In Proceedings of the 9th ESA Workshop on Advanced Space Technologies for Robotics and Automation
'ASTRA 2006' ESTEC, Noordwijk, The Netherlands, November 28-30, 2006

An Efficient Software Architecture for a Planetary Rover

G. Battistoni[1], F. Ravera[1], A. Sgorbissa[2] , R. Zaccaria[2]
[1]*Alcatel Alenia Space Italia S.p.a.*
Corso Marche 41
10146 Torino, Italy
[2]*DIST- University of Genova*
Via Opera Pia 13
16145 Genova, Italy

INTRODUCTION

Research on rovers for planetary exploration has led to a number of different hardware and software architectures [1][2][3], some of which have been successfully experimented, e.g. in the Mars missions. Proposed models in literature are constrained by some main principles: i) a generally low computing power (with respect to standard “Earth”electronics), ii) the need for allowing reprogramming from earth during the mission, iii) a superior reliability. To comply with these principles, proposed architectures largely use static allocation of tasks in some “zone” of the on-board computer, or into a dedicated processor. The different algorithms involved (planning, communication, navigation, trajectory control, locomotion, localisation, vision, hazard management, manipulation, experiment control, etc.) are deterministically allocated in order to safely use a portion of computational resources, whereas communication and synchronization among them is provided by real time protocols. Nevertheless, this “classical” approach has some drawbacks. First, the use of limited computational resources is largely inefficient, because of static tasks allocation. For the same reason, soft degradation capabilities are limited, so that the whole system is weaker, for example, with regard to a processor failure. Second, a necessarily simple kernel does not prevent the occurrence of concurrency errors (priority inversion, lock, starvation and so on), as it did occur in real cases. Third, when we have to consider systems composed of many robots, which actively cooperate to achieve a common objective or to perform a tightly coupled task [4][5], inter-robot real-time communication and task allocation pose further constraints on the system architecture. The paper proposes an approach in which the different rover’s tasks are analysed, showing as they can be grouped in algorithm sets and related functional / temporal dependencies sets; such sets are redundant and can activate a subset of functionalities on the basis of the local mission requirements, of the computing power demand, and of the available resources. A critical component of this approach is the algorithm for the analysis of dependencies and schedulability. Its purposes are: i) task choice on the basis of available computational resources and mission requirements, and ii) efficient task scheduling, using the resources at the highest possible level. In this way it is possible to increase the system’s robustness and graceful degradation capabilities, optimizing the hardware redundancy by re-loading in different ways software modules into the processors. For example, if the hardware has some failure, functions that normally operate concurrently (e.g., visual odometry and trajectory control) may be automatically loaded into a unique processor and made operate serially, so that the mission may continue at a lower speed without re-programming from Earth. The paper describes the automated analysis of tasks dependencies and schedulability, and the scheduling algorithm; the scenario includes a rover capable of mission planning, GNC, vision; the results are discussed by means of both simulations and real experiments.

ETHNOS

ETHNOS (Expert Tribe in a Hybrid Network Operating System) is a framework for the development of Multi Robot systems. It is composed of:

- A dedicated distributed real-time operating system, supporting different communication, execution, and representation requirements; these functionalities are built on top of a Posix compliant Linux kernel.

- A dedicated network communication protocol designed both for single robot and Multi Robot systems, taking noisy wireless communication into account.
- An object oriented Application Programming Interface (API) based on the C++ language.
- Additional development tools (simulator, graphical interfaces for monitoring and remote controlling, etc.).

The reference architecture of a single robot, and consequently of the ETHNOS operating system, is entirely based on the concept of "experts", concurrent agents responsible for specific activities. Experts in ETHNOS can be organised in groups, possibly distributed in a computer network, depending on their computational characteristics: the type of cognitive activity carried out, duration, type of data managed, etc. However these groups do not have a hierarchical organisation but, on the contrary, are executed in a flat model in such a way that the real-time constraints are not violated. This generality in the expert specification allows the use of ETHNOS with different architectural choices without loosing real-time analysis, execution and inter-robot communication support. This aspect are very important in autonomous and semi-autonomous planetary rovers, in which different cognitive approaches (e.g., behaviour based, hybrid deliberative/reactive, etc.) can be adopted depending on the computational power and on the sensors and actuators available, the temporary malfunctioning of a part of a system (i.e., the equivalent of a cognitive deficiency), or even the tasks which the single robot or a group of robots are requested to perform.
The next paragraphs will deal with these properties in greater detail.

