Distributed Computing in Reconfigurable Picosatellite Networks

Tanya Vladimirova, Xiaofeng Wu, Abdul-Halim Jallad and Christopher P. Bridges
Surrey Space Centre
Department of Electronic Engineering
University of Surrey
Guildford, UK GU2 7XH
{ T.Vladimirova, X.Wu, A.Jallad, C.P.Bridges}@surrey.ac.uk

Abstract

This paper presents the results of a research project, which aims to develop enabling technologies for future distributed space architectures based on flexible, reconfigurable, evolvable, and intelligent multi-spacecraft sensing networks. One important goal of the project is to propose a distributed computing platform over wireless inter-satellite links. The paper discusses initial results on the application of distributed computing technologies to future networked constellations of picosatellites.

Key Words: *Distributed computing, picosatellite, middleware, autonomous agents, wireless communication*

1. Introduction

Future spacecraft are envisioned as autonomous, miniature, intelligent and massively distributed space systems [1]. Large multi-functional satellites, typically used for Earth monitoring and communication, are highly customised and expensive to manufacture and launch. It is envisioned that networked constellations of hundreds-to-thousands of very small satellites could be used to eventually replace the functionality of these larger satellites. The use of standard mass-producible satellite design weighing less than 1 kg will reduce substantially build and launch costs [2].

A number of picosatellites based on the CubeSat platform [3] have been launched and operated successfully. Currently many Universities all over the world are actively involved in development of CubeSat-based picosatellite missions [4, 5].

The ESPACENET (Evolvable Networks of Intelligent and Secure Integrated and Distributed Reconfigurable System-On-Chip Sensor Nodes for Aerospace Based Monitoring and Diagnostics) project, targets the development of a robust space sensor network based on flexible picosatellite nodes or *piconodes*. The network will consist of a constellation of picosatellites at various locations in the same or different orbital planes. The picosatellites will be grouped in clusters of varying density. It was originally conceived that a two-tier hierarchy of a larger microsatellite, or cluster head, in a higher orbit and the piconodes in a lower orbit would be required [6]. Standard commercial-of-the-shelf (COTS) wireless protocols will be employed to realize the inter-satellite connectivity.

Important feature of the network is its reconfigurability which will be manifested at two levels – node level and system level. The piconodes will consist of complete reconfigurable system-on-a-chip (SoC) devices to process data from various sensing elements. A generic SoC controller design, which encompasses a number of soft intellectual property (IP) cores and driver capability, will be utilized. This will enable SoC customization at run time to suit best the processing requirements of the network.

The network must be capable of adapting its topology to achieve efficient distribution of tasks among the resource-constrained piconodes. The network must also be able to concurrently drive both network and hardware resources to effectively and efficiently solve problems in case of node failures and change of mission requirements. The system-level reconfiguration will be guided by a multi-objective evolutionary (MOE) algorithm [6] to optimise parameter values. Due to the computationally intensive nature of MOE algorithms, is will be executed in a distributed fashion.

In this research we are concerned with missions that involve picosatellites flying in close formations to each other, considering picosatellite networks to be a special case of wireless sensor networks (WSNs), i.e. a space-

Second NASA/ESA Conference on Adaptive Hardware and Systems(AHS 2007)
0-7695-2866-X/07 $25.00 © 2007 IEEE

based wireless sensor network. *Formation flying* missions are of particular interest from the point of view of using wireless COTS protocols for inter-satellite connectivity due to the close proximity of the nodes in the network.

The paper is structured as follows. Section 2 discusses issues related to approaches to distributed computing, picosatellite mobility and on-board computing support. Section 3 presents initial work on the software architecture. Section 4 gives details of the test-bed design with CubeSat. Section 5 concludes the paper.

