HORUS: an Agent System for Home Automation

Attilio Giordana
Computer Science Department
Universita‘ del Piemonte Orientale
Via Teresea Michel 11, 15121 - Alessandria
Email: attilio@mfn.unipmn.it

Dino Mendola, Andrea Moio, and Davide Monfrecola
Penta Dynamic Solutions
Via Teresa Michel 11, 15121 Alessandria, Italy
Email: dino.mendola@pec.it

***Abstract*—This paper presents a system for home automation, called HORUS, which is based on an agent architecture. The benefits deriving from this type of solution are essentially three. Firstly, the agent paradigm provides a good abstraction mechanism for implementing a modular system, easy to configure according to the requirements of a specific application. The second reason is the possibility of scaling up to complex applications exploiting parallel and distributed computing on low power micro-pc. The third reason is load balancing and fault tolerance. When required, the agents migrate from one host to another providing fault tolerance and load balancing capabilities. Some agents are provided with learning capabilities and can learn the model on the environment where they operate. Finally, the agents communicate in xml over https and are integrated in a web environment which offers an ubiquitous access from any kind of portable device, while providing a secure access. HORUS is now a commercial product, which provides either a sophisticated anti-intrusion system, and energy control.**

I. Introduction

This paper describes a distributed system designed for automation named HORUS[1]. The characteristic of HORUS, which distinguishes it from other commercial systems designed for the same task is the agent based architecture, which is in the line of [1], [2]. This choice is due to several different reasons. The first one is modularity. Every agent implements a specific function, and is a self-consistent building block. Depending on the user needs a site dependent network of agents is defined in order to implement the required functions. Several instances of every agent type may be active at the same time. The second reason is parallel and distributed computing. The hardware suitable for home automation is usually based on low power cpus, which may be installed in a wall-box, but do not provide enough power for complex applications. Exploiting parallel and distributed computing is a solution that allows computing intensive applications be implemented without scaling up to more expensive hardware. A third reason is fault tolerance [3]. Several home applications, for instance anti-intrusion, are critical with respect to fault tolerance. A failure in the system may endanger things and human beings. Agents may be redundant and migrate from one computational node to another in order to guarantee the most critical functionalities. Finally, a last reason is load balancing. Migrating an agent is the typical way of balancing the load on a network of hosts.

[1]Horus is a trademark of Penta Dynamic Solutions

Another important feature of Horus is the user interface, which is based on a protocol based on XML/HTTP and provides an ubiquitous access from Internet and from any kind of fixed of portable device. Moreover, Horus is assisted by a WEB based configuration module, which supports a technician in order to design and configure a custom installation.

Finally, an indirect benefit deriving from the modularity inherent to the agent based architecture, is the possibility of experimenting innovative methods, like adaptive algorithms based on a machine learning approach, without compromising the critical functionalities of the system. In fact, some of this features are now included in the official release (Horus-3.0), which is a commercial product, and counts several installations in domestic end business environments.

II. Horus Architecture

HORUS is implemented as an overlay network of agents distributed on a TCP/IP network of hosts providing a UNIX compatible platform like Linux or MacOS. More specifically, hosts may belong to two categories:

- General purpose mini/micro-pc, offering a full UNIX environment;
- Custom devices like IP-videocameras and IO-boards, which provides http based interface and are accessible as web sites.

The agents are allocated on the general purpose hosts and communicate over http using an xml based protocol. Custom hosts, are then included in the agent community as *special* agents. The agent communication protocol is extensible in order to include the characteristic idioms of the custom hosts. An example of agent network including IO-boards and IP-videocameras is provided in Figure 1.

A. The Platform

IP-videocameras are becoming quite popular because of the possibility of being directly accessible from Internet. In our case, this solution is appealing because avoids the need of dedicating a host to serve a videocamera. IP-videocameras typically offer one, or more, streaming service, accessible at specific url. Very frequently they implement some elementary motion detection algorithm, which uploads a videoclip on a server, any time severe changes in the recorded image are observed.

