The 2010 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 18-22, 2010, Taipei, Taiwan

A Platform for Networked Robotics

Eleri Cardozo, Eliane Guimarães, Lucio Rocha, Ricardo Souza, Fernando Paolieri and Fernando Pinho

Abstract— **Most of the environments where mobile robots are deployed offer a comprehensive networking infrastructure. In order to take part of these environments, mobile robots must enhance not only their communication capabilities, but also their interaction capabilities. Current robotic frameworks are oriented to gather data from the robotic sensors and to act on the robot's actuators. In other words, such frameworks are oriented to perform embedded control functions. In order to provide mobile robots with enhanced interaction capabilities robotic applications must rely on networking solutions focusing on integration, communication, and security. This paper presents a platform for networked robotics that favors the integration of mobile robots with other networked devices, providing a secure and federated access to the mobile robots.**

I. INTRODUCTION

Many mobile robotic frameworks such as ARIA [1], Player [2], and Orca [3] are inherently distributed in the sense that a client program can interact over the network with a robot running these frameworks. Mobile robotic frameworks employ specialized network protocols for supporting client-server interactions, a design decision that restricts the integration of robots with the organization's distributed applications. The first restriction is related to software development and integration. In order to talk specific network protocols, client-side code must employ the API (Application Programming Interface) supplied by the framework. As a consequence, modern communication devices such as cell and soft phones that do not support these APIs are hard to integrate into a mobile robotics application. Usually this integration requires a mediator running in between the accessing device and the mobile robot (e.g., a Java servlet) with obvious drawbacks in terms of performance and complexity.

The second restriction relates to the way networks operate nowadays. A common network design employs private IP (Internet Protocol) addresses and firewalls. A machine configured with private IP address is able to communicate with other networks only through routers configured with specialized forwarding functions that perform protocol and address translations. HTTP (Hypertext Transfer Protocol) proxies and NAT (Network Address Translation) are examples of such functions. As proxy and NAT functions are restricted to well known networking protocols, the specialized protocols employed by the robotic frameworks are commonly blocked when private IP addressing is employed. Firewalls also block specialized protocols employing port numbers different from those allowed to pass the firewall. A costly solution to cross proxies, NAT and firewalls is to employ tunneling schemes where protocol messages are transferred inside messages of another protocol not blocked by these networking functions (e.g., the Secure Shell protocol). Tunneling causes extra overhead and loss of functionality.

Finally, a more severe restriction is related to security, a requirement not addressed by the present robotic frameworks. These frameworks are not able to distinguish access from different users, domains, and applications; to verify if users or applications are authenticated and authorized to access the resource; and to secure the communication in order to avoid security threats on the robotic resources.

This paper presents a distributed software platform that favors interdomain communication and interaction with mobile robots. The platform is built above robotic frameworks, enlarging the robot's communication and interaction capabilities by employing open protocols, services, and security solutions adopted by the modern distributed applications. The paper is organized as follows. Section II presents the architecture of the platform. Section III addresses issues related to the design of the platform. Section IV details the REALabs platform. Section V presents qualitative and quantitative evaluations of the platform. Section VI presents some related works. Finally, section VII concludes the paper.

II. PLATFORM ARCHITECTURE

Fig. 1 shows the four main packages of the platform in the UML (Uniform Modeling Language) notation. The figure shows the main components a network robotics platform must provide and serves as a guideline for platform design. The Embedded package is composed of components that run on the mobile robots. These components are microservers able to run on embedded processors with limited processing power. Typically, microservers process the HTTP protocol, and, in some cases, other protocols such as SIP (Session Initiation Protocol) and SOAP (Simple Object Access Protocol). The microservers process the requests and act on the robot, usually, through a robotic framework installed on the robot.