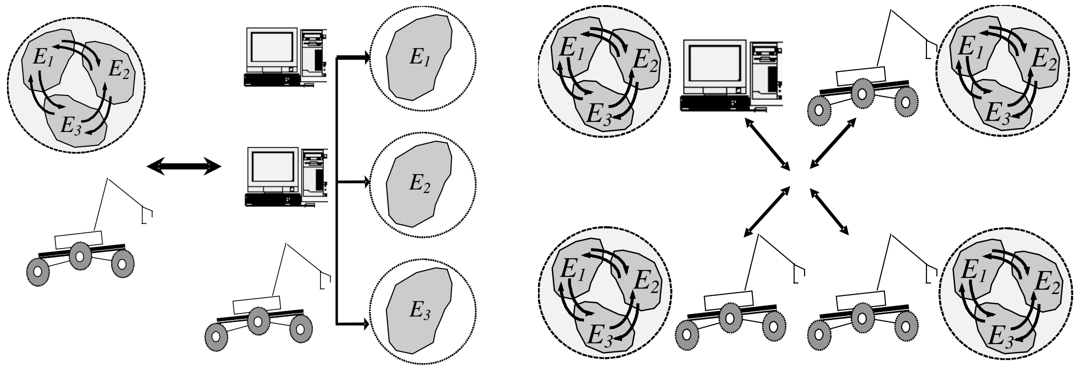


Fig. 1. Left: Experts E_1, E_2, E_3 are distributed in a LAN. Right: multi-robot configuration.

Inter-Expert and Inter-Robot Communication

ETHNOS allows the robots to be programmed in such a way that the different experts can be integrated, during development but also at run time, with little or no intervention in the software of the expert itself, thus facilitating both rapid prototyping and dynamic architectural reconfiguration. The first property allows the development of a robotic application, from a set of basic components or behaviours, even by non-highly specialised programmers: this is particularly relevant in a preliminary phase, in which experiments are conducted in a laboratory on the Earth to test different hardware and software solutions. The second property allows the system to switch at run-time from a configuration in which it is performing a certain task (for example in which the robot is tracking a distant target and thus the active experts are responsible of path planning, visual odometry, trajectory generation) to a different configuration (for example in which the robot is collecting samples and the active experts are responsible of fine motions, maintaining the asset, etc.). These properties are achieved by exploiting a suitable inter-expert communication protocol which comprises of three different communication methods (a comparison between communication patterns and their relevance in component-based robotics can be found in [6]):
- Asynchronous message-based communication.
- Synchronous access to an expert (similar to function calling).
- Access to a shared representation.

The first method is the most general of the three and it is at the base of ETHNOS applications, hence it is the only one discussed here. It is a message-based communication protocol (EIEP – Expert Information Exchange Protocol) fully integrated with the expert scheduler. EIEP encourages the system developer to de-couple the different experts in execution to reach the limit situation in which the single expert is not aware of what and how many other experts are running; in this way, an expert can be added or removed at run-time without altering the other components of the system. Expert de-coupling is achieved by eliminating any static communication link: EIEP is an asynchronous message-based method in which the single expert "subscribes" to the desired message types, known a priori. When another expert "publishes" any message of the subscribed types, the subscriber receives it transparently. In this way, an expert does not know, explicitly, who the sender of a message is or to which other experts, if any, its message is being distributed. Moreover, the same principles apply also if planning & control activities are distributed in a network of on-board processors (for example, to deal separately with on-board reactive components and remote high-level components responsible of computationally intensive tasks: Fig. 1 on the left).