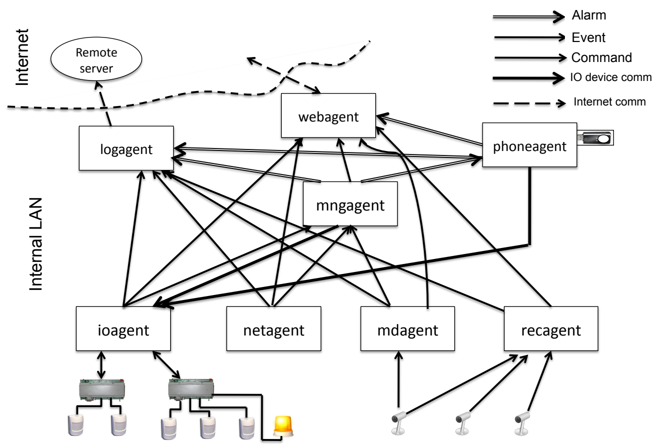


Fig. 1. Example of agent network.

IO-boards provide the interface toward sensors and actuators via a set of digital or analogical channels. Many IO-boards can be installed in a box of the electric plant, and provide an ethernet interface accessible via cable UTP or Wi-Fi. In general they implement the stack TCP/IP and can be polled or commanded via messages encoded in HTML/XML. Nevertheless the specific message format is strongly dependent upon the board type.

B. Agent Types

Agents are subdivided into different types characterized by the function they implement. Agents of the same type in most cases share the same code, end differentiate the behavior only in dependence of the configuration parameters. All agents use a same communication language encoded in XML over http. Three kinds of messages are defined: *Events*, *Actions*, and

Alarms. The former ones simply communicate changes in sensor or agent status. Actions are commands that force the destination agent to change its status or to activate some actuator. Alarms correspond to urgent and potentially dangerous events, and are processed with priority with respect to the others.

Agent interaction is event driven and has been designed in order to guarantee strict real time requirements, as it may be necessary in a security application. More specifically, the agent interaction is always atomic. When an agent sends a message it receives an immediate response, without waiting for the accomplishing of a remote procedure. In this way, delays and deadlocks due to chains of requests waiting for a remote answer are impossible.

Nevertheless, this choice restricts the possible information content in an answer to the knowledge currently available from the agent status. No actions are possible in order to increase the local knowledge before answering.

In the following we will briefly review the different agent types.

1) Managers: They are the agents responsible for taking decisions reacting to events received from other agents. (IO-handlers or MD-agents). They are provided with a knowledge base encoded in form of production rules. More precisely a rule has the following general form:

$$\phi \rightarrow actions, alarms \tag{1}$$

where the precondition ϕ is logic clause on some of the variables characterizing the agent status, $actions$ is a list of actions to execute, and $alarms$ is a list of alarm event to communicate to other agents. Either alarm or action list may be empty. Moreover, the action list may include internal actions entailing the immediate change of the manager internal status.

Managers are typically used for controlling subsystems like the anti-intrusion and the air conditioning. Depending on the rule base and the events it receive a manager may instantiate one or the other functionality.

In addition to the rule base, managers are provided with an activity program, which defines the activity phases. When a manager is active it behaves according to the rule base. When it is not active, only registers the received events but does not send any alarm, nor execute any action.

2) IO-handlers: Sensors and actuators are interfaced through IO-boards. Nevertheless, IO-boards communicate using device specific languages, which does not complain with HORUS standard. In order to integrate them with the HORUS agent community, an IO-handler acts as a wrapper, which provides a standard interface toward the different IO-board types. Any time a change in an input channel of the IO-board is detected, an event message is sent to the agents interested in the specific event. Any time, an action message is received from another agent, a proper sequence of commands is sent to the IO-board in order to activate an output channel.

3) Video-camera handlers: Two basic functions can be implemented using this type of agent:

- Continuous recording according to a specified time scheduling (REC-agent).
- Motion detection (MD-agent).

The alternative behavior is specified by the parameters into the configuration files the agent reads at the start up time.

Recording activity simply consists in creating an archive of videos, which can be consulted when the user want to trace same event happened in the past. Motion detection, is a more critical task, which consists in discovering critical events in the scenario observed by a camera. The first step is accomplished by the camera itself, which upload a video-clip, in a directory monitored by an MD-agent, any time some major change is observed in the scenario. The MD-agent analyzes the video-clip using a sophisticated vision algorithm and, if it decides that some critical event actually happened, sends an event message to the manager in charge of security.

