Reliable and reconfigurable Wi-Fi monitoring network based on software agents

D. Bordencea, H. Vălean, S. Folea, A. Dobîrcău

***Abstract* — A reconfiguration system based on software agents, is presented. The agents use the Paxos protocol in order to allocate the hardware resources via Wi-Fi. If an Access Point fails, the agents negotiate to reallocate its clients to that Access Point with a smaller number of clients. In this way the system implements virtual redundancy, a low cost reliable configuration: if a part of the system fails, other parts will assume its tasks.**

***Keywords* — software agents, consensus, PAXOS, virtual redundancy.**

I. Introduction

So far, researchers, have not been able to agree upon a common definition for software agents, defining them according to their own point of view. Russell and Norvig define an agent as performing two tasks: it senses its surrounding environment through sensors and performs actions in it with the help of its effectors [1]. Otherwise, a software agent was described as “an umbrella term for a heterogeneous body of research and development” [2]. "Autonomous agents are computational systems that inhabit some complex dynamic environment; sense and act autonomously in this environment and by doing so realize set of goals or task for which they are designed" [3] was another definition gave by Maes and Pattie. One of the most used definitions is the definition of Jennings and Wooldridge, that says: “An agent is a hardware or software-based computer system with the following properties: autonomy, social ability, reactivity and pro-activeness.”

Thereby, a program can be called agent because it can be scheduled in advance to perform tasks on a remote machine; another because it accomplish low-level computing tasks while being instructed in a higher-level of programming language or script (Apple Computer 1993); another because it abstract out or encapsulate the details of differences between information sources or computing services (Knoblock and Ambite 1997); another because it implement a primitive or aggregate “cognitive function” (Minsky 1986, Minsky and Riecken 1994); another because it shows characteristics of distributed intelligence (Moulin and Chaib-draa 1996); another because it serve a mediating role among people and programs (Coutaz 1990; Wiederhold 1989; Wiederhold 1992); another because it perform the role of an “intelligent assistant” (Boy 1991, Maes 1997); another because it can migrate in a self-directed way from one computer to another (White 1996); another because it present themselves to users as believable characters (Ball et al. 1996, Bates 1994, Hayes-Roth, Brownston and Gent 1995); another because it knows and speak an agent communication language (Genesereth 1997, Finin et al. 1997) and another because it is viewed by users as manifesting intentionality and other aspects of “mental state” (Shoham 1997) [4].

Sometime, for solving a problem, some competences are needed and a single software agent is unable to address it. In this case, several agents form a distributed network and work together to solve the problem. This management is called Multi-agent System, a system composed of multiple interacting software agents. It can be used for problems distributed in nature which are difficult or impossible for a single agent to solve. In multi-agent system task is decomposed into several subtasks distributed among various agents that interact with each other. Because different agents can have different capabilities, the system must be built so that the agents may cooperate, coordinate or negotiate to achieve the task [5].

Industry is a very important field of application for multi-agent systems because they are where the first multi-agent system techniques were experimented with and demonstrated their initial potential. Examples of industrial domains where multi-agent systems have been usefully employed include process control, system diagnostics, manufacturing, testing, transportation logistics, and network management and distributed monitoring [6]. In particular, the Internet has been shown as an ideal domain for multi-agent systems due to its intrinsically distributed nature and the sheer volume of information available. Agents can be used, for example, for searching and filtering this mass of information [7]. The Internet has also pushed the use of agent technologies in the commerce and business process management fields. In fact, before the spread of Internet commerce, business process management was almost entirely driven by human interactions: humans deciding when to buy goods, how much they are willing to pay, and so on. Now electronic commerce and automated business processes are increasingly assuming a pivotal role in many organizations

D. Bordencea is PhD student at the Faculty of Automation and Computer Science, Technical University of Cluj-Napoca, Cluj, Romania (phone: 0040751036923 e-mail: daniela.bordencea@aut.utcluj.ro).

H. Vălean is Professor at the Faculty of Automation and Computer Science, Technical University of Cluj-Napoca, Cluj, Romania (e-mail: honoriu.valean@aut.utcluj.ro).

S. Folea is Associate Professor at the Faculty of Automation and Computer Science, Technical University of Cluj-Napoca Cluj, Romania (e-mail: silviu.folea@aut.utcluj.ro).

A. Dobîrcău is PhD student at the Faculty of Automation and Computer Science, Technical University of Cluj-Napoca, Cluj, Romania (e-mail: ancuta.dobircau@aut.utcluj.ro).

because they offer opportunities to significantly improve the way in which the different entities involved in the business process interact.

Within the constraints of the system's communication protocol, agents can share knowledge using any language for agreement (for example Knowledge Query Manipulation Language-KQML or FIPA's Agent Communication Language-ACL).

II. SYSTEM ARCHITECTURE

Figure 1 present the system architecture. The clients connect to the Access Point and transfer information to a PC, via Wi-Fi. The PC contains a multi-agent society for receive client's information and manage the connection between clients and Access Points.