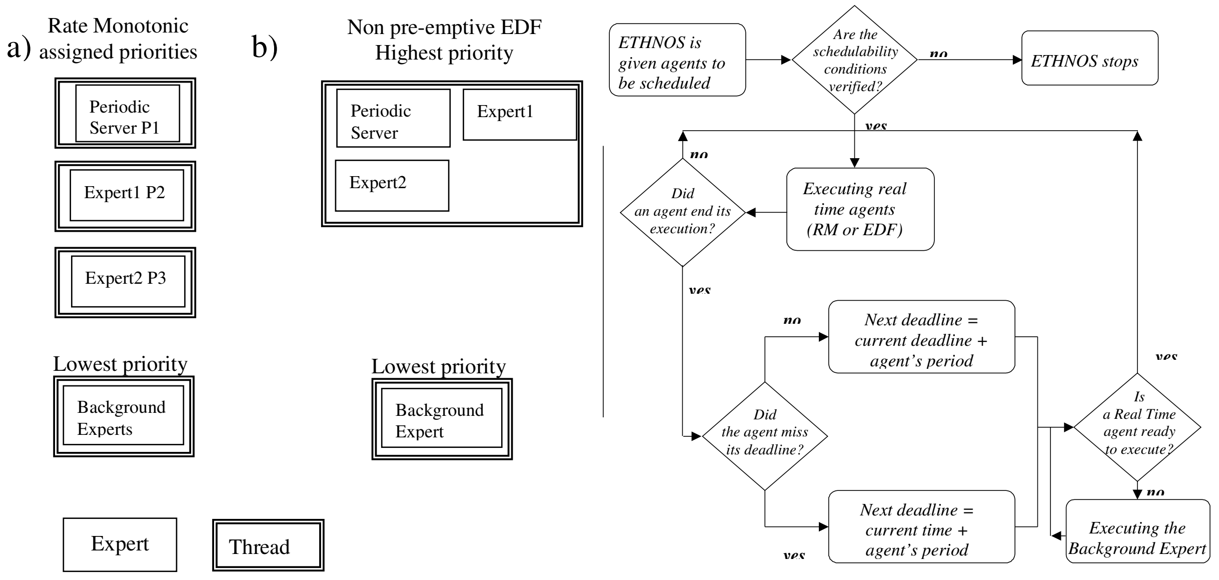
EIEP provides also transparent inter-robot communication support. Using the same publish/subscribe principle, messages can be exchanged on the basis of their content not only within the same robot but also among different robots (e.g., a colony of planetary rovers which cooperate for the achievement of a common task). However, in inter-robot communication, it is often important to distinguish between internal messages (messages to be distributed within the experts implementing a single rover) and external messages (messages to be sent from a rover to another) to avoid the explosion of unnecessary network data communication (e.g., commands to be issued to actuators or attitude measurements are meaningful only locally). In ETHNOS the different experts are allowed to subscribe to "communication clubs"; we may envisage a single club to which the different rovers belong or even different clubs, to which only rovers which are performing a cooperative activity belong, for example two robots which carry an heavy object. Message subscription and publication can thus be distinguished in internal (possibly distributed), internal and external in a specific club, or internal and external in all clubs. Again, it is the responsibility of the system to transparently distribute the messages to the appropriate receivers; Fig. 1 on the right shows robots that communicate in a single club, to which all of them have subscribed together with an external supervisor. Because of EIEP, whenever we want to add or remove a member from the team, it is not necessary to explicitly inform other team members about the modifications in the team's composition. This has been very important in space application, in which a rover can be temporarily unavailable, both for a system crash or a fault in communication; in all this situations, the team can be easily and dynamically modified to a different composition.

Finally, in wireless communication, transmission packets are sometimes lost due to noise or interference. In this context, standard communication protocols such as TCP/IP and UDP/IP cannot be used: the former because it is intrinsically not efficient in a noisy environment; the latter because it does not guarantee the arrival of a message, nor provides any information on whether it has arrived or not (for efficiency reasons, ETHNOS communication is built on top of the standard Unix Socket interface). To deal with this, ETHNOS comprises a protocol called EEUDP (Ethnos Extended UDP). EEUDP allows the transmission of messages with different priorities. The minimum priority corresponds to the basic UDP (there is no guarantee on the message arrival) and should be used for data of little importance or data that is frequently updated (for example the robot position in the environment, that is periodically published). The maximum priority is similar to TCP because the message is sent until its reception is acknowledged. However, it differs because it does not guarantee that the order of arrival of the different messages is identical to the order in which they have been sent (irrelevant in ETHNOS applications because every message is independent of the others), which is the cause of the major TCP overhead. Different in-between priorities allow the re-transmission of a message until its reception is acknowledged for different time periods (i.e. 5 ms, 10 ms, etc.). Notice that, when sending messages with the minimum priority, EEUDP is an efficient implementation of MailBoxes in a distributed scenario, an asynchronous communication method particularly suited to real-time applications [7] since it does not delay the execution of hard-real tasks (even if Ethernet-based communication – in a strict sense – is not real-time, since the time required for packet-delivering is not predictable).

