial vehicles versatility, the remarkable and growing use of these aircrafts nowadays has driven not only the expectations of that market values but also the amount of research in several related areas. Following forecasts of specialized roadmaps, these vehicles must evolve regarding both the maturity of the technology and the level of autonomous behavior. The organization of all the internal systems as a layered approach can make this evolution easier. Thus, by using the Mission-Oriented Sensors Array (MOSA) to decouple the mission from the aircraft control systems, the development of new applications for these vehicles can be benefited. Then the Smart Sensor Protocol (SSP) was developed in order to connect MOSA to UAV. All the grammar of the protocol was written using the Backus-Naur Form (BNF) and the communication over the SSP was modeled in UPPAAL tool using state diagrams. In this testing tool the system was mathematically described and tested using temporal logic rules.**

I. Introduction

The recent and growing popularization of Unmanned Aerial Vehicles (UAVs) – or drones, as they are commonly known – has driven an increase number of researches in this field. Inspired by the German flying bombs, V1 [1], [2], and by harmless radio-controlled air-models, the UAVs were initially created to supply military needs, such as air patrols, battlefield monitoring, access to inhospitable areas, and others. Because of the variety and versatility of these robots, they are now being deployed also in civilian's operations, mainly in the fields of energy and agro-industry, in operations such as forest monitoring, control and ministration of crops, air support for oil stations and pipelines, remote sensing, mapping, traffic monitoring and other applications.

According to data collected by Teal Group – a company of market analysis for aerospace industry – in 2012 [3], the world market of UAVs should nearly double in a decade, achieving a revenue of approximately US$11.6 billion in the year of 2022. Incorporating these values into the market of UAV payloads, a study by the company RnR Market Research [4] showed that the global market of UAVs has reached the mark of US$ 43.7 billion in 2012 and, at the same study, it is made a forecast that in 2022 the sector of unmanned aerial vehicles will move in about US$ 69.6 billion.

Due to not having a pilot on board to control them, these planes can be remotely operated by electronic and computational means, or can be completely autonomous, in which there is no human intervention in either the flight control or in the mission accomplishment.

The flight control is performed by crossing the data from the aircraft's sensors – such as gyroscope, barometric unit, pitot tubes, and others – with data from the pre-established flight plan – such as the aerial routes to be covered, the flight attitudes to be performed, and information regarding the map of overflown areas – and the manipulation and starting of the aircraft's actuators, such as propellers and flaps, for instance.

In modern autonomous UAVs, the mission control is integrated to the flight control system. Thus, the aircraft control system must also perform tasks related to the embedded mission processing, such as control radars, take pictures, start loading

978-1-4799-2376-2/14/$31.00 ©2014 IEEE

and/or unloading equipment or manipulate any other device that is used during the mission.

This association of responsibilities in the same embedded computer makes the process of maintaining and expanding the involved technologies more complex than it could be. The separation of responsibilities into layers is a very efficient organization model that is highly adopted in the computational environment when better organized storage, processing or data transport between different entities are needed. A layered architecture is easier to be maintained, updated, scaled and it is more fault tolerant. In UAVs, the splitting of the mission processing environment and the aircraft control environment can bring great benefits to the development of these technologies by providing more independence and cohesion between each part of the system.

That is why it is proposed in [5] a model for a smart mission processor able to complete missions and it is used in this new paradigm, in which mission processing is detached from the aircraft system. This model is called MOSA (Mission-Oriented Sensors Array), and aims to free the UAV to control devices that are not part of its fuselage, reestablishing it as a pure transportation mean.

II. Related Work

A. Layering model

An UAV is a typical application of a complex critical embedded system. The term UAS (Unmanned Aerial System) was adopted by both the FAA (American Federal Aviation Administration) and the international academic community to describe systems that include not only the aircraft but all the associated and support elements, such as the payload, the ground control station and communication links [6].

There are different types of UAVs presenting different abilities. Some aircrafts can fly autonomously, following a pre-programmed flight path (based on a grid or a sequence of waypoints) [7], [8], while others can fly receiving commands from pilot-operated ground stations. The aircraft size can range from the micro to the large and the ground control station can be implemented in smartphones, tablets, laptops or networks of workstations (distributed control stations). The plane may vary not only in size but also on shape, type of propulsion and performance. The human-machine interface can range from a joystick to a tangible user interface (for example, a table with tangible augmented reality). The performance of the communication links and load type are also very important to fulfill the mission intended to system.

All papers related to building UAVs found in the literature have typically implemented these machines using traditional approaches [9], [10], [11], [12], [13], [14], where there is no specification or applications of standards. There are roadmaps that illustrate the expected advances to UAVs, published periodically by military organizations, such as the United States Air Force [13]. The practical use of innovations such as those presented in this proposal is not expected in the short term. However, they will be very important within the next decades, when the UAVs may permeate a good part of the airspace to carry out different missions, from agricultural border inspections to automatic cargo transportation [15], [16].

It is expected in a slightly more distant future that the only manned aircrafts that will need to remain active are those carrying passengers, as it happens in other transport systems, e.g., some underground railway systems, where the driver must act in emergency cases only.

This growing use of UAVs may cause them to be increasingly common, becoming more widely marketed. In this scenario, the techniques proposed in this work will facilitate the development of automated applications for UAVs, allowing these vehicles to be more easily inserted and incorporated into the airspace and contributing to their spread.