The Protocol Handler package performs functions such as security checking, proxying, and network address translations. The components in this package inspect the request to the resource (robot, camera, etc), perform security checking, and route the message to the resource's microserver. An example of such components is an HTTP proxy agent that verifies if the message carries the proper security credentials, and forwards the request to a resource placed on a private addressed network. The proxying process may translate

E. Cardozo, L. Rocha, R. Souza and F. Pinho are with the School of Electrical and Computer Enginnering, University of Campinas, Brazil. eleri@dca.fee.unicamp.br

E. Guimarães and F. Paolieri are with the Information Technology Center Renato Archer, Campinas/SP, Brazil. eliane.guimaraes@cti.gov.br

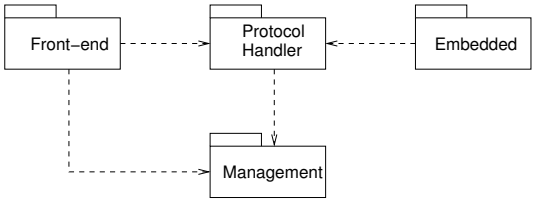

Fig. 1. Main packages of the proposed platform (dashed arrows model dependencies)

protocols, for instance, an original HTTPS (HTTP Secure) request is translated to plain HTTP before be forwarded to the resource. In summary, the Protocol Handler package reliefs the resources from performing functions that require extensive computation such as data encryption/decryption, protocol translation, and security checking.

The Front-end package holds components for supporting interactions over the network between the robotic application and the mobile robots. These components offer a high level interface for robot manipulation in different programming languages and platforms, and hide the underline interaction protocols from the robotic application. The functionalities provided by this package are similar to those provided by the robotic frameworks, except that the components can run on Web browsers, soft phones, and any device able to access the network. Importantly, communication generated by the Front-end components will not be blocked by NAT boxes and firewalls.

The Management package aggregates components that perform management actions at the level of federations, domains, and resources. Management of federations allows the establishment of an entity that acts as a trusted party for the domains. Management of certificates is a typical management function at the federation level. Management of domains allows the creation of trusted relationships among a set of domains through the federation. Management of users, credentials, and certificates are typical management functions at the domain level. Management of resources keeps track of the resources maintained by the domain. Management of access and resource reservations are typical management functions at the resource level.

III. PLATFORM DESIGN ISSUES

In this section we show how the platform architecture described above can be designed and implemented. The division of the platform into packages helps to choose software products able to support the packages' components.

A. Embedded Package

The Embedded package aggregates microservers that run on the robot's internal processors. Although large mobile robots have enough processing power to run full featured servers, this is not the case of small and mid-sized robots. For these devices the microservers must be compact enough to execute on the robot's internal processor. There are at least three design decisions regarding the microservers:

1) How client applications interact with the microservers?
2) How the microservers interact with the robot hardware?
3) How real time issues are addressed?

The interaction client-microserver must rely on general purpose and widespread protocols. HTTP and SIP are natural candidates for three reasons: i. they are simple, text-based protocols; ii. proxies simplify the processing of these protocols at the server-side; iii. they already have powerful client applications available, e.g., Web browsers in case of HTTP and the internet communicators in case of SIP.

Client-microserver interactions can adopt the REST (Representational State Transfer) interaction pattern [4]. REST employs messages to inspect and change the state of objects maintained by a Web server. Objects are data structures representing logical entities (e.g., databases) or physical entities (e.g., robots). Messages can be based on any protocol, although HTTP and SOAP are the preferred ones. REST over HTTP employs the HTTP PUT, GET, POST, and DELETE messages for state installation, retrieving, updating, and dropping, respectively. In most cases the resource installs a state during startup and never drops it (the case of robots, for instance). In such cases, only the GET and PUT messages are employed. We favor REST over HTTP since it simplifies the microservers when compared to REST over SOAP.

The interaction microserver-robot hardware is usually intermediated by a robotic framework such as Player or ARIA. The framework simplifies the translation of messages into actions on the robot. For example, an HTTP POST message carrying data indicating a new state of the robot (e.g., a new pose) must be translated into activations performed by the robotic framework. As usually the robotic frameworks are called from C/C++, an HTTP microserver must be able to call C/C++ functions during the processing of requests, a mechanism called server-side extension.

Real time issues must be addressed in the design and implementation of the microservers. HTTP and SIP processing are fast but this is not enough. In order to favor real-time processing, the microservers must also present some properties such as non-blocking and preemptive operation, priority-based request processing, and assurance of timing constraints related to the execution of periodic tasks and interrupt processing.