Real-Time Expert Scheduling

ETHNOS supports real-time analysis and execution of experts. Throughout the chapter, the term real-time is used in its formal sense, by referring to the mathematical theory of real-time scheduling and communication as depicted, for example, in [7]. This is not a common approach in mobile robotic architectures (with some exception, e.g. [8][9]); however, we argue that schedulability analyses assume primary importance whenever computational resources are limited, which is exactly the case for space applications (and all other systems which have constraints on their dimensions, weight, an on-board power supply). Suppose that one needs to understand the cause of an observed anomalous behaviour: e.g., the robot stands still whereas we would expect it to move towards an area of interest to collect samples or just explore it. Where does this behaviour come from? Is it a bug in the code, or a temporary CPU overload that delays time critical tasks? Verifying that each task, when taken singularly, meets deadlines is not sufficient; real-time scheduling analyses allow to consider the set of tasks as a whole, and to know in advance if what we are asking the CPU to do is feasible at all.
Before discussing ETHNOS scheduling policies, we qualitatively analyse the timing requirements of planetary rover applications. Results can be summarized as follows:

- Sensor acquisition is handled by periodic tasks. They are executed at a high frequency, their computational time is usually low and predictable: e.g., reading encoders; acquiring video; measuring the rover attitude through gyroscopes, accelerometers, inclinometers, etc.
- Reactive planning & control activities are periodic as well. Frequency is high, computational time is low and predictable: e.g., computing speed & jog values for reaching a target position; avoiding obstacles; updating odometry, etc.
- Data filtering, segmentation, and aggregation activities are periodic or sporadic (their period depends on sensor acquisition) with a computational time at least one order of magnitude greater, but still predictable: e.g., visual tracking; visual odometry; map building; localization of important features, etc.
- Inter-robot communication is aperiodic, with a non-predictable computational time; in a distributed context, a robot does not know in advance when and for how long it will receive messages from team-mates.
- Aperiodic, computationally intensive activities can be optionally considered, whose time cannot be easily predicted but – usually – is orders of magnitude greater: collaborative self-localization, symbolic planning, etc.


a) Rate Monotonic assigned priorities
Periodic Server P1
Expert1 P2
Expert2 P3
Lowest priority
Background Experts
b) Non pre-emptive EDF Highest priority
Periodic Server
Expert1
Expert2
Lowest priority
Background Expert
Expert
Thread
ETHNOS is given agents to be scheduled
Are the schedulability conditions verified?
no
ETHNOS stops
yes
Executing real time agents (RM or EDF)
Did an agent end its execution?
no
yes
Did the agent miss its deadline?
no
Next deadline = current deadline + agent's period
yes
Next deadline = current time + agent's period
Is a Real Time agent ready to execute?
yes
no
Executing the Background Expert
Fig. 2. Left: Mapping experts onto threads with RM and EDF. Right: ETHNOS scheduler

Notice that only the former four activities are critical for the system, whereas the latter class includes activities which are usually performed when the robot is not moving, i.e. they can be considered as off-line activities. An overall architecture for the development of a rover for planetary exploration must take these aspects into account and thus permit the integration of less computationally intensive activities (whose execution is critical for the system, and therefore have hard real-time requirements) with more computational intensive ones (whose execution is not critical for the system, and therefore have not real-time requirements).

Periodic, time bounded, critical activities (Sensor acquisition, Reactive planning & control, Data filtering, segmentation, and aggregation) are executed in high priority experts according to two possible algorithms: Rate Monotonic or the non-preemptive Earliest Deadline First algorithm [10]. EIEP allows exchanging information between real-time experts asynchronously, thus avoiding unpredictable delays or priority inversion phenomena. Aperiodic, not time-bounded, non-critical activities are executed in low priority experts; they run in the background and communicate their results to real-time experts whenever their outputs can improve the behaviour of the system. Asynchronous communication through EIEP guarantees that background experts do not interfere with the real-time scheduling policy of the system. Inter-robot communication deserves a special attention. In fact, if we want the robot to coordinate through explicit communication, this should be considered a critical activity; however, it cannot be scheduled as the other real-time tasks since we are not allowed to make reasonable assumptions about when and for how long messages are going to be exchanged with other robots. The solution adopted is common in literature for aperiodic real-time tasks: we insert in the system a couple of Periodic Servers [11], i.e. real-time periodic tasks which are given the lowest period (hence the minimum latency) and a pre-defined "capacity" to serve requests from aperiodic tasks. In our case, the capacity of RxServer and TxServer is spent, respectively, for receiving and sending messages. The mechanism guarantees that a predefined amount of CPU time is always devoted to inter-robot communication, thus avoiding starvation. Taking all these consideration into account, ETHNOS implements different possible scheduling policies ranging from two extremes (Fig. 2) in which:

- Real time Experts (periodic, sporadic, and Periodic Servers) are mapped, one to one, into different Posix threads and scheduled as stated by the Rate Monotonic algorithm, whereas background experts are assigned a lower priority and scheduled in time-sharing.
- All real time Experts are mapped into a single high priority thread, which non pre-emptively schedules the real-time as stated by the non-preemptive Earlies Deadline First algorithm, whereas background experts are assigned a lower priority and scheduled in time-sharing.