Thramboulidis et al (2007) [17] propose in their work an approach to use service-oriented architecture (SOA) in general embedded systems. The paper proposes a framework that facilitates the development of embedded systems using the available resources as services. They provide an

easy way to integrate the desired features and components (through techniques such as Plug-and-Play), which must be provided by service-oriented architectures.

Critical embedded systems like UAVs have special requirements. The real-time performance is almost mandatory and must be guaranteed under any circumstances. Some mechanisms of SOA, such as service discovery, have the potential to nondeterministic behavior and are not compatible with the basic requirements of critical embedded systems. Alternatively, complex critical embedded systems can often be divided into critical and noncritical (or low criticality) sections. It is an approach presented on [18] and illustrated on Figure 1.

A definition of a model as that of Figure 1 is important not only for establishment of a standard for UAVs development – then benefiting compatibility between different platforms – but also for creation of means for certification of this vehicles so they can occupy the unsegregated airspace too.

A layered organization allows a system to be divided into subsystems that can have different and independent implementations. Moreover it helps separate parts of a complex embedded system into different levels of criticality.

Parts with lower criticality can make use of advantages offered by SOA, without them being harmed by possible non-deterministic behaviors guiding this architecture.

Regarding this consideration, the three green-highlighted layers on Figure 1 are considered low-critical parts of the aerial segment and is the current environment of this work.

B. Interoperability through plug-and-play protocols

Many of protocols considered plug and play, or having this characteristic, are associated with the automation industry, especially those intended for home automation (for smart homes). Among the options, technologies such as Jini [19], Universal Plug-and-Play (UPnP) architecture [20], HomePnP [21] and DLNA [22] are stand out.

Despite the diversity of protocols for distributed architecture (and some of them even with open architecture) providing communication between devices that offer and request services, none of them apply to the model proposed in this paper. Current solutions have in common the proposal to distribute the task of control between multi-function devices and connect them through an appropriate network architecture, allowing information exchange between them for the execution of scheduled tasks without the need for centralized control units, which, in general, has increased the efficiency and reliability. In contrast, there is a substantial complexity in design and management of these systems.

The communication and integration for cooperation in the backdrop of UAVs is the key to interoperability between devices such as smart sensors and processing subsystems (such as mission processing boards, navigation and autopilot systems and possible embedded web-servers). Such integration has connection and communication requirements more restrictive than the previously mentioned architectures and can be benefited with more appropriate and tuned solutions for this environment.

III. DECOUPLING MISSION PROCESSING AND AIRCRAFT CONTROL

Mission-Oriented Sensors Array (MOSA) [5] is a specific set of sensors and devices arranged and controlled by a dedicated processor, conceived to execute a specific mission. Figure 2 shows MOSA architecture.

In a mission where an UAV must fly over a forest and, for instance, identify irregular clearings, a MOSA might be mounted with a single RGB camera, and uploaded to it a mission file containing the area to be overflown and every point to be photographed. The mission processor itself will be responsible for operating the camera in the prestored waypoints. Also, another example might be the identification of clandestine constructions or fraudulent flora exploration in the same previous forest. For accomplishing that an infrared camera could be added to MOSA and furthermore the tasks of controlling this new sensor could be inserted into the previous mission file.

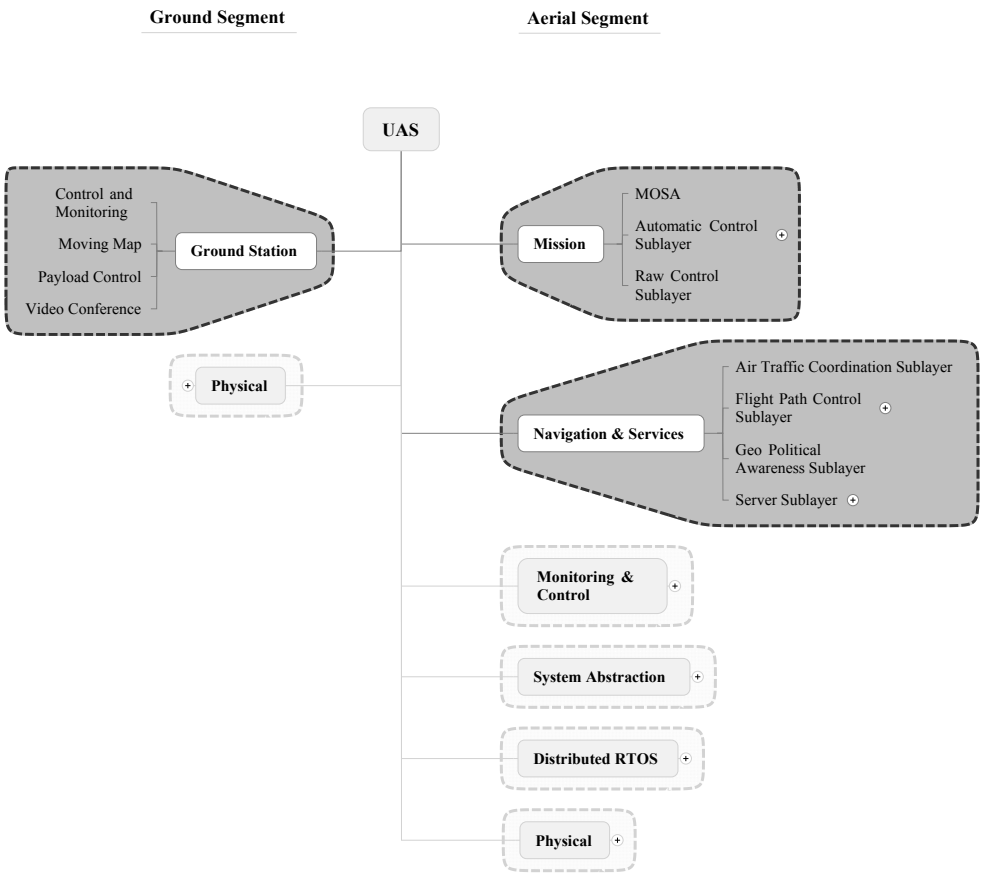

Fig. 1. This is a simplified illustration of a larger UAS segmentation model proposed in [18]. In this model some layers are considered as high-critical and others as low-critical, which have requirements of operation more flexible (or not hard real-time). The three low-critical layers are the upper and highlighted layers.