Multithreaded microservers running above real-time kernels can reach near real-time performance. For example, real-time Linux [5] extensions provide privileged execution to critical tasks running as kernel extensions (modules). In such kernels, a microserver implemented as a module can run without be blocked by processes executing at user-space. Requests are processed as threads scheduled according to the request priority (e.g., an emergency stop request preempts any other ongoing requests).

B. Protocol Handler Package

The Protocol Handler package must rely on a router or server in between the client application and the microservers. NAT and firewall functions are performed by low level kernel modules such as *iptables* on Linux routers. Proxying can be

performed by servers such as Apache HTTP server (httpd) and Mobicents SIP servlets. In addition to proxying, these servers must support extensions for authentication and authorization checking. Apache httpd allows server-side extensions that can perform extra processing before the request is forwarded to the resource. Security checking usually demand interactions with components of the Management package, for instance, resource reservation and authentication services.

C. Front-end Package

The components in the Front-end package can be offered for any programming environment able to generate HTTP requests and process the reply. This is common to a wide range of today's programming environments. The components in this package offer their functionalities in the form of object-oriented APIs for programming languages supporting this concept (C++, Java, and Python); Web components for applications running inside Web browsers; and software artifacts particular to some specialized environments such as Matlab functions, LabView blocks, TinyOS's NesC components (for sensor networks).

D. Management Package

The management functions are commonly implemented as Web services with secure access (via HTTPS) and subject to authentication and authorization. Such services are supported by Web containers such as Apache Tomcat and Sun's Glassfish. A common requirement for the management functions is security. Basically, security comprises user authentication and authorization. Authentication can rely on systems such as Shibboleth [6] and Open SSO [7] that support federated authentication, known as Single Sign On (SSO). SSO allows user credentials be securely exchanged among the federated domains.

Authorization usually comprises the verification if authenticated users are allowed to access a given resource. This verification can be based on policies. Typical authorization policies require in advance resource reservation and a valid access session. In these cases a reservation and access services must be offered by the Management package. A reservation service allows users to set a time slot for resource manipulation. An access service checks if the access policies (e.g., reservation) is fulfilled, and, if the case, supply the user with an access session identifier. Each interaction with the resource carries this identifier. The Protocol Handler package verifies if the access session identifier is a valid one. If not the case, the access to the resource is blocked.

IV. THE REALABS PLATFORM

REALabs is a software platform for networked robotics in line with the architecture and design presented above. Fig. 2 shows the components of the REALabs platform, the software packages they require, and their placement on processing nodes (boxes) in a loose UML deployment diagram.

REALabs offers two HTTP multithreaded servers interacting with the ARIA and Player robotic frameworks. The

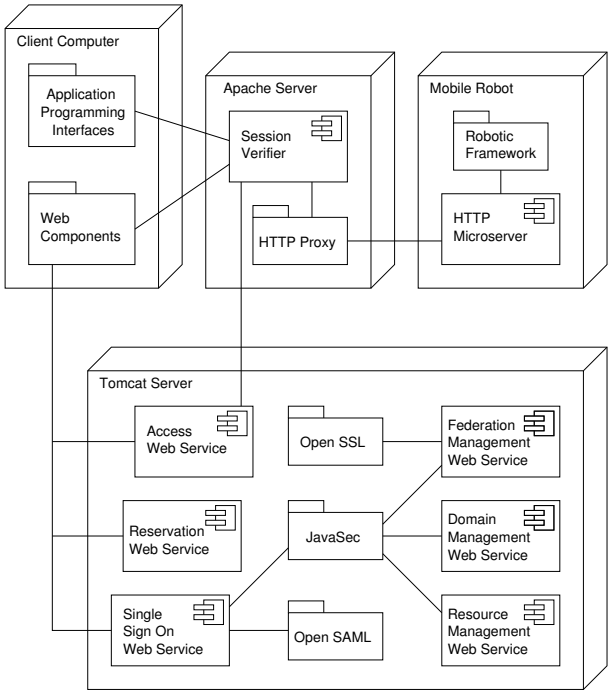

Fig. 2. Components of the REALabs platform

microservers run above Linux without real-time extensions. The REST interaction style is supported. Operations that do not change the robot's state (e.g., sensor readings) are performed through HTTP GET messages while operations that change the robot's state (e.g., movements) are performed through HTTP POST messages. All HTTP operations return a XML document containing the requested data (e.g., a laser scanning), a notification (e.g., a movement completion), or an exception (e.g., operation not supported or error). Although XML can be transported more efficiently over UDP (User Datagram Protocol), this solution has two main drawbacks: control messages require a reliable protocol functionally similar to TCP (Transfer Control Protocol, employed by HTTP), and firewalls and NAT boxes usually block UDP traffic.