The two extreme solutions (as well as intermediate ones) can be adopted depending on the application considered. In the former, the Rate Monotonic algorithm (chosen because of its ease of implementation in every Posix compliant OS) is exploited, allowing low latencies in expert execution (depending on the granularity of the Posix scheduler). In the latter, low latency is traded off with an increased processor utilisation, since the presence of only one thread for high frequency tasks guarantees a lower context switching rate with respect to a fully pre-emptive scheduler; moreover, it allows to access critical regions without synchronization primitives and related problems.

As well as providing transparent real-time execution (automatic computation of the required priorities) ETHNOS also includes tools for analysing the feasibility of the scheduling. To allow this, in a preliminary phase, the worst execution time for each expert is computed, and continuously updated when the system is running and also across different runs. Whenever the user asks the system to schedule a set of experts with the selected RM or EDF algorithms, ETHNOS acts as follows (Fig. 2 on the right):

- For each expert (periodic or sporadic) the scheduling conditions (for RM or EDF) are tested referring to its worst execution time and the period (for sporadic experts, the Minimum Interrarival Time between two consequent runs). The "capacity" of RxServer and TxServer is automatically computed.
- If the scheduling is possible, experts are scheduled according to the selected policy. Periodic experts are scheduled according to their period, whereas sporadic and aperiodic ones are activated whenever they receive a EIEP message to which they have subscribed.

- Should an expert miss its deadline (this is possible because of the approximate execution time analysis) a very simple recovery policy is adopted: the next deadline of the expert is obtained by adding the expert's period to the end of the last run (not to the end of the previous period). This is equivalent to changing the phase of the expert, saving the processor from computational overload and from the consequent deadline missing by other experts.
- Background experts are scheduled when no other expert is ready for execution. Should we need a temporary intensive activity in background it is a matter of the specific application to reduce real-time tasks activities.

Finally, the system allows to group experts into subsets of "semantically equivalent" activities. Consider for example traditional dead reckoning methods (i.e., computing the rover position on the basis of wheels encoder and inertial units) and visual odometry (i.e., computing the rover position on the basis of video input). The latter obviously require heavier computations than the former, and is possibly executed at a different rate (since both activities depend on the rate according to which inputs are available); nevertheless, they provide exactly the same kind of information (possibly with a different degree of uncertainty). In this context it seems reasonable that, whenever a set of experts is not schedulable, the system tries to substitute them with alternative experts providing the same information at a lower computational cost. Thus, for example, visual odometry can be substituted by traditional odometry; motion planning in a 3D map of the environment can be substituted by reactive planning. Obviously, selecting different experts possibly requires the activation in cascade of other experts, which on their turn require preconditions to be satisfied. The system performs this automatically; if a set of experts is not schedulable according to the tests in Fig. 2, an alternative configuration is attempted, and the new requirements are back propagated until all the preconditions are fully satisfied and the whole set of experts successfully passes the schedulability test (this is done through a standard, STRIP-like planner). Notice also that it is possible to define some experts as "mandatory" (i.e., not substitutable), thus allowing the user to manually select experts which he/she wants to be scheduled even if they require expensive computations.

Some technical considerations: recently, programming frameworks have been proposed relying on Linux RT extensions (e.g., RTAI – www.rtai.org) to exhibit hard real-time characteristics. On the opposite, we have chosen the Linux standard distribution, which follows Posix specifications for real-time (thus being very portable), but have many limitations: in particular, does not allow scheduling tasks at very high frequency (>100Hz) and has significant jitters in task execution. Our choice is justified by the following two considerations. First, hardware platforms for planetary rovers have dedicated processors or microcontrollers for low-level I/O functions that are executed at a high frequency: the tasks running on the on-board computer have looser requirements (i.e., frequency is << 100Hz), and are therefore compatible with the standard Linux kernel capabilities. This is not by chance: motion control is often a complex task in planetary rovers as a consequence of the many degree of freedom to be controlled in a coordinated fashion, and it is consequently very common to distinguish between the "locomotion subsystem" and the "navigation subsystem". Here we focus on the latter, where it is most important to deal with limitations in computational resources, than guaranteeing latencies below 1ms. Second, Posix compliance guarantees portability, and allows us to easily recompile a new version of ETHNOS as soon as a new version of the underlying Linux kernel is available. ETHNOS currently takes benefit of the 2.6 series of the Linux kernel, which is considerably improved and therefore more adequate to real-time applications: among the others, it is now fully re-entrant, the priority-based internal scheduler works in constant time, and its granularity is one order of magnitude smaller (1ms). An RTAI-enhanced version of ETHNOS designed for the control of hybrid systems, composed of mobile robots and manipulators, has been recently made available.