These manipulations in MOSA do not require any modification in UAV's control system, making it feasible the re-utilization of the aircraft for more than one type of mission.

Under new perspective, the MOSA model allows for aircraft manufacturers to produce their planes without concern or knowledge in producing a mission processing system. Other companies can also produce a great variety of MOSAs that are compatible with commercial aircrafts without the need to build their own aerial vehicles. This portability solution favors the market growth and expansion of these technologies.

In layered models, it must be defined the access interfaces between adjacent layers. In this scenario where the layers are two hardwares, it must be defined both physical connection and logical communication layers, by selecting existing standards or creating new ones more suited to the demand.

The logical layer definition, by software, is the focus of this paper, since the hardware is usually part of each subsystem and will be abstracted by the software layer.

Articles in opened literature do not contemplate, at the present, a layered model for a critical-system communication, which is the context of this paper. Therefore, the authors have not found protocols that could meet the minimum requirements or a simple and suited protocol to provide such communication.

In this work, we designed and validated an application level protocol that mediates all communication between MOSA and UAV, called Smart Sensors Protocol (SSP).

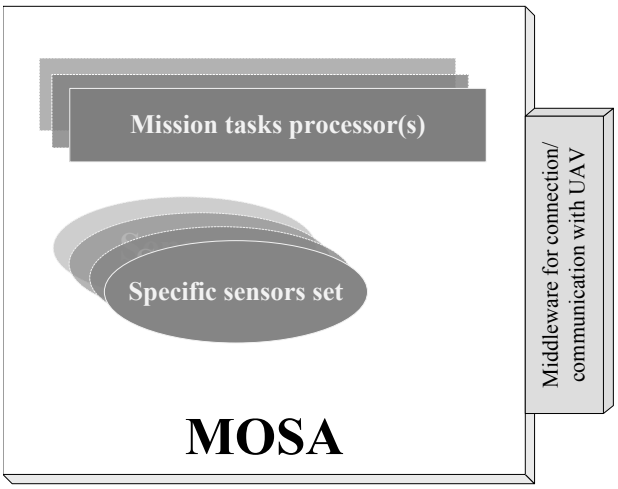

Fig. 2. A representation of MOSA, with its set of sensors specific for a programmed mission; jointly with the processor responsible for executing mission tasks pre-stored in an uploaded file, and the middleware which is responsible for providing connection with the UAV and possibly for providing power to MOSA board.

IV. The SSP protocol

SSP coordinates the communication between a MOSA instance and an UAV instance. It has a set of operations that allow to verify the executability of the mission, allowing a communication between subsystems during mission execution, and closing this cooperation session. Nevertheless it was designed to be simple, easy to update and extendable to other domains of autonomous vehicles.

The architecture of the SSP protocol is formed by four elements:

1) Mission processor – MOSA, which role is to read and interpret data from sensors attached to itself, and send to UAV requests for flight characteristics, displacement and services;
2) MOSA middleware, which role is to allow the proprietary implementation of messages from MOSA to be translated to SSP standard, and vice versa;
3) UAV middleware, that is responsible to do the bidirectional translation between the messages of the UAV's subsystems and the SSP message format;
4) UAV, which receives and responds requests from MOSA, such as, displacement requests, information about the aircraft's characteristics or about the current flight etc.

TABLE I. SSP primitives, along with their modifiers.

Step		Primitive	Modifiers
Negotiation		REQ	Environment, Luminosity, Weather, AltitudeBoundaries, SpeedBoundaries, Endurance, Secrecy
		RET	
Negotiation, Execution & Finishing		SEND	`<DATA>`
		ACK	Send, Goto, Notify, Land, Close
Execution		TAKEOFF	`<HomeWaypoint>`
		READY	
		GOTO	ID, Priority, Latitude, Longitude, MinAltitude, MaxAltitude, Speed
		NOTIFY	`<WpID>`
Finishing	**Landing**	LAND	ID, Priority, Latitude, Longitude, MinAltitude, MaxAltitude, Speed, Heading
	Closing	CLOSE	
Cancellation		ABORT	

The first and last elements above interact with each other through their respective SSP middlewares. These middlewares, on the other hand, make the intermediation of that communication through a set of primitives. This interaction is illustrated in Figure 3.

The primitives are classified in five possible stages of communication: (i) Negotiation, (ii) Execution, (iii) Landing, (iv) Closing, (v) Cancellation.