2. Distributed Computing Platform Considerations

2.1. Approaches to Distributed Computing

There are two different approaches to distributed computing that are applicable to picosatellite networks: *address-centric* and *data-centric*. Each of these approaches drives a particular set of possible communication architectures and styles. For example an address centric architecture opens the possibility of using a client-server architecture or a mobile agent based architecture. On the other hand a data-centric approach may use architectures such as publish-subscribe.

We envisage that picosatellite networks, as WSNs, "are largely data-centric, with the objective of delivering time sensitive data, in a timely fashion to the required destination" [7]. A data-centric architecture has numerous advantages over an address-centric architecture as follows:

- it facilitates in-network processing of the data and therefore leads to reductions in power consumption, bandwidth and memory;
- it could enable merging of the intra-satellite network with the inter-satellite network;
- it is fault-tolerant by nature as data is the focus of the network and not the nodes themselves.

A communication architecture is evaluated for a particular application against a set of criteria. In order to evaluate the effectiveness of the data-centric and the address centric approaches with respect to picosatellite networks the following set of criteria is identified:

- impact of mobility,
- scalability,
- fault-tolerance/reliability,
- determinism/real-time performance and
- efficiency (memory capacity, computing performance, power consumption and bandwidth).

Due to space limitation we report here the results from only one of these evaluation tests – the impact of mobility.

2.2. Impact of Satellite Mobility

A unique aspect of picosatellite networks that differentiates them from other wireless sensor networks is the mobility of the satellite nodes. Research in wireless sensor networks has mostly focused on static nodes and issues related to mobility are not well addressed. Picosatellite networks are different from WSNs, in that the nodes are mobile and their mobility is highly predictable.

Several formation flying configurations could be used for picosatellite networks, each having its advantages and disadvantages. The simplest one is the so called leader-follower configuration (also referred to as string-of-pearls) where the satellites have the same orbital parameters, but are available for communication at different times. The same-ground-track formation, termed “ideal” by NASA Goddard [8], is the one in which two or more satellites have identical ground tracks.

A simulation with Omnet++ [9] was carried out to determine the effect that the relative satellite mobility would have on the performance of a data-centric architecture. The simulated formation consists of 15 satellite nodes in two different configurations: leader-follower and same-ground-track. The communication method between the nodes is broadcasting. A publish-subscribe communication architecture is used which utilizes a directed diffusion protocol.

Directed diffusion is a data-centric dissemination protocol that provides mechanisms: (a) for a sink node to flood a query towards the sensors of interest, and (b) for intermediate nodes to set up gradients for sending data along the routes towards the sink node [10]. The simulation results in Figure 1 show that the number of hops, which is proportional to the communications energy consumption, varies with the subscription time. It can be concluded from Figure 1 that communication power and bandwidth may be saved by appropriately adjusting the publication and subscription times. This could be used to design an energy-efficient protocol for data dissemination in picosatellite networks.

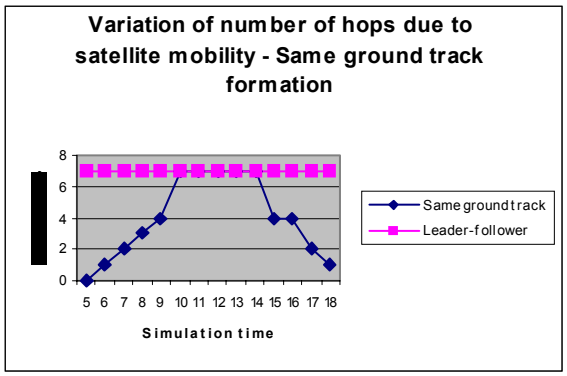

Figure 1. Perfomance evalution of the publish-subscribe architecture.

2.3. On-Board Computing Support for Distributed Processing

The on-board high-performance computing is based on a reconfigurable system-on-a-chip platform, which can be tailored to suit different data processing needs [11]. All sensed data (raw or processed) are submitted to the cluster-head satellite, and then passed to the ground station. Smart onboard antennas, adequate power budgeting and high performance inter-satellite links are required to account for the distances within a cluster.