EXPERIMENTAL RESULTS

The reliability of EIEP communication has been experimentally verified both in benchmarks and with our outdoor robot team ANSER (one of them is shown in Fig. 3). ANSER has been designed for autonomous surveillance, and therefore its motion capabilities are limited to almost flat outdoor surfaces (grass, paved road, etc.) with a maximum slope of 15 degrees. However, since we are mostly focusing on the "navigation subsystem" (i.e., we ignore the “locomotion subsystem”) it is reasonable to assume that the ANSER team can be taken as a testbed even for planetary exploration requirements.

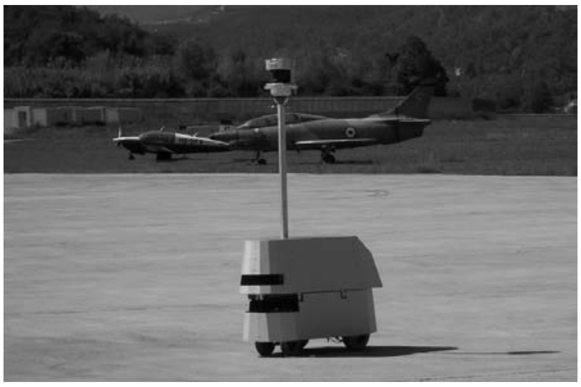


Fig. 3. ANSER mobile robot

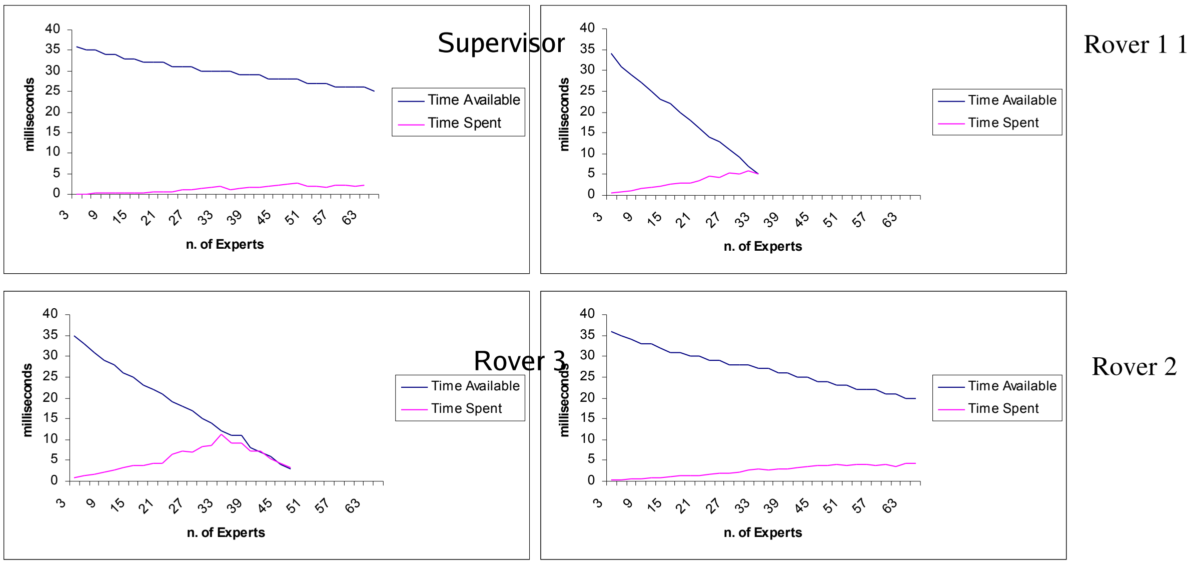


Fig. 4. Network Communication in ETHNOS

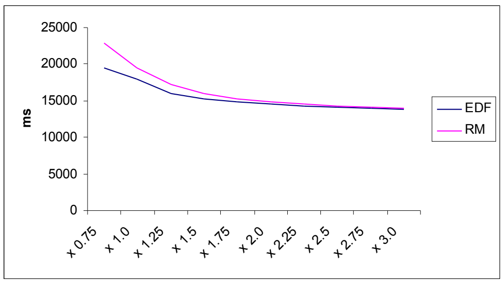


Fig. 5. Time taken to complete operation with different task periods.