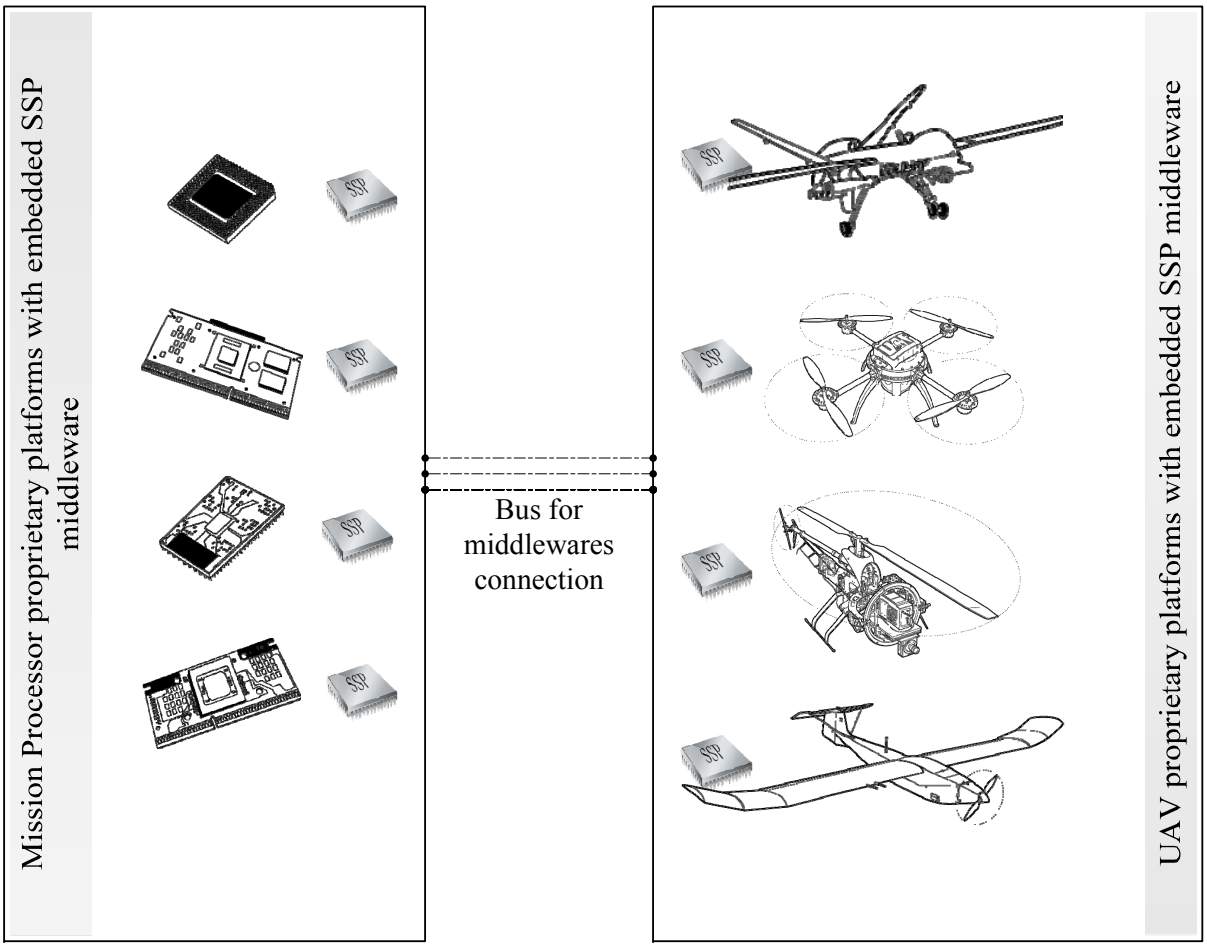

Fig. 3. Representation of SSP operation environment.

Table I is a simplified representation of the SSP grammar. The first column of the table presents the four phases a communication over SSP is done. The second column presents the primitives of the protocol, and the third one shows main modifiers of those primitives.

SSP messages have a simple format $[\ header \mid data\ ]$. The header contains two fields. The first field is the version of the protocol – which was defined by the string "1.0" at the present – and the second field is the service primitive of the message packet, where the possible options correspond to primitives cited in second column of Table I.

Example of message packets are shown in Figures 4 and 5. Packet fields are filled with constant values, represented by text surrounded by double quotes, or with structures defined in SSP's grammar, represented by words surrounded by angle brackets.

The general perspective of the SSP communication is also described by a state diagram, as shown in Figure 6.

In this diagram, each edge makes transition from one state to another through sending a message packet, represented by the edge label. Message sending occurs alternately between communicating entities. In this way, two successive state transitions are never triggered by the same entity, which avoids messages collision.

The conversation between both middlewares starts after the connection of boards and the first

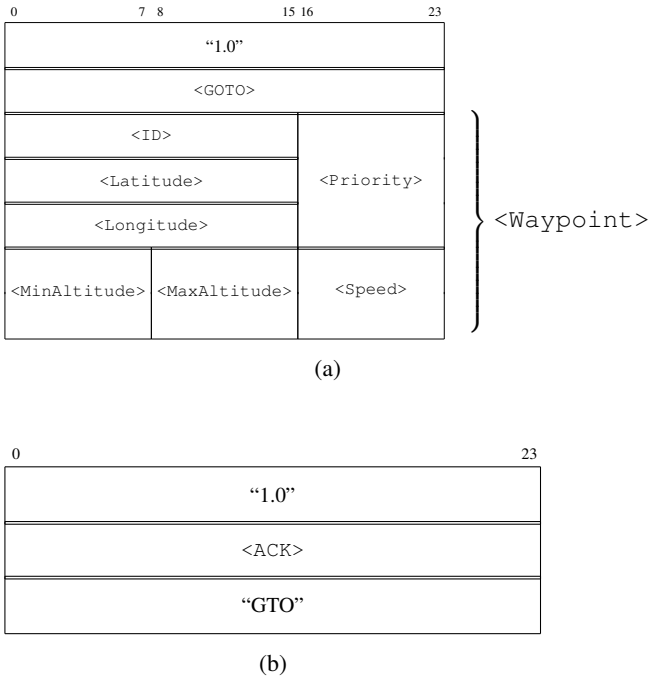

Fig. 4. Message packets exchanged during a displacement request. 4(a) is the request sent by MOSA to UAV, while 4(b) is the UAV response to MOSA, acknowledging the reception of the former message.