A generic SoC-based computing platform for development of on-board controllers is proposed [12]. The SoC is implemented using Xilinx Virtex Field Programmable Gate Arrays (FPGAs), which support partial run-time reconfiguration [13]. The central processing unit of the SoC is the LEON microprocessor by Gaisler Technologies, which is a SPARC V8 soft IP core written in VHDL [14]. The LEON processor IP core has flown in space as part of the Venus Express imaging payload controller and is adopted by ESA as the main CPU for future on-board computers. It incorporates multiple processor usage, fast AMBA2 AHB bus, 8/16/32-bit memory controller, 16-bit I/O and many peripherals. The LEON3 processor can run at up to 200 MHz, consuming around 250 mW.

Image compression and multi-objective evolutionary algorithms are computationally intensive. To speed up the execution of the image compression algorithm it will be implemented as an IP core, which will be integrated as a peripheral to the LEON processor.

One approach to hardware acceleration of MOE algorithms is to implement some functionality of the algorithm as an IP core, and to run the remaining functionality as software on the LEON processor. Optimal software-hardware split is achieved by profiling. Here we take the MOE algorithm in [15] as an example. The MOE algorithm is initially programmed in C. It runs on the LEON3 processor under the RTEMS real-time operating system (RTOS). The LEON3 processor is clocked at 40 MHz. There are a large number of 'multiply' (MUL) and 'divide' (DIV) operations in the MOE algorithm. The LEON3 processor is configured according to the following three options:

- soft 'MUL/DIV' – software 'multiply' and 'divide' operations (fixed and floating point)
- hard 'MUL/DIV' option - hardwired fixed-point 'multiply' and 'divide' operations (IP core)
- floating point unit (FPU) + hard 'MUL/DIV'

The C program of the MOE algorithm is compiled using the RTEMS cross-compiler. Table 1 shows the profiling results, where the second column is the execution time of the MOE algorithm and the third column - the time spent on the 'MUL/DIV' operations.

Table 1. Profiling results of the MOE algorithm for LEON3

Option	Time (msec)	'MUL/DIV' (msec)	Percentage (%)
Soft 'MUL/DIV'	1.54	0.159	10.32
Hard 'MUL/DIV'	0.96	0.038	3.96
FPU and Hard 'MUL/DIV'	0.19	0	0

It can be seen from Table 1 that the soft implementation of the 'MUL/DIV' is the slowest approach. The MOE program runs 1.6 times faster when the hardware IP core is integrated. The third option in Table 1, when both the FPU and the hard 'MUL/DIV' are involved, is the fastest one improving the performance 8.1 times.

3. Software Architecture

3.1. Operating System

Several COTS real-time operating systems are employed in satellite on-board computers (OBC) including VxWorks and QNX. Mostly used for on-board data handling and attitude determination these operating systems are quite heavy and expensive. Open-source RTOS such as RTEMS are also flown on some missions. The CubeSat kit comes with an RTOS called Salvo. This operating system (OS) is designed for low memory embedded systems and can be

configured depending on the needs of the picosatellite node.

The distributed computing tasks will be executed by the LEON-based SoC controller, which will be located in a payload subsystem of the picosatellite node. An operating system is required to run on LEON supporting middleware and applications. It is identified that the OS should have the following features:

- Open-source software
- Small memory footprint
- Real-time functionality

Open-source software is preferred because it offers the possibility to configure the components of the OS, while commercial RTOS, like VxWorks and QNX, can not be modified. TinyOS is an open-source OS widely used in WSN environments. TinyOS, adopts the component-based approach to system development and is extremely efficient with regard to footprint, power consumption and memory utilization. A major downside of TinyOS is that it does not support real-time operation and would have to be customised to include this important feature for picosatellite networks.