The Protocol Handler package relies on the Apache httpd server. HTTP proxy functions are directly supported by this server. The REALabs platform provides an extension module to this server that verifies if the user accessing a resource has a valid access session already established (the Session Verifier module). The establishment of access sessions is subject to authentication and authorization. HTTP proxying and session verification are independent of the message contents as these functions inspect only the HTTP header.

The Management package employs Web services for federation, domain, and resource management. These services are supported by the Apache Tomcat application server. Federation management relies on certificate management provided by JavaSec, a native Java package for XML document signing and signature verification, and OpenSSL, an extensive C-based security package.

The management of domains employs a secure Single Sign

On service for user authentication. This service is based on SAML (Security Assertion Markup Language) [8], a standard for exchanging user credentials among the domains. This implementation is very lightweight when compared with Shibboleth or Open SSO and is fully integrated into the platform code. Our SSO implementation employs Open SAML, a Java-based SAML API for processing SAML assertions, in conjunction with JavaSec for assertion signing.

The management of resources offers an access service for session establishment, maintenance and termination. An access session can terminate explicitly by the user, implicitly by inactivity, or when the reservation time has expired. This reservation service allows a single or a group of resources be reserved on time periods that the administrator opens for reservation.

The Front-end package in REALabs offers a set of common functionalities found on ARIA and Player robotic frameworks. These functionalities are grouped into categories such as rangefinder sensoring, locomotion, and image acquisition. The REALabs platform offers APIs in the following programming languages: C++, Java, Java2 ME (Micro Edition, for cell and smart phones), Python, and Matlab. In addition, clients in any programming language can directly generate an HTTP requests to the robots and parse the returned XML documents. Two key differences between the APIs provided by the platform and those provided by the robotic frameworks are in order:

1) The REALabs APIs generate HTTP or HTTPS requests that are not blocked by NAT boxes and firewalls.
2) The return of the operation is always an agnostic XML document.

The advantage of returning XML documents in HTTP interactions with mobile robots is important for networked robotics applications. XML processing is today embedded into practically all programming languages and environments, including Web browsers.

V. PLATFORM EVALUATION

This section provides both qualitative and quantitative evaluations of the REALabs platform.

A. Qualitative Evaluation

The REALabs platform was employed to build a mobile robotics Web-accessible laboratory (WebLab) described in [9] and in [10]. The WebLab is operated in two domains: the School of Electrical and Computer Engineering of the University of Campinas (FEEC), and the Information Technology Center Renato Archer (CTI), both located in Campinas, Brazil. The two domains are connected by a high speed network (the KyaTera network) and form a federation with the objective of sharing part of their mobile robotic infrastructures.

The WebLab employs one MobileRobots' Pioneer P3-DX at each domain. Robots are equipped with laser scanner, sonars, on-board camera, and gripper. In addition to the mobile robots, the WebLab operates one Axis 214 PTZ panoramic camera at each domain for enhancing the interaction between the user and the WebLab.

The layout of the WebLab is shown in Fig. 3. Each domain has a server with the REALabs platform installed. Access to the resources is performed through the public internet or through the KyaTera network.