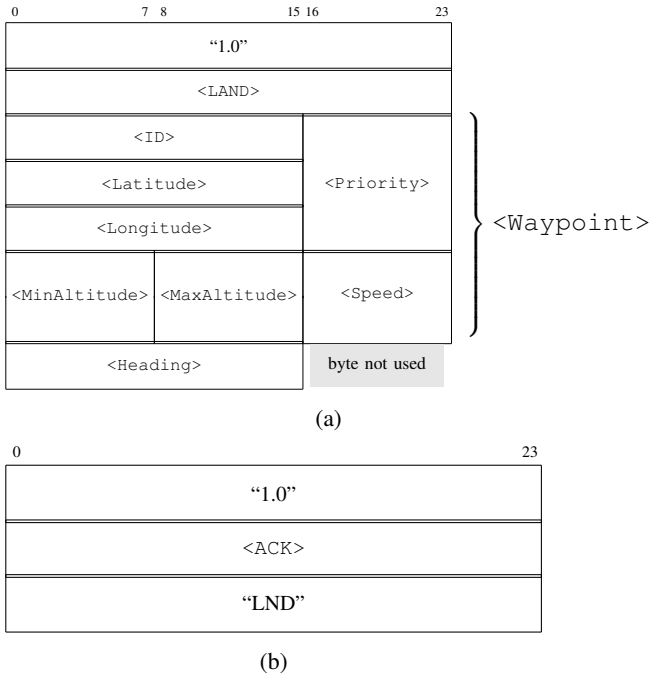

Fig. 5. MOSA uses a message packet as shown in 5(a) to indicate to UAV its landing waypoint and the angle of approach (<Heading> field). UAV acknowledges MOSA the reception of a LAND message with a message packet like 5(b).

message emission by the aircraft side. Cooperation for mission accomplishment occurs in three sequential phases: (i) the negotiation of mission parameters and the definition of its feasibility, (ii) processing of this mission and (iii) the closing of mission cooperation session and finalization of communication.

The first stage of the conversation between the entities is named Negotiation, where requests and responses about aircraft features are made. Transitions of Negotiation phase are ranging from state $q0$ to state $q3$ in Figure 6.

Ended the negotiation phase, the mission processor has labeled its embedded mission as feasible or unfeasible. If it is feasible, then it proceeds to the second stage, the Execution phase. During this stage, the aircraft provides the ability to air displacement and, in case of being implemented, it offers some services to mission processor. Transitions of Execution phase appear from state $q2$ to state $q7$.

Completed the execution of all tasks of the mission, the aircraft is still in the air and needs to be landed. This operation must be requested by the processor when there are no more tasks to be performed. This is the Landing stage. In this phase, the processor sends a LAND message informing the runway position and its angle of approach. Transitions of phase are $q7 \rightarrow q10$ and $q10 \rightarrow q11$.

When is received from UAV the acknowledgment of the previous LAND order (transition $q10 \rightarrow q11$), communication proceeds to mission finalization and connection shutdown. It is the Finalization phase. Its transitions range from state $q11$ to state $q13$.

One can note ABORT transitions in $q6 \rightarrow q13$ and $q7 \rightarrow q13$, which are triggered exclusively by UAV when some more preemptive interruption occurs internally to the aircraft, causing mission execution interruption and cooperation finalization. These transition may be triggered during mission running time, or when a ground station takes control of the aircraft, or the aircraft is forced to execute some pre-programmed code due to unexpected occurrence, such as fuel end or

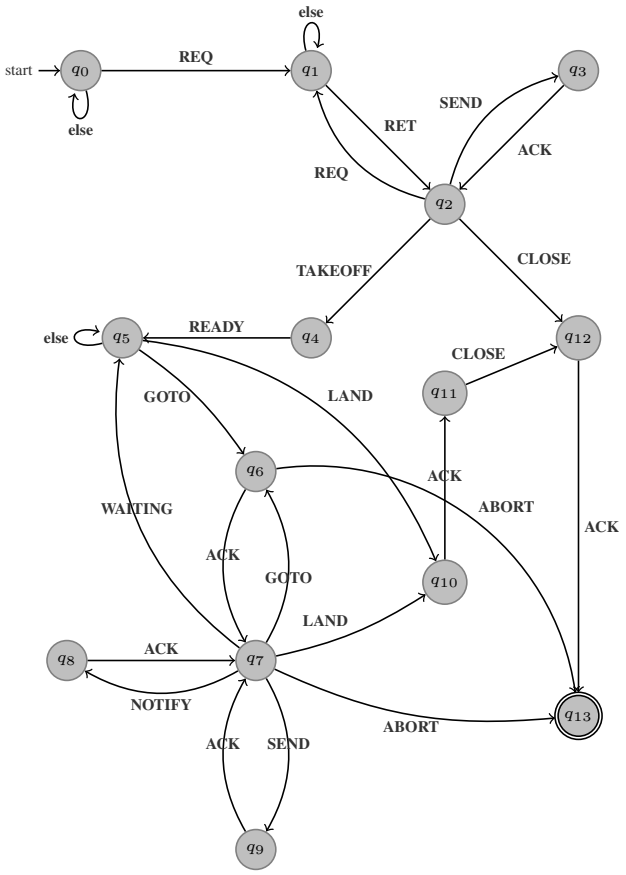

Fig. 6. State diagram of messages exchange between two SSP middlewares. State *q0* is the start state and *q13* is the final state. One observes the presence of *else* transitions to ignore unrecognized packets.

unauthorized foreign airspace reach etc.