In order to select an OS for the distributed computing platform three open-source operating systems that support the LEON processor are investigated: eCos, RTEMS and Snapgear embedded Linux. A simple kernel that includes all the necessary components to run basic applications is tested for each of the three OS. The real-time capability of the OS is evaluated based on the measured preemption latency. The test results with respects to footprint and preemption latency are presented in Table 2.

Table 2 shows that RTEMS has a minimum footprint, and the SnapGear Linux has the largest. Table 2 also shows that the preemption latency of RTEMS is very small (30% and 9% of the latency of eCos and SnapGear Linux, respectively). This makes RTEMS the winning real-time operating system for the distributed computing platform.

Linux can be configured towards an application both with and without a memory management unit (MMU). RTEMS and eCos, however, do not have MMU and thus do not have any virtual memory/backup protection. However, RTEMS has features, which are not available in SnapGear and eCos, for example, a multiprocessor manager which supports a simple and flexible real-time multiprocessing mechanism e.g. for sharing data and providing global resources between many different types of processors.

Table 2. Operating systems comparison

Operating System	Minimum Footprint (KB)	Preemption latency (μsec)
SnapGear Linux (2.6 kernel)	> 1024	412
eCos	> 50	127
RTEMS	< 10	38

3.2. Middleware Design

Middleware is important for managing distributed systems. Therefore, investigation is undertaken into one of the most widely used middleware technologies for distributed systems – CORBA [16]. An implementation of CORBA, called OmniORB, was selected for evaluation purposes. Table 3 shows a minimum implementation of the OmniORB with the RTEMS operating system. The components include RTEMS, C++ IOSTREMS, TCP/IP stack, IEEE 802.11 driver, OmniORB 2 and a dynamic library.

Table 3. Implementing CORBA with RTEMS

Component	Size (KB)
RTEMS	191
C++ IOSTREAMS	64
TCP/IP stack	527
802.11 driver	319
Omni ORB 2.0	483
Dynamic library	1,55
Total	1,739

It can be seen from Table 3 that the footprint of such an implementation amounts to 1.7 MB, which is rather large for an embedded system and therefore makes CORBA inappropriate for our project.

A completely different approach is offered by middleware technologies based on the publish-subscribe commutation scheme. As part of a separate project running in parallel with ESPACENET a middleware design referred to as Middleware for Inter-Spacecraft Applications (MISA) is developed, which is illustrated in Figure 2.

Contrary to CORBA that is based on the client-server paradigm, MISA is a data centric middleware that uses the publish-subscribe model, in which each

message is associated with a specific topic or event. Applications interested in the occurrence of a specific event may subscribe to messages for that event. When the awaited event occurs, the process publishes a message announcing the event or topic. The middleware message system distributes the messages to all its subscribers. Usually the only property a publisher needs in order to communicate with a subscriber is the name and definition of the data. The publisher does not need any information about the subscribers, and vice versa.

As long as the interested applications know *what* data is being communicated, a publish-subscribe infrastructure is capable of delivering that data to the appropriate nodes without having to set up individual connections. Publishers are responsible for gathering the appropriate data and sending it out to all registered subscribers. Subscribers are responsible for receiving data from the appropriate publishers and presenting the data to the interested user application. Agents in MISA are built using MISA's component-based system based on TinyOS.