4) Alarm communicators: An alarm communicator (PHONE-agent) is an agent provided with an output device such as a GSM card, or an IP-phone, which is used to send alarm messages to a user, or to a security service. Several kinds of messages can be used ranging from sms to vocal messages, depending on the user preference. A PHONE-agent is usually provided with a table that specifies, for every alarm type, which user must be alerted.

5) Self monitoring subsystem: A key activity of the agent system is continuous self-monitoring in order to immediately detect critical failures or tampering attempts, which could prevent it from reacting to critical events. Two kind of agents are in charge of this activity:

- NET-agents
- TAMPER-agents

A NET-agent is very simple. It periodically checks the reachability of an assigned set hosts. If one of them is not reachable for more than a few seconds it sends an event message to a manager. In this way, hardware failures (either natural or caused by an intruder) of the network components are immediately detected and communicated to an administrator.

Tampering is a kind of attack aimed at compromising the sensorial apparatus of a surveillance system. Typical targets are the infrared detectors and the videocameras. Infrared detectors are usually provided with an anti-tamper mechanism, which is interfaced to HORUS using some input channel of an IO-board. Videocameras are more fragile because many attacks can be done without a physical contact. TAMPER-agents periodically check the videocameras functionality by verifying that the recorded image corresponds to the expectation. Some more details are provided in Section V.

6) Loggers: Goal of the loggers is to provide a complete logging of events and alarms to be used for diagnosis. The type of agent in charge of this activity is the LOG-agent. It receives alarm and event messages from the other agents and periodically upload them on a remote server. However, logging is also accomplished by the WEB-agent, in order to provide the user with a complete information source. The WEB-agent is nothing else than a classical web server, which is integrated in the agent community in order to provide a flexible interface to the user.

C. Agent Architecture

All agents have a similar architecture organized as an acyclic forward graph of threads communicating by means of message queues. The reason for choosing this architecture is to avoid possible dead-locks or delays on critical sections. Examples of agents are reported in Figure 2 and 3, corresponding to an MD-agent and to a manager, respectively. More specifically the WEB-Thread provides the http interface to the incoming message from other agents, and is able to immediately answer without delays. The Trigger thread, accounts for the time flow and schedules internal actions at prefixed intervals. The Core thread is agent specific and implements the real work carried by the agent. Finally, the Destination and the Action threads are in charge of forwarding event, alarm, and action messages, to a list of other agents.

The Archive thread, which is presents only in some agents like MD-agents and REC-agents, is in charge of storing in the archive the videos acquired from the cameras. Finally, the Filter thread, visible in the MD-agent architecture, is the one executing the video analysis checking for events corresponding to items moving in the scenario. Finally, the Log thread (to not be confused with the LOG-agent) provides to the other threads a logging service, on a file, for debugging purposes.

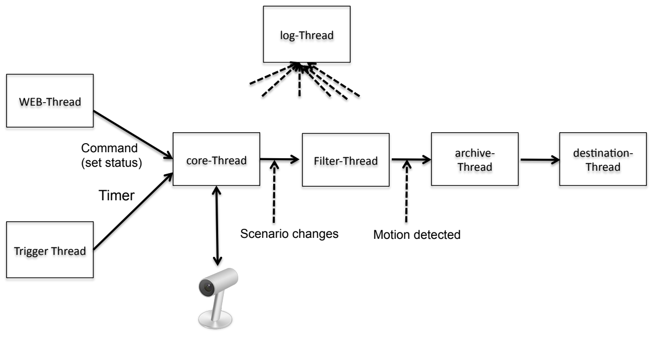


Fig. 2. Every agent is organized as a set of threads communicating through message queues. As an example, the figure describes an MD-agent