This is an important primitive of SSP, because it informs to mission processor that it will have no more response by the aircraft system. Furthermore, it makes possible to UAV platforms to inform they are assuming control of themselves in special situations.

As formerly written, the same entity never triggers two successive transition of states. This model's characteristic of alternating the use of the communication channel is a high level mechanism to guarantee messages order. Although it is not part of the current model yet, message integrity guarantee can be added to it easily, just inserting a new field to integrity hash in message packets.

Because it is modeled at application layer and its mainly objective is the interconnection between two different platforms, the current version of SSP protocol avails the benefit of message exchange technique, widely used in this sort of scenario [23]. Also, it was not modeled a fragmentation mechanism for messages, which lays under responsibility of the transport layer below.

In order to occur a communication between two entities, it is not enough define only the vocabulary that they should use. It is necessary to lay down rules for grouping vocabulary terms. The SSP protocol modeling work also included description of a grammar to define the rules for using the service primitives.

The grammar's syntax was formally described using Backus-Naur Form (BNF), a meta syntax used to express context-free grammars (CFG). CFG are formal grammars in which each production rule has the notation $V \mapsto w$, where V is a unique non-terminal symbol and w is a string of terminals and/or non-terminals or the empty string $\emptyset$.

BNF is widely used as a notation for grammars of programming languages, for instruction sets in compilers projects, and for description of communication protocols. That's why it is used in this work.

A BNF specification is a set of production or derivation rules, noted as follows:

```
<symbol> ::= <expression with symbols>
```

where `<symbol>` is a non-terminal, and `<expression with symbols>` is formed by sequence of symbols separated by a vertical bar `|`, indicating a choice for substitution of whole left symbol of `::=` signal. Symbols that are never written on the left are called terminals.

A complete grammar were described for SSP. Due to its big extension it was not included in this paper, but can be obtained with authors.

V. Validation of Specifications

To verify and test the developed protocol, this system was modeled into a tool for validation and model checking already used in consolidated communication protocols. Model checking is a powerful technique to make debugging of complex reactive systems, such as communication

protocols. In model checking, specification of a system is made with logical sentences, and efficient symbolic algorithms are used to analyze the model and cross it with its specification. The tools for verifying models capture the dynamic behavior of systems using for this only states and transitions between states of the model.

The tool used to check the SSP model is UPPAAL[1] in 4.0.13 version. As described by the authors in [24], UPPAAL is a set of tools for the task of verification of real-time systems, developed jointly by Uppsala University, at Sweden, and the Aalborg University, at Denmark.

Among all verification tools found at literature, UPPAAL was chosen because there are many successful case studies of modeling and validation of communication protocols widely known and utilized [25], [26], [27], [28], [29], [30], [31], [32], [33], [34].

In UPPAAL, a system is modeled as a network of several timed automata in parallel. A timed automaton is a finite-state machine extended with clock variables, that progress all of them synchronously. The system model is further extended with bounded discrete variables that are part of the state. These variables are used as in programming languages: they can be read, written, and are subject to common arithmetic operations. A state of the system is defined by the locations of all automata, the clock values, and the values of the discrete variables. Every automaton may fire an edge separately or synchronize with another automaton, which leads to a new state [24].

In UPPAAL the components of the system being modeled are called templates. The two main templates of SSP model implemented were "MOSA" and "UAV", that share with each other a common channel of communication. UAV template also have exclusive shared channels with other templates that were modeled for supporting all tasks UAV should perform. Those templates are "UAV_negotiator", "UAV_waypointProcessor" and "UAV_msgProcessor".

All five templates are instantiated and these instances compose the system that UPPAAL simulator should run. This simulator starts all instances at initial states and transitions are made with non-deterministic choice of possible inputs. While running, the simulator shows in graphical UI all the changes that are being made into system variables, as well as all state transitions are getting triggered. This helped identifying errors in model and has yielded a well tested model.

UPPAAL also provided a way of mathematically test the model, using Temporal Logic, a logic the considers time as a sequence of states [35]. Temporal Logic has an important application in formal verification to establish hardware and software requirements. UPPAAL recognizes five prepositions of temporal logic: (i) possibly, (ii) invariantly, (iii) probably always, (iv) eventually and (v) leads to.

The rules for the modeled system defined, essentially, that every process must start together at the initial state and, whenever a process reach its final state, all the others necessarily need to reach their respective final states too. It aims guarantee three important properties of a communication protocol: reachability, safety and liveness.

Based on the modeled diagrams inside the tool, we created temporal logic prepositions to test the instantiated systems. As the diagrams, due to large amount of data, these rules were not included in this paper but are available through the authors.

All prepositions we have created were exhaustively tested along with the diagram model and modifications and corrections were being made until we get successful results.