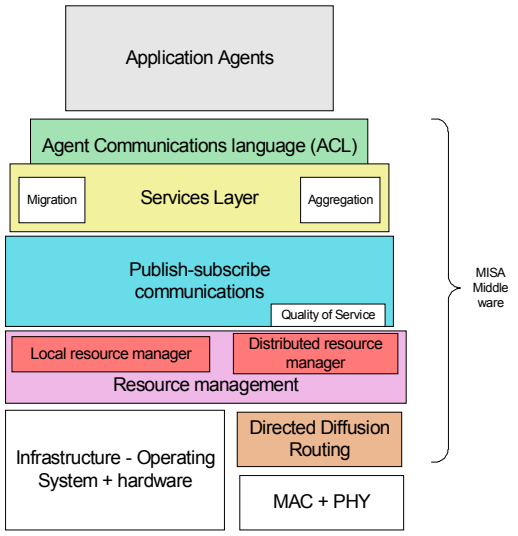

Figure 2. Structure of the MISA middleware

There are three types of publish-subscribe systems based on the addressing scheme that they adapt: group-based addressing, subject-based addressing and content-based addressing. The content-based approach is of considerable importance to sensor networks as it offers a high degree of flexibility in the formulation of the subscription criteria. Therefore the content-based approach is adopted in MISA.

The publish-subscribe architecture suits well the challenging nature of embedded communications. Finding the right data and sending the data at the right time is straightforward. Publish-subscribe can be efficient because the data flows directly from source to sink without requiring intermediate servers. Multiple data sources and sinks are easily defined within the model, which facilitates the implementation of redundancy and fault tolerance measures.

3.3. Mobile Software Agents

The agent paradigm has been used to describe complex networked environments in a behavioural way, addressing the new network complexities in massively distributed computing systems. TechSat-21 was the first satellite system encountered to suggest using agent technology (called ObjectAgent) in space and how they could be utilised [17]. A re-designed version of the ObjectAgent software called Messaging Architecture for Networked and Threaded Applications (MANTA) has been used for a variety of projects, including formation flying. The most recent application of MANTA was for a contract with NASA to develop a reconfigurable, decentralized guidance and control architecture for formation flying missions. It is still in the development stages of implementation and therefore not considered for this project.

There are many potential benefits to using multi-agent software systems on board spacecraft [18]. These include (but not exhaustively) the following:

- increasing the level of autonomy through decomposition of high-level goals;
- increasing the flexibility and adaptability of flight software through dynamic agent uploads;
- improving the reliability of spacecraft and clusters of spacecraft by incorporating fault detection at both high and low levels.

Multi-agent systems are used primarily in:

- Building blocks for control systems to parallelize a device's computing functions
- Communication techniques and protocols for improved reliability and robustness

Control agents systems based on MAS are to be used to control high level variables, such as large routing tables and decision making. The communicative agents must be able to deal with and effectively send megabytes of data, found in satellite imaging, but also small telemetry or message packets, typically a few kilobytes. The aim is to reliably and robustly transport messages and services across an ad-hoc network of picosatellites using agents [19].

To accommodate the distributed computing environment, a Java Virtual Machine (JVM) for embedded devices is to be used. JVM can provide a communication medium between various heterogeneous platforms [20]. Two environments have been widely adopted for agent development:

Federation for Intelligent Physical Agents OS (FIPA-OS) [21] and JADE [22]. Both systems provide agent based management services for collaborative development/co-ordination on a base platform for interoperability and testing purposes.

Reduced platforms for embedded devices have been including JADE-LEAP following FIPA's Nomadic Application Specification [23]. The JADE-LEAP development environment has been used extensively as a platform to develop new systems and novel application areas for resource-constrained devices using wireless links.

4. Test-Bed Design with CubeSat

A "technology demonstration" picosatellite mission in low Earth orbit (LEO) is proposed, which will demonstrate applications of advanced terrestrial technologies in space as part of the ESPACENET project. One of these technologies is distributed computing over IEEE 802.11-based inter-satellite links (ISLs).

A picosatellite design based on the CubeSat Development Kit is in the process of construction. The picosatellite will serve as a basic node in a wireless network to test and design a distributed agent-based system. The picosatellite computing and power resources are very limited, e.g. the total power is less than 3 Watt, which makes the picosatellite node comparable to wireless embedded devices.