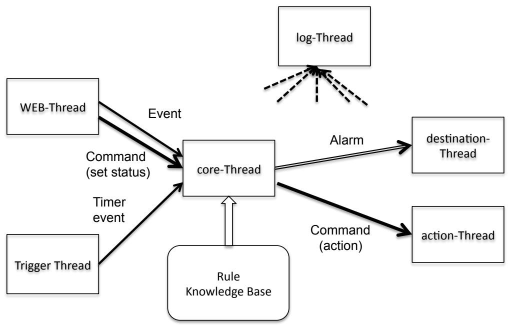


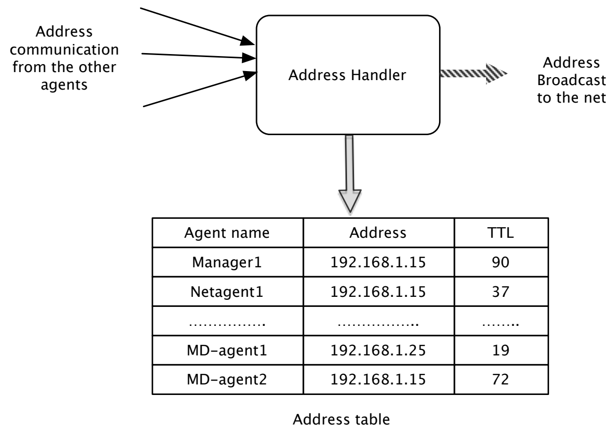
Fig. 3. The manager agents are provided with a rule knowledge base.

D. Agent Addressing

Agents communicate via a standard http/tcp protocol. A critical point is how an agent may find the IP address of the others agents. In order to support migration and fault-recovery, the agent addressing mechanism must be dynamically reconfigurable. Nevertheless, the classical method of using a dynamic DNS-server is fragile, because the failure of the server would compromise the entire system functionality. Then a different solution has been implemented.

Every agent is provided with a thread (not described in Figure 2 and 3), which runs in background and maintains a local addressing map of the active agents in the local network. Periodically, an agent announces, in broadcast over UDP, its name and its IP address to all agents listening on the internal network. When an agent receives the address of another agent, it updates the corresponding entry (see Figure 4) and the Time To Live (TTL) value is set to the maximum.

The TTL of all entries is decremented according to the cpu clock. If it expires before receiving a new announcement from


Address communication from the other agents
Address Handler
Address Broadcast to the net
Address table

Agent name	Address	TTL
Manager1	192.168.1.15	90
Netagent1	192.168.1.15	37
……………	……………..	……..
MD-agent1	192.168.1.25	19
MD-agent2	192.168.1.15	72

Fig. 4. Address manager thread

the corresponding agent, this one is considered inactive. In order to prevent the possibility for false messages poisoning the address tables, all messages communicating IP address are authenticated using a standard signature protocol based on RSA.

III. Web Interface and Ubiquitous Access

The interface to Horus is provided by the WEB-agent, which is a traditional web server. In the standard configuration is located in the same local network where the agent system is installed. Nevertheless, if required, it can be located in any place in Internet (e.g., in a cloud).

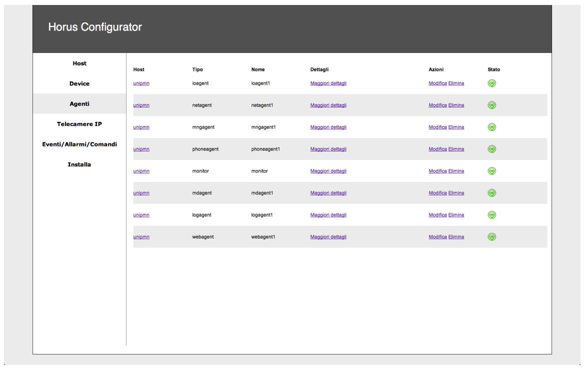
Two kinds of interface are provides by the web server. One is designed for assisting a technician accomplishing an HORUS installation. The other offers an ubiquitous access to HORUS from any kind of portable or fixed device provided with a web browser.

A. Horus Configurator

An example of HORUS configuration interface is provided in Figure 5. The configuration module has been designed in order to assist a technician during the steps necessary to customize the agent system for a specific installation. No specific knowledge in programming is assumed, but only a technical background concerning the installed components: sensors, actuators and videocameras.