VI. Conclusion

Unmanned Aerial Vehicles are very employed in military operations since the 60s, as in case of Vietnam's attacks to extinguished USSR. Recently, they have been protagonists of many news in which their use, mainly by USA and Israel armies, are cited. But, besides military use, one have seen arising civilian applications, as use in civil police for supervision of activities of companies that explore the environment; the use by agro-industrial companies for fault recognition

[1] http://www.uppaal.org/

and disease detection in plants; crowd control in big events like the world cup, beyond many usages in cinematography industry.

This large variety of applications for those robots needs a standard to make programming them more homogeneous.

In order to mathematically validate the model presented in this paper, we used a formal model verification tool called UPPAAL [36]. UPPAAL is an integrated tool environment for modeling, validation and verification of real-time systems modeled as networks of timed-automata.

By using the context-free grammar described in BNF for the protocol description, a whole environment was modeled and its behavior checked in UPPAAL. Then, a mathematical validation was made with inferences using temporal logic supported by the tool.

The rules for the model created in the tool defined that, essentially, all processes involved must start in the initial state; that every state is likely to be reached; and that when any of the processes reach a final state, all other processes must have reached their own final state.

When these mathematical tests were executed, all rules were satisfied. It indicates that characteristics like progress realizations, lack of deadlocks and correct termination are guaranteed in the current model. So, this guarantee allows the usage of the protocol in a test environment such as hardware-in-the-loop.

Also the SSP protocol and the MOSA model presented in this paper can be considered as an initial impulse for a standardization of the communication with unmanned flight platforms.

ACKNOWLEDGMENT

The authors would like to thank the support granted by CNPq and FAPESP to the INCT-SEC (National Institute of Science and Technology – Critical Embedded Systems – Brazil), processes 573963/2008-9 and 08/57870-9, as well the support granted by FAPESP, processes 2011/06086-9, 2011/044161 and 2012/08498-5.

REFERENCES

[1] K. Cook, "The silent force multiplier: The history and role of UAVs in warfare," in *Aerospace Conference, 2007 IEEE*. IEEE, March 2007, pp. 1–7.

[2] J. Sullivan, "Revolution or evolution? the rise of the UAVs," in *Proc. 2005 International Symposium on Technology and Society. Weapons and Wires: Prevention and Safety in a Time of Fear.* IEEE, Jun. 2005, pp. 94–101.

[3] Teal Group Corporation. (2013, Jun. 17) Teal group predicts worldwide UAV market will total $89 billion in its 2013 UAV Market Profile and Forecast. Electronic. Paris, FR. [Online]. Available: http://tealgroup.com/index.php/about-teal-group-corporation/press-releases/94-2013-uav-press-release

[4] Strategic Defence Intelligence, "The global UAV payload market 2012-2022," RnR Market Research, Tech. Rep., Dec. 2012.

[5] R. M. Pires, D. Rodrigues, K. R. L. J. C. Branco, and O. Trindade Jr, "MOSA – mission oriented sensors array: a proposal," in *Proc. XXXVII Conferencia Latinoamericana de Informática – CLEI*, Quito, Ecuador, 2011, pp. 1309–1318.

[6] J. Mica and J. F. Costello, "Unmanned aircraft systems: Federal actions needed to ensure safety and expand their potential uses within the national airspace system," GAO-08-511, U.S. Government Accountability Office, Washington, DC 20548, May 2008. [Online]. Available: http://www.gao.gov/products/GAO-08-511

[7] O. Trindade Jr, L. C. P. Barbosa, L. d. O. Neris, and L. A. C. Jorge, "A mission planner and navigation system for the ARARA project," *23rd ICAS - Congress of International Council of the Aeronautical Sciences*, Sep.8–13 2002.

[8] O. Trindade Jr, L. de Oliveira Neris, L. C. P. Barbosa, and K. R. L. J. C. Branco, "A layered approach to design autopilots," in *IEEE International Conference on Industrial Technology - ICIT*, Mar. 2010, pp. 1415–1420.

[9] E. C. Aldridge Jr and J. P. Stenbit, "Unmanned aerial vehicles roadmap 2002-2027," US Department of Defense (DoD), Report, Dec. 2002.

[10] S. A. Cambone, K. J. Krieg, P. Pace, and L. WellsII, "Unmanned aircraft systems roadmap 2005-2030," US Department of Defense (DoD), Report, Aug. 2005.

[11] J. R. Clapper Jr, J. J. Young Jr, J. E. Cartwright, and J. G. Grimes, "Unmanned systems roadmap 2007-2032," US Department of Defense (DoD), Report, Jan. 2007.

[12] J. R. Clapper Jr, J. J. Young Jr, J. E. Cartwright, J. G. Grimes, S. C. Payton, S. J. Stackley, and D. Popps, "Unmanned systems integrated roadmap fy20092034," US Department of Defense (DoD), Report, Apr. 2009.

[13] M. B. Donley and N. A. Schwartz, "United states air force unmanned aircraft systems flight plan 2009-2047," United States Air Force, Whashington, DC, Report, May 2009.

[14] K. P. Valavanis, *Advances In Unmanned Aerial Vehicles: State Of The Art And The Road to Autonomy*, ser. International series on intelligent systems, control, and automation. Springer, 2007, vol. 33.

[15] K. Branco, J. Pelizzoni, L. Oliveira Neris, O. Trindade, F. Osorio, and D. Wolf, "Tiriba – a new approach of UAV based on model driven development and multiprocessors," in

Robotics and Automation (ICRA), 2011 IEEE International Conference on, vol. , no. , May 2011, pp. 1–4.