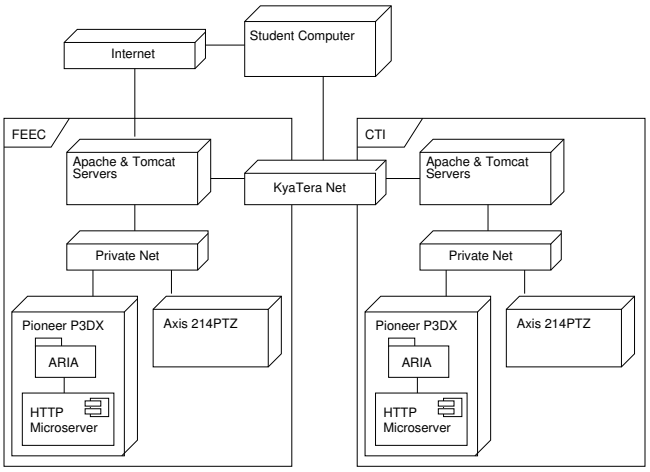

Fig. 3. Layout of the WebLab built above the REALabs platform

The WebLab was employed in an introductory course on mobile robotics. Students code their experiments employing the APIs provided by the platform. Panoramic cameras are accessed directly from the Web browser. Experiments on kinematics; perception; location and mapping; and planning and navigation were proposed according to the adopted textbook [11]. First, the students tune the experiment code on the simulator, then run the experiment of the real robots. Reservation time was set to one hour per accessing session.

Fig. 4 shows an environment map built from sonar readings acquired in real-time from the P3-DX mobile robot and processed on the student's computer. In terms of qualitative evaluation the REALabs platform allowed the building of a secure WebLab simply by grouping resources needed by the experiments, subscribing the students with proper credentials, and opening the experiments for reservation at the appropriate periods of time. Complex issues related to resource protection, concurrent access, and operation across domains and firewalls are handled integrally by the platform. Given the students the choice to develop their mobile robotic experiments using their favorite programming environment contributed to the positive evaluation of the platform.

B. Quantitative Evaluation

In order to evaluate the overheads imposed by the REALabs platform we conducted measurements on sensor readings, image gathering, and robot control. The overheads imposed by the platform can be broken into:

- protocol overheads: caused by HTTP and XML processing at the client and server sides;
- network overheads: caused by network latencies and HTTP proxy operations;

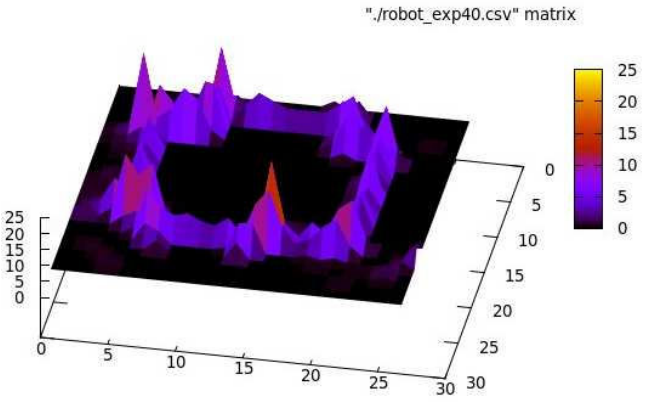

Fig. 4. Experiment on environment mapping

- security overheads: caused by HTTPS data encryption/decryption between the client application and the HTTP proxy agent.

C++ code was written for performing three operations:

1) Robot control (Op1): sets an speed of the mobile robot and returns a very small XML document with the operation result.
2) Sensor readings (Op2): reads the 16 robot's sonars and returns a large XML document with sensor data.
3) Image capturing (Op3): acquires a 320x240 pixels image from the robot's on-board camera. OpenCV was employed for image capturing as ARIA does not support this operation. The operation returns a binary JPEG image.

The series of measurements were conducted in the three scenarios, each one exposing the overheads listed above. One hundred measurements for each operation were taken for each scenario. Mean, standard deviation (SD), and a confidence interval (CI) of 90% were determined. All results are presented in milliseconds.

1) Centralized Scenario: In this scenario the test code runs on the robot's on board processor (Intel Pentium M). The protocol overhead of the REALabs platform is estimated in this scenario by comparing its performance with the centralized version of the ARIA robotic framework. Table I presents the results. Operations 1 and 2 show that protocol overheads in the REALabs platform range from 0.2 to 2 milliseconds according to the size of XML data returned. Operation 3 shows that HTTP processing overheads is negligible when the HTTP payload is large and needs no XML parsing (such as binary images).