One of ETHNOS peculiarities is the fact that, for network communication, the system allocates a maximum guaranteed and dedicated time (also referred to as capacity) within a couple of expert which are given the highest priority possible (i.e., RxServer and TxServer). The server capacity value is computed automatically on the basis of the schedulability analysis, in such a way that the real-time execution of other experts is also guaranteed; moreover, since servers has the maximum priority, we are guaranteed that, within a predictable and reasonable amount of time, all messages to and from other robots will be sent and received (without considering problems related to communication noise).

In Fig. 4, the four graphs represent different machines (with different processing power) corresponding to three rovers and a monitor (a visual interface), connected using wireless Ethernet. The top line in the graph indicates the calculated time available every 100ms for communication purposes; clearly this value decreases as the number of experts in execution increase (and so the computational load). The bottom line indicates the time spent in communication every 100ms; since, for this experiment, we have assumed that the activity of every expert involves either transmission or reception of messages, this value increases with the number of experts. In this way it is always possible to determine a priori whether the system is capable of both communicating information and executing in real-time. For example, if we consider Rover 1, the limit situations in which the two lines converge is also the limit beyond which the schedulability analysis fails. Instead, if we consider Rover 2, real-time scheduling is possible also beyond the intersecting point, but only if we accept communication degradation. Notice that, in real-time systems, an alternative and very common solution to deal with network communication is to execute it in low-priority, non real-time tasks (e.g., QnX, LinuxRT, RTAI). The problem is that, if real-time tasks heavily utilize the CPU, background communication activities can be delayed for an indefinite time, which is definitely not desirable.
To clarify the concept consider Fig. 5, which compares the two scheduling techniques RM and non-preemptive EDF available in ETHNOS. In the experiment, a fixed number of experts with a pre-defined computational time are scheduled together with a single background expert; in absence of RxServer and Tx Server, the background expert can be devoted to network communication, or other computationally intensive activities (collaborative localization, planning, etc.). The time taken to carry out the background expert is plotted against the factor applied to the base period of the periodic experts; the smaller the factor, the shorter the periods and therefore the higher the processor load. As expected, when the processor load is high, the time required by the background expert to end its computations increases; moreover, the time available significantly depends on the context switches rate, as outlined by the fact that non-preemptive EDF is more efficient than RM in these conditions. As a consequence, communication in background turns out to be very hardly predictable.

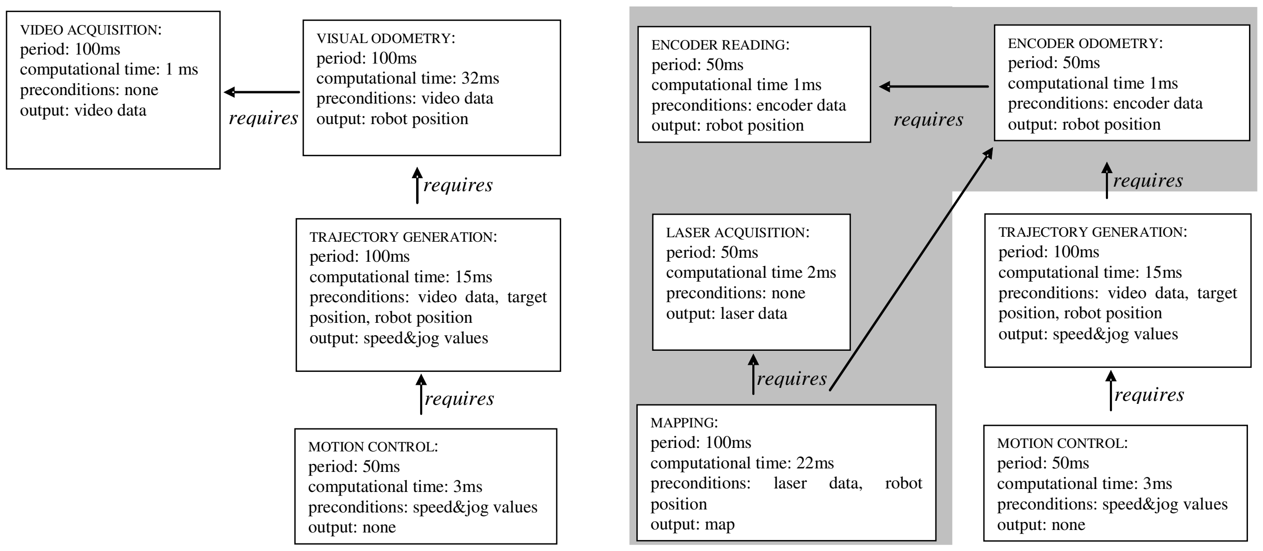


Fig. 6. Automatic re-configuration of the system from time *t* (left) to time *t+1* (right).