B. Horus Access

The user interaction with the Agent system is provided by a set of dynamic pages, which automatically adapt to the browser and to the specific circumstances. The web pages are constructed considering the size of the display of the device the user is connecting from and possibly the urgency. In case of emergency, the first item on the display is a synthetic description of what happened. Example of the pages generated for an i-phone end for a tablet are provided in Figure 6 and 7. In order to guarantee secure connections from every Internet access point, the connection is over https and requires a certificate pre-installed on the access device.



Host	Tipo	Nome	Dettagli	Azioni	Stato
unipmn	ioagent	ioagent1	Maggiori dettagli	Modifica Elimina	✓
unipmn	netagent	netagent1	Maggiori dettagli	Modifica Elimina	✓
unipmn	mngagent	mngagent1	Maggiori dettagli	Modifica Elimina	✓
unipmn	phoneagent	phoneagent1	Maggiori dettagli	Modifica Elimina	✓
unipmn	monitor	monitor	Maggiori dettagli	Modifica Elimina	✓
unipmn	mdagent	mdagent1	Maggiori dettagli	Modifica Elimina	✓
unipmn	logagent	logagent1	Maggiori dettagli	Modifica Elimina	✓
unipmn	webagent	webagent1	Maggiori dettagli	Modifica Elimina	✓

Fig. 5. Example of Horus Configurator interface

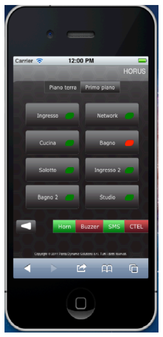


Fig. 6. Horus interface for a 3G phone

IV. Dynamic Reconfiguration

Agent allocation to hosts may be dynamically changed in order to handle fault-tolerance and load balancing. In fact, it is very simple to migrate the activity of an agent from a host to another. When dynamic reconfiguration is required

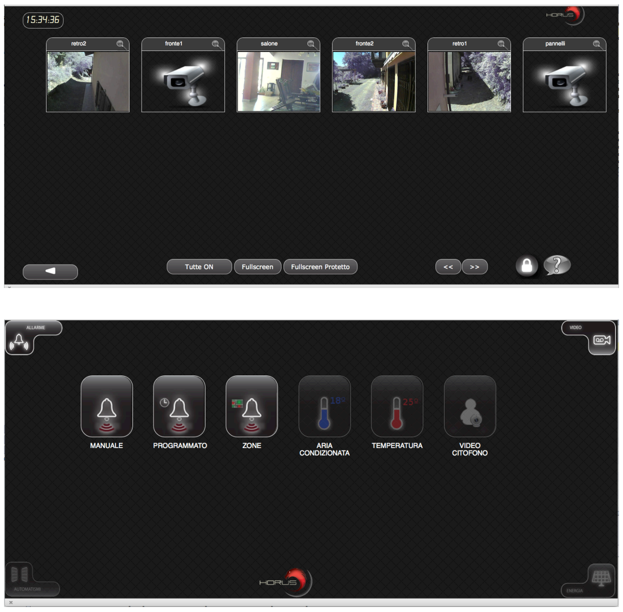


Fig. 7. Examples of Horus interface for a netbook

the code of all agent types is always installed on the hosts, which are potentially involved in the reconfiguration process. Agents have a soft status, which can be reconstructed in a short time by interacting with the other agents. Then, migrating an agent does not requires to migrate the agent memory, but simply requires to start a new copy of the agent in a host while the old copy stops. The mechanism, which handles the addressing table, automatically notifies the new address to the other agents.

In the current HORUS release, fault-tolerance and load balancing are handled using a very simple mechanism. Critical agents, like managers may be duplicated on different hosts. However, the two copies have assigned a different behavior. One copy works as *master*, while the second copy works as *backup* in stand by. When a backup agent does not receive the address notification from the master copy, so that the TTL in the address table expires, automatically switches to a *master* behavior. When the original master begins again to notify its address, the backup agent switches back to its original role. This mechanism implements fault-tolerance. Load balancing can be obtained by commanding the two agent copies to switch their role. Currently, this mechanism is controlled manually through a page of the configuration module. Nevertheless, the next release should provide a policy for automatically handling load balancing.

V. Adaptive Capabilities

Some agents are provided with learning capabilities in order to adapt automatically to the environment where HORUS is installed. More specifically they are the agents dedicated to anti tampering of the video-surveillance sub-system and to the motion detection.