TABLE I
RESULT FOR THE CENTRALIZED SCENARIO

	ARIA/OpenCV			REALabs		
	Op1	Op2	Op3	Op1	Op2	Op3
Mean	0.00087	0.01151	34.58	0.191	1.929	34.41
SD	0.00042	0.00074	12.18	0.042	0.054	10.40
CI	0.00007	0.00012	2.00	0.007	0.009	1.72

2) Distributed Scenario: In this scenario it is possible to estimate the network overheads by accessing the robot through the network and HTTP proxy agent. A high speed network, the KyaTera network, was employed in the tests. The network overhead of the REALabs platform is estimated in this scenario by comparing its performance with the centralized scenario above. Table II presents the results. It can be noticed that network and proxy overheads is around 5 to 6 milliseconds for REALabs. Image transferring generate a negligible overhead (the overhead is on image capturing not on image transmission).

TABLE II
RESULT FOR THE DISTRIBUTED AND SECURE SCENARIOS

	Distributed			Secure		
	Op1	Op2	Op3	Op1	Op2	Op3
Mean	4.83	6.23	32.91	35.05	34.76	48.29
SD	1.64	0.44	3.91	4.09	3.37	3.08
CI	0.27	0.07	0.64	0.67	0.56	0.51

3) Secure Scenario: In this scenario it is possible to estimate the security overheads by accessing the robot via HTTPS. Table II presents the results. Overheads around 30 milliseconds are imposed by HTTPS. Again, overhead imposed on image transferring is proportionally lower than those imposed on XML documents.

4) General Comments: Due to the high overhead it imposes, HTTPS must be employed only when security is a strong requirement. An overhead of 10 milliseconds for sensing and acting via HTTP is negligible when a human operator is in the control loop (e.g., in teleoperations). For control schemes requiring fast responses, this overhead may cause instabilities and high error rates. A common approach is to embed certain functions that demand fast control responses and low processing power (e.g., motion control and self-protection functions) and distribute those functions with less stringent time constraints but demanding high processing power (e.g., tasks based on computational intelligence techniques).

We compared the overhead obtained in the distributed scenario with the Microsoft Robotics Studio (MSRS) [12] performing the same sensor readings operation. We set a .NET DSS (Descentralized Software Service) service acquiring the sonar readings using the MSRS facilities and an HTTP GET Service Handling for returning a XML document with the range data. The server side runs on a Dell 510 notebook with Windows XP. The client program was the same used in our tests. We obtained an average of 7.33 milliseconds with standard deviation and confidence interval of 0.73 and 0.14, respectively. The similarity between the REALabs and MSRS protocol overheads indicates that the cost of distribution employing open protocols is limited to a few milliseconds.

VI. RELATED WORKS

The control of remote devices through the internet is not a new topic. The subject of general purpose, application-

independent Web-based platforms is much more recent. Microsoft Robotics Studio (MSRS), introduced in 2006, is a commercial robotic framework that runs above the .NET platform. MSRS models a mobile robot as a set RESTful Web services accessed via SOAP or HTTP carrying XML data. .NET and Windows are always mandatory at the server side and at the client side too in case of interaction via SOAP. This is a strong limitation in terms of programming languages ($C\sharp$ is the preferred development language), performance, and ability to run on small embedded processors. Real-time issues are not addressed by MSRS. A visual programming language composed of simple blocks is provided by MSRS.

REALabs is related with MSRS in terms of adopting Web protocols and the REST interaction pattern. However, REALabs imposes no restrictions in terms of programming languages, supporting platform, and operating system on the client or server sides. MSRS performs access control to the resources locally at the host level while REALabs performs distributed access control at the federation (interdomain) level.

Another recently published Web-based mobile robotic platform is Robopedia [13]. This platform also employs RESTful Web services based on HTTP. Differently of REALabs, data transferred over HTTP is not formatted in XML. Another difference is that Robopedia employs a Web server in between the client application and the mobile robot while REALabs favors small footprint Web servers inside the mobile robots. Robopedia provides no access control mechanisms.

Web-SUN (Small Universal Navigator) [14] is another recently published Web-based robotic platform for educational activities. As Robopedia, Web-SUN employs a Web server for mediating the communication of the client application with the mobile robots. Differently from REALabs that supplies APIs, Web-SUN offer a sophisticated (but specialized) GUI (Graphics User Interface) for performing experimentations with the mobile robots.