Finally, Fig. 6 shows an example of autonomous re-configuration of a rover during an experimental run, when schedulability analysis fails due to the introduction of new experts (the preconditions, outputs and timing requirements are shown only for a subset of the experts currently running. At time *t* the following experts are running: MOTION CONTROL, VIDEO ACQUISITION, VISUAL ODOMETRY, TRAJECTORY GENERATION; at time *t+1* the activation of MAPPING is requested, a computationally expensive activity which – in our setup – relies on the availability of range measurement provided by a laser rangefinder. The system automatically activate LASER ACQUISITION (to satisfy the preconditions of SLAM), but it realizes that the new set of experts is no more schedulable. Consequently, it substitute VISUAL ODOMETRY with ENCODER

ODOMETRY, which on its turn requires ENCODER READING to be activated. The new subset of experts is schedulable and consequently activated. Notice also that VIDEO ACQUISITION is active as well, since it is one of the precondition of OBSTACLE AVOIDANCE.

CONCLUSIONS

The paper has described an approach in which different rover's tasks are grouped in algorithm sets and related functional / temporal dependencies sets; such sets are redundant and can activate a subset of functionalities on the basis of the local mission requirements, of the computing power demand, and of the available resources. A critical component of this approach is the algorithm for the analysis of dependencies and schedulability. Its purposes are: i) task choice on the basis of available computational resources and mission requirements, and ii) efficient task scheduling, using the resources at the highest possible level. With this approach it is possible to increase the system's robustness and graceful degradation capabilities, optimizing the hardware redundancy by re-loading in different ways software modules into the processors.

The paper has described also the automated analysis of tasks dependencies and schedulability, and the scheduling algorithm in a scenario including a rover capable of mission planning, GNC, vision; the results have been discussed by means of both simulations and real experiments.

REFERENCES

[1] Albus, J.S., Lumia, R., McCain, H.G., "Hierarchical Control of intelligent Machines Applied to Space Station Telerobots", 1987 IEEE International Symposium on Intelligent Control, Philadelphia, PA, January 18-20, 1987

[2] Bonasso, R., Firby, R., Gat, E., Kortenkamp, D., Miller, D., and Slack, M. (1997). Experiences with and architecture for intelligent, reactive agents. Journal of Experimental and Theoretical Artificial Intelligence, 9(2):237 256.

[3] Pirjanian, P., Huntsberger, T., Trebi-Ollennu, A., Aghazarian, H., Das, H., Joshi, S., and Schenker, P. 2000. Campout: a control architecture for multi-robot planetary outposts. In Proceeding of the SPIE Symposium on Sensor Fusion and Decentralized Control in Robotic Systems III, vol 4196, Boston, MA.

[4] Engwirda, E., Sustainable polymorphic colonial robotics and large scale off-world construction, Autonomous Robots (2006) 20:137–148

[5] Huntsberger, T.L., Trebi-Ollennu, A., Aghazarian, H., Schenker, P.S., Pirjanian, P., Nayar, H.D., Distributed Control of Multi-Robot Systems Engaged, in Tightly Coupled Tasks, Autonomous Robots 17, 79–92, 2004

[6] Schlegel, C. (2006). Communication Patterns as Key Towards Component-Based Robotics, International Journal of Advanced Robotic Systems, Vol. 3, No. 1, 49-54.

[7] Buttazzo, G. C. Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications. Kluwer Academic Publishers, 1997.

[8] Musliner, D., Durfee, E., and Shin, K. Circa: A cooperative, intelligent, real-time control architecture. IEEE Transactions on Systems, Man, and Cybernetics, 23(6), 1993.

[9] Rybski, P. E.; Stoeter, S. A.; Gini, M.; Hougen, D. F.; & Papanikolopoulos, N. (2002). Performance of a distributed robotic system using shared communications channels. *IEEE Trans. on Robotics and Automation*, Vol. 22, No. 5, 713--727.

[10]Jeffay, K.; Stanat, D.F. & Martel, C.U. (1991) On Non-preemptive Scheduling of Periodic and Sporadic Tasks, *IEEE Real time Systems Symp.*, pp 121-139, December 1991.

[11]Lehoczky, J. P.; L. Sha; & J. K. Strosnider. (1992). Enhanced Aperiodic Responsiveness in Hard RealTime Environments. *Proceedings of the Real Time Systems Symposium,* pp. 110-123, , December 1992, Phoenix, AZ.