4.1. Picosatellite Platform Based on the CubeSat Bus

The CubeSat platform is a 10 x 10 x 10 cm standard bus structure weighing at around 1 kg, which is compatible with the PC104 format. The CubeSat bus also comes in double (2U) and triple unit (3U) sizes to conform to the P-POD deployment mechanism. Foreseeable problems include limited amount of energy from battery cells or solar panels for communicating to ground or to operate payloads. Typical CubeSats operate between 1 to 2 Watt of total satellite power budget.

The picosatellite prototype includes the CubeSat flight worthy onboard computer based on the MSP430 microcontroller, an imaging payload implemented in a Xilinx Virtex 4 FPGA, and an IEEE 802.11 communications board. The picosatellite nodes are to be computationally able to run a MOE algorithm to optimise the network performance. This will require additional hardware and software resources to be fitted in the constrained CubeSat structure. Current CubeSat missions show that reliability and simplicity of the bus are key requirements for mission success whereby more complex data processing systems are designed as separate payloads. We propose a satellite bus architecture that implements the technology testing experiments as payload subsystems, as shown in Figure 3.

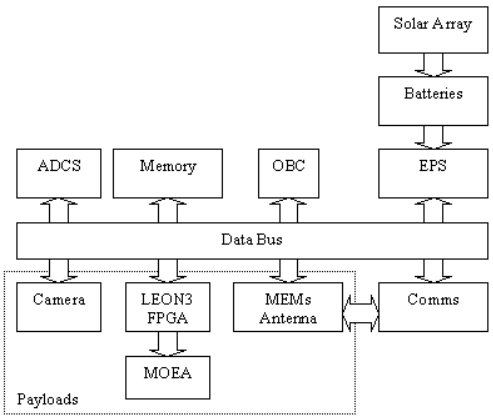

Figure 3. ESPACENET satellite architecture

The OBC is to be a simple microcontroller with a reduced embedded operating system or task manager to control the overall satellite. The payloads are as follows:

- Xilinx Virtex-4 FPGA with the LEON-based SoC controller,
- COTS camera and mass memory block
- MEMS antenna with wireless IEEE 802.11 ISL board and a GPS receiver.

4.2. Case Study: Imaging

Wireless IEEE 802.11 USB boards from Elcard [24] are employed to prototype a 3-node distributed computing platform, based on the client/server scheme. The standard TCP/IP protocol is used for data communication. Additional middleware is currently under development and will be used with a more complex demonstrator in the future. Figure 6 shows the diagram of the prototyped computing platform. A three-node satellite network is emulated using IEEE 802.11b wireless ISLs. One of the nodes is a server, and the other two are clients.

Distributed processing of image compression is demonstrated using the Lena benchmark image as an example. The image compression core is based on the JPEG 2000 lossless algorithm. The application works as follows. The server terminal collects information from the client satellites. The sensed data could be telemetry data, image, power, attitude, and GPS data. These data are then distributed to the clients via TCP/IP over the IEEE 802.11 wireless links to be processed according to the server's instruction. After

the data are processed, the results are sent back to the server.

We assume that the three picosatellites are distributed in LEO orbit using a 'string-of-pearls' formation, in which all satellites are in the same orbit; and the master satellite is located at an equal distance to the slave satellites. The master satellite evaluates and compares the distance, the communication power and the link quality (such as bit error rate) with respect to both slave satellites. Depending on these parameters, the master satellite decides what portion of the image to be distributed to each client satellite (Figure 4).

Because the satellites are located in the same orbit, the ISL ranges will remain relatively unchanged. The Satellite Tool Kit (STK) is used to simulate the satellite network in LEO orbit based on an omni-directional antenna. Table 4 shows the Effective Isotropic Radiated Power (EIRP) of the transmitter, the Received Isotropic Power (RIP) is of the receiver and the bit error rate at data rate of 1 Mbps for three distances. It can be seen that the link quality at smaller distances is good but it deteriorates significantly at distances of the order of 100 km. However, if a directional antenna is used, the BER is reduced to 10^{-25}.