[16] O. Trindade Jr, L. A. C. Jorge, and J. G. B. Aguiar, "Field of dreams – using UAVs for precision farming," *Unmanned Systems Magazine*, pp. 35–39, Nov./Dec. 2004.

[17] K. C. Thramboulidis, G. Doukas, and G. Koumoutsos, "A SOA-based embedded systems development environment for industrial automation," *EURASIP J. Embedded Syst.*, vol. 2008, pp. 3:1–3:15, Jan. 2008.

[18] D. Rodrigues, J. C. Estrella, M. Vieira, J. B. Camargo Jr, K. R. L. J. C. Branco, and O. Trindade Jr, "Service-oriented architectures for complex safety-critical embedded systems: a case study on UAVs," in *Proc. I Brazilian Conference on Critical Embedded Systems – CBSEC*, São Carlos-SP, Brazil, 2011, p. 130.

[19] SUN, "Jini architecture specification," Sun Microsystems, Palo Alto, Tech. Rep., 2011, 26p.

[20] Microsoft, "Understanding universal plug and play: White paper," Microsoft Corporation, Redmond, Tech. Rep., 2000, 45p.

[21] CEBus Industry Council, "HomePNP$^{\mathrm{TM}}$ specification version 1.0," HomePlug&Play$^{\mathrm{TM}}$*: CAL-Based Interoperability for Home Systems, Indianapolis, Ind*, pp. 1–111, 1998.

[22] DLNA Organization, "Volume 1: Architectures and protocols," in *DLNA Networked Device Interoperability Guidelines*, Mar. 2006, vol. 1, p. 594.

[23] G. S. Hura and M. Singhal, *Data and computer communications: networking and internetworking*. Boca Raton, FL, USA: CRC Press, Inc., 2001.

[24] G. Behrmann, A. David, and K. G. Larsen, "A tutorial on UPPAAL," in *Formal Methods for the Design of Real-Time Systems: International School on Formal Methods for the Design of Computer, Communication, and Software Systems*, ser. Lecture Notes in Computer Science, M. Bernardo and F. Corradini, Eds. Bertinora, Italy: Springer Berlin Heidelberg, September 2004, vol. 3185, pp. 200–236, revised Lectures.

[25] A. Ravn, S. Vighio, and J. Srba, "A formal analysis of the web services atomic transaction protocol with uppaal," in *Proceedings of the 4th International Symposium On Leveraging Applications of Formal Methods, Verification and Validation (ISOLA'10)*. Springer-Verlag, 2010.

[26] A. Ravn, J. Srba, and S. Vighio, "Modelling and verication of web services business activity protocol," in *Proceedings of the 17th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, 2010.

[27] A. Hessel and P. Pettersson, "Model-based testing of a WAP gateway: an industrial study," in *Proceedings of FMICS and PDMC 2006, LNCS 4346*, 2006.

[28] B. Bordbar, R. Anane, and K. Okano, "An evaluation mechanism for QoS management in wireless systems," in *proceedings of the 11th International Conference on Parallel and Distributed Systems, ICPADS 2005*, vol. II, 2005, pp. 150–154.

[29] J. Bengtsson, W. O. D. Griffioen, K. J. Kristoffersen, K. G. Larsen, F. Larsson, P. Pettersson, and W. Yi, "Automated analysis of an audio control protocol using UPPAAL," *Journal of Logic and Algebraic Programming*, vol. 52–53, pp. 163–181, Jul./Aug. 2002. [Online]. Available: http://www.it.uu.se/research/group/darts/papers/texts/bgkllpw-jlap02.pdf

[30] M. Lindahl, P. Pettersson, and W. Yi, "Formal Design and Analysis of a Gearbox Controller," *Springer International Journal of Software Tools for Technology Transfer (STTT)*, vol. 3, no. 3, pp. 353–368, 2001.

[31] A. David and W. Yi, "Modelling and analysis of a commercial field bus protocol," in *Proceedings of the 12th Euromicro Conference on Real Time Systems*. IEEE Computer Society, 2000, pp. 165–172.

[32] M. Lindahl, P. Pettersson, and W. Yi, "Formal Design and Analysis of a Gear-Box Controller," in *Proc. of the 4*th *Workshop on Tools and Algorithms for the Construction and Analysis of Systems*, ser. Lecture Notes in Computer Science, vol. , no. 1384. Springer–Verlag, Mar. 1998, pp. 281–297.

[33] H. Lönn and P. Pettersson, "Formal Verification of a TDMA Protocol Startup Mechanism," in *Proc. of the Pacific Rim Int. Symp. on Fault-Tolerant Systems*, Dec. 1997, pp. 235–242.

[34] K. Havelund, A. Skou, K. G. Larsen, and K. Lund, "Formal modelling and analysis of an audiovideo protocol: An industrial case study using uppaal," in *Proceedings of the 18th IEEE Real-Time Systems Symposium*, San Francisco, California, USA, 3-5 December 1997, pp. 2–13.

[35] A. Galton, "Temporal logic," in *The Stanford Encyclopedia of Philosophy*, Fall 2008 ed., E. N. Zalta, Ed., 2008. [Online]. Available: http://plato.stanford.edu/archives/fall2008/entries/logic-temporal/

[36] UPPAAL Team, "UPPAAL: About," http://www.it.uu.se/research/group/darts/uppaal/about.shtml, nov 2013.