Preventing tampering of videocameras is critical and important problem, which has been investigated by many authors, as reviewed in [4]. Nevertheless, no completely satisfactory solutions, capable of dealing any environmental condition have been found in the literature [4], [5], in spite of the claims made. For this reason, a new algorithm has been implemented[6]. Differently from others approaches to the same problem[4], [5], which focuses on the detection of abrupt changes occurring owing to a tamper, the anti-tamper algorithm we propose is based on a model of the correct behavior of the visual apparatus. The model accounts for a set of items, which must be detectable under different light conditions during the day and the night. When these item are not detectable for a given period of time a security warning is sent to the house administrator. The model is automatically learned by observing the environment, without intervention by side of the user. The initial training period last about three days. Then the learning process continues as a long term background activity, in order to account for the environmental changes due to the different condition of light in the different seasons, while the agent is already operational.

Another agent type, which is provided of adaptive capabilities is the one devoted to motion detection. As described in Figure 2 this kind of agent is provided with a filter, which analyzes the video-clips, generated by the camera when the motion detection is triggered, in order to distinguish moving items corresponding to intrusions from false alarms. The filter thread is provided with a short term model of the world (different from the long term one used by the anti tamper agent), which is continuously updated using an adaptive algorithm. In this way the agent is able to detect major changes in the scenario, which activate a deeper analysis of the image aimed at detecting blobs corresponding to moving objects. Blobs are then classified into *alarms* or *non-alarms*. The classifier is an SVM [7] trained from a set of positive and negative examples.

Finally, a new agent, not yet included in HORUS, is now under development. Its aim is to provide a behavioral model of devices like appliances, which will be exploited in the Power management sub-system. This kind of agent is based on a Dynamic Bayesian network [8]

VI. Conclusion

In this paper, we described commercial system based on an agent architecture, which is now operational in houses and shops. Commercial systems are necessarily based on consolidated methods, which make them less advanced than one could expect, considering the state of the art emerging from the literature. Nevertheless, HORUS, due to the critical tasks it needs to face, incorporates a number of advanced features, which have been a key aspect for its success. More specifically, the agent based architecture has been fundamental to implement modularity, scalability and fault-tolerance. Moreover, features inherently originating from the agent system literature, such adaptability, have been included in the most recent HORUS release.

References

[1] C.-L. Wu, C.-F. Liao, and L.-C. Fu, "Service-oriented smart-home architecture based on osgi and mobile-agent technology," *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, vol. 37, no. 2, pp. 193–205, 2007.

[2] W. Shen, Q. Hao, S. Wang, Y. Li, and H. Ghenniwa, "An agent-based service-oriented integration architecture for collaborative intelligent manufacturing," *Robot. Comput.-Integr. Manuf.*, vol. 23, no. 3, pp. 315–325, Jun. 2007. [Online]. Available: http://dx.doi.org/10.1016/j.rcim.2006.02.009

[3] S. Kumar, "The adaptive agent architecture: Achieving fault-tolerance using persistent broker teams," in *In Proceedings of the Fourth International Conference on Multi-Agent Systems*. IEEE Computer Society, 2000, pp. 159–166.

[4] A. Saglam, "Adaptive camera tamper detection for video surveillance," June 2009.

[5] E. Ribnick, S. Atev, O. Masoud, N. Papanikolopoulos, and R. Voyles, "Real-time detection of camera tampering," in *Proceedings of the IEEE International Conference on Video and Signal Based Surveillance*, ser. AVSS '06. Washington, DC, USA: IEEE Computer Society, 2006, pp. 10–. [Online]. Available: http://dx.doi.org/10.1109/AVSS.2006.94

[6] A. Moio, "An anti-tampering algorithm bbased on an ai approach," 2012, technical report TR-02-12.

[7] J. Shawe-Taylor and N. Cristianini, *Support Vector Machines and other kernel-based learning methods*. Cambridge University Press, 2000.

[8] V. Mihajlovic and M. Petkovic, "Dynamic bayesian networks: A state of the art," Enschede, 2001, dMW-project. [Online]. Available: http://doc.utwente.nl/36632/