In our example, we assume that the ISL range is 10 km. The image is split into two even parts and sent to the client satellites. After the images are compressed, they are sent back to the master satellite, and merged. The raw data of the Lena image is 512 x 512 bytes. After the split, each slave satellite gets 512 x 256 bytes. After compression, one satellite returns 79,358 bytes and the other returns 74,734 bytes. The total size of the compressed image is 154,092 bytes, which is exactly the same size as when compressing without distributed processing.

Table 4. Quality of wireless inter-satellite link based on IEEE 802.11b

Parameters	1 km	10 km	100 km
EIRP (dBW)	0	0	0
RIR (dBW)	-100	-120	-140
BER	10^{-25}	10^{-25}	1.3×10^{-2}

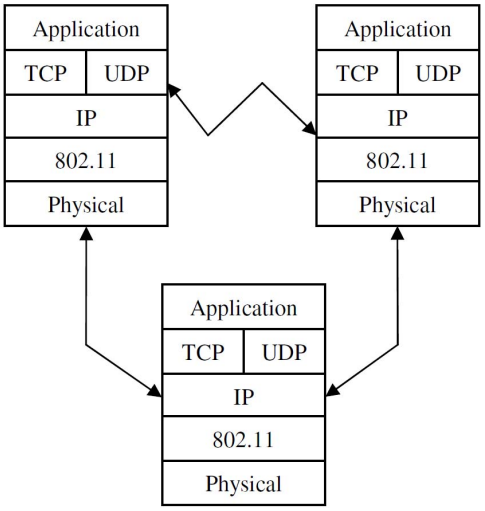

Figure 4. Distributed computing platform based on IEEE 802.11 for three nodes

5. Conclusions

In this paper we present the initial investigation into possible solutions to the challenging problem of implementing distributed computing technology in picosatellite networks. Several key technologies are addressed, including RTOS, middleware design, communication architectures, autonomous mobile agents and inter-satellite links.

Details of the on-board computing to support distributed processing are also reported. A network with three satellites, which is modelled in STK, demonstrates a client/server based distributed image compression task.

The adopted approach to developing the picosatellite platforms is to use COTS components. The CubeSat structure can accommodate the industrial standard PC104 boards. The main products for developing the picosatellite demonstrator include a PC104 FPGA board for the SoC implementation; a PC104 wireless board for inter-satellite links; and a PC104 GPS board for determining the position of the satellites.

Acknowledgements

This research is sponsored by the EPSRC under grant EP/C546318/01

References

[1] T. Vladimirova, X. Wu, K. Sidibeh, D. Barnhart and A.-H. Jallad. Enabling Technologies for Distributed Picosatellite Missions in LEO - Proceedings of 1st NASA/ESA Conference on Adaptive Hardware and Systems (AHS 2006), pp. 330-337, 15-18 June, Istanbul, Turkey, IEEE Computer Society

[2] C. I. Underwood, V. J. Lappas, A. da Silva Curiel, M. Unwin, A. M. Baker, and M.N. Sweeting, Using PALMSAT Picosatellite Technologies to Meet Mission Scenarios, in Proceedings of 55th International Astronautical Congress, 2004, Paper IAC-04-P.5.A.01.

[3] CubeSat Kit, http://www.cubesatkit.com/

[4] DNEPR Launch 1, http://cubesat.atl.calpoly.edu/pages/missions/dnepr-launch-1.php

[5] DNEPR Launch 2, http://cubesat.atl.calpoly.edu/pages/missions/dnepr-launch-2.php

[6] N.Haridas, E.Yang, A.T.Erdogan, T.Arslan, N.Barton, A.J.Walton, J.S.Thompson, A.Stoica, T.Vladimirova, X.Wu, K.D. McDonald-Maier, W.G.J. Howells. ESPACENET: A Joint Project for Evolvable and Reconfigurable Sensor Networks with Application to Aerospace–Based Monitoring and Diagnostics – Proceedings of 6th International Conference on Recent Advances in Soft Computing (RASC2006), Ed. K.Sirlantzis, pp. 410-415, 10-12 July 2006, Canterbury

[7] J. A. Stankovic, T. F.Abdelzaher, C. Lu, L.Sha, J. C. Hou. Real-Time Communication and Coordination in Embedded Sensor Networks – Proceedings of the IEEE, Vol.91, n 7, pp. 1002-1022, July 2003.