ROCI (Remote Object Control Interface) [15] is a component framework with a Web interface for human-robot interactions. As REALabs, ROCI favors XML over HTTP interactions according to the REST interaction pattern. ROCI supports, via scripting, other data formats such as RSS (Really Simple Sindication), binary, and text. RSS is a publish-subscribe protocol for notifying internet users about updated contents. This protocol can be employed for notifying events generated by the robot as well, e.g., energy levels and position updating.

VII. CONCLUSIONS

As mobile robots become part of rich networked environments, most of the tasks executed on the mobile robot's internal processors can be executed on powerful processors outside the robots. This allows small robots to perform complex tasks such as image-based navigation, 3D environmental mapping, and cooperative robotics. For such situations, this paper presented a software platform for network robotics. The platform employs internet protocols such as HTTP/HTTPS, XML, SAML, and XML Security. Such protocols favor the integration of robotic application and the already existing applications distributed across the internet and private networks. By offering APIs in different programming languages and environments, the platform allows many existing robotic algorithms to drive different mobile robots without the need to be adapted to different robotic frameworks.

The platform was evaluated in a real world application, a WebLab accessed by fifteen students during three months. Tests were conducted in order to assess the platform overheads. The overheads due to distribution, HTTP/XML protocols, and security were estimated.

We are extending the REALabs platform in subjects related to interdomain operations. Models for policy-based authorization, Service Level Agreements (SLA), and Quality of Service (QoS) are being evaluated and expected to be incorporated in the platform soon.

VIII. ACKNOWLEDGMENTS

This research was supported by Fapesp (grant 2006/06005-0).

REFERENCES

[1] MobileRobots, Inc. ARIA Wiki, `http://robots.mobilerobots.com/wiki/ARIA`, March 2010.

[2] Player Project Web Site, `http://playerstage.sourceforge.net/`, March 2010.

[3] Orca Web Site, `http://orca-robotics.sourceforge.net/`, March 2010.

[4] L. Rchardson and S. Ruby, *RESTful Web Services*, O'Reilly, 2007.

[5] RTAI (RealTime Application Interface for Linux) Web Site, `http://www.rtai.org`, March 2010.

[6] Internet2 Middleware Initiative, Shibboleth Web Site, `http://shibboleth.internet2.edu/`, March 2010.

[7] OpenSSO Web Site, `https:/opensso.dev.java.net/`, March 2010.

[8] S. Cantor, J. Kemp, R. Philpott and E. Maler, Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0, OASIS Standard, March 2005. Available at `http://www.oasis-open.org`.p

[9] E. Cardozo, E. Guimarães, F. Paolieri and V. Pinto, *REALabs-BOT: a WebLab in Mobile Robotics Over High Speed Networks*, 1st IFAC Workshop on Networked Robotics, Golden, USA, October 2009.

[10] D. Moraes, P. Coelho, E. Cardozo, E. Guimarães, T. Johnson, F. Atizani, *A Network Architecture for Large Mobile Robotics Environments*, Second International Conference on Robot Communication and Coordination (Robocomm), Odense, Denmark, April 2009.

[11] R. Siegwart and I. Nourbakhsh, *Introduction to Autonomous Mobile Robots*, The MIT Press, 2004.

[12] J. Jackson, Microsoft Robotics Studio: A Technical Introduction, *IEEE Robotics and Automation Magazine*, vol. 14, no. 4, 2007, pp 82-87.

[13] R. Edwards, L. Parker and D. Resseguie, Robopedia: Leveraging Sensorpedia for Web-Enabled Robot Control, *7th Intl. PerCom Workshop on Managing Ubiquitous Communication and Services*, Mannhein, Germany, April 2009.

[14] S. Sagiroglu and N. Yilmaz, *Web-based Mobile Robot Platform for Real-time Exercices*, Expert Systems with Applications, vol. 36, 2009, pp 3153-3168.

[15] A. Cowley, H. Hsu, C. Taylor, *Opening the Dialog: Robotics ad the Internet*, International Conference on Robotics and Automation (ICRA 2006), Orlando, USA, May 2006.