[8] J. B. Mueller, D. M. Surka, B. Udrea. Agent-Based Control of Multiple Satellite Formation Flying, Proceedings of 6th International Symposium on Artificial Intelligence, Robotics and Automation in Space, June 18-21,2001, Montreal, Canada.

[9] A. Varga. The Omnet++ Discrete Event Simulation System, in Proceedings of the European Simulation Multiconference. Prague, Czech Republic: SCS – European Publishing House, June 2001, pp. 319–324.

[10] C. C. Intanagonwiwat, R. Govindan and D. Estrin, Directed diffusion: A scalable and robust communication paradigm for sensor networks, in the Proceedings of the 6th Annual ACM/IEEE MobiCom'00, Boston, MA, August 2000

[11] C.H. Wang, et al, "High Throughput and Low Power FIR Filtering IP Cores", IEEE International SOC Conference (SOCC 2004), pp. 127-130, Santa Clara, California, September 12-15, 2004.

[12] T.Vladimirova, M.N.Sweeting. "System-on-a-Chip Development for Small Satellite On-Board Data Handling", Journal of Aerospace Computing, Information, and Communication, vol. 1, n 1, pp. 36-43, January 2004, AIAA

[13] X. Wu and T. Vladimirova. An Evolvable and Reconfigurable System-on-Chip Architecture for Future Small Satellite Missions - Proceedings of 9th Military and Aerospace Applications of Programmable Logic Devices and Technologies International Conference (MAPLD'2006), P-1016, September 26-28, 2006, Washington DC, US, NASA

[14] Jiri Gaisler, "GRLIB IP Library User's Manual (Version 1.0.4)", Gaisler Research, 2005.

[15] K.Deb, A.Pratap, S.Agarwal, and T.Meyarivan, A fast and elitist multi-objective genetic algorithm: NSGA-II, IEEE Transaction on Evolutionary Computation, vol. 6, n 2, pp.181-197, 2002.

[16] The Common Object Request Broker: Architecture and Specification. http://www.omg.org/docs/formal/01-09-34.pdf.

[17] P. Zetocha, L.Self, R.Wainwright, R.Burns, R.; M.Brito, D.Surka, Commanding and Controlling Satellite Clusters, IEEE Intelligent Systems Magazine, vol. 16. n 6, pp: 8-13, November/December 2000.

[18] A.-H. Jallad and T. Vladimirova, Distributed Computing for Formation Flying Missions, Proceedings of the ACS/IEEE International Conference on Computer Systems and Applications (AICCSA-06), pp: 616-620, 8-11 March 2006, UAE.

[19] FIPA Work Plan, Agents in Ad-hoc Environment, www.fipa.org/docs/wps/f-wp-00020/f-wp-00020.pdf

[20] C.P.Bridges and T.Vladimirova. Autonomous Software Agents in Wireless Embedded Systems, Proceedings of the 3rd UK Embedded Systems Forum, Durham University, p. 167, 2007, IET.

[21] FIPA-OS @ Sourceforge, http://sourceforge.net/projects/fipa-os/

[22] JADE Whitepaper, http://jade.tilab.com/papers/2003/WhitePaperJADEEXP.pdf

[23] FIPA Nomadic Application Support Specification, www.fipa.org/specs/fipa00014/SI00014H.pdf

[24] WUSB20G B1 Series USB 802.11b/g WLAN Modules, http://www.elcard.fi/pdfs/DSWUSB20GB1-01.pdf