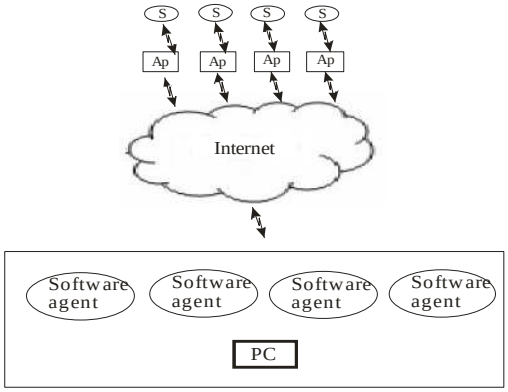

Fig. 1. System architecture

A. Software architecture

The agents are used for read client's information and manage number of Access Point client's connection. When a new client wants to connect to Access Point, the agent associated with Access Point will start a negotiation with the other agents. Thus, it sends a message that contains the number of Access Point client connection to each agent available. The agents that receive this kind of message compare the number with its own number of client connection. If its own number is bigger than the number received, then the agents will not do anything. If it is smaller, it will send a message back to the initiator with its number of connections. After receiving the message from all agents, the initiator agent compares the numbers and extracts the smaller one. If more Access Point with the same number of clients exists, the agent that initiates the message can choose between them randomly or it can use again consensus to agree with allocation the client to only one Access Point. Consensus decides when a majority of agents are agreeing on the same Access Point. Then, the initiator agent sends a message to the agent with the smaller number and it proposes to take on it the client. When an Access Points fails its clients will choose an Access Point to connect with them.

An example of failure is presented in figure 2:

- **Step 1**: AP_1 fails;
- **Step 2:** AP_1 became un-available, so its client associated with Ap_1 will re-scan the network, and will decide to associate with Ap_2; configuration agent_1 will take the status un-available.
- **Step 3:** Configuration agent_2 will start the negotiation process, sending to the configuration agent_3, the number of connection = 1;
- **Step 4:** Configuration agent_3 will extract the number 1 from the message and will compare with the AP_3 number of connection which is 1.
- **Step 5:** Configuration agent_3 will send to configuration agent_2, the number =1;
- **Step 6:** Configuration agent_2 compares its number of clients with the number received and it will decide to take two clients of the Ap_1 clients;
- **Step 7:** Configuration agent_2 will send a message to two of Ap_1 clients with the AP_2 connection information;
- **Step 8:** This clients will start a new association with AP_2;
- **Step 9**: Configuration agent_2 will ask configuration agent_3 to take the last client connection from the AP_1;
- **Step 10:** Configuration agent_3 will send to configuration agent_2 the Ap_3 connection information;
- **Step 11:** Configuration agent_2 will send a message to the last Ap_1 client with the AP_3 connection information;
- **Step 12:** This client will start a new association with AP_3.

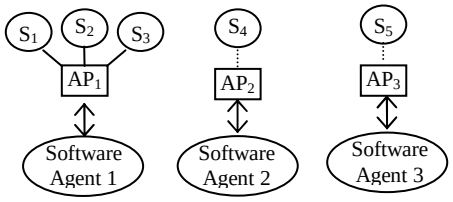

Fig. 2. Initial Step

The agents use consensus in order to avoid possible allocation conflicts (the negotiate client allocation). For each join or fail of an Access Point, consensus it will be used.

If an Access Point will fail, his configuration agent_1 will take the status un-available. The rest of configuration agents will figure out about the failure and will run Paxos in order to reallocate the clients of the failed Access Point. They will choose the agent with the highest number of available connections as a leader. If the number of failed Access Point clients is smaller than the number of the leader available connection, these clients will start an association with the leader. Otherwise, the leader will run consensus again for the rest of the clients. It will send a message to the agents for asking about their available connections. The configuration agents that receive the message will reply to the leader with their number of connections. The leader will allocate the clients to the rest of the Access Points. If the clients can start a new association with the Access Points they were distributed, "the job is well done". Otherwise, the Paxos algorithm will run again, in a new instance, when a new leader is chooses and the process will repeat.

Consensus is an algorithm for agreement in the presence of failures. It uses a set of processes, which can be called agents. The agents use the consensus abstraction to agree on a common value out of values they initially propose.

Consensus uses two primitives: propose and decide. Each agent has an initial value that it proposes for agreement, through the primitive propose. In abortable consensus the value returned can also be a specific indication ⊥ which means that consensus has aborted, is not necessarily to be a value that was proposed by some process. This is possible when another process tries to concurrently propose a value.

A consensus algorithm ensures that a single value that was proposed is chosen; in this case a client is allocated to a single Access Point. If no client wants to connect to an Access Point, then no client should be allocated. If a client is allocated to an Access Point, then all agents should be able to learn this.

The safety requirements for consensus are [9]:

- Only a value that has been proposed may be chosen.
- Only a single value is chosen.
- A process never learns that a value has been chosen unless it actually has been.

However, the purpose is to ensure that a value that was proposed is eventually chosen and, some process can eventually learn the chosen value.

Because a client may choose to authenticate to multiple Access Points simultaneously, consensus ensures that a client is connected to only one Access Point.

The Paxos algorithm is both an elegant solution for the consensus problem and a mechanism for the delivery of totally ordered messages that can be used to support active replication. Through some communication protocol, all of them should come to an agreement and unambiguously choose a single value among the initial ones. The protocol is fully decentralized; it operates under message failures and requires only a majority of the processes to be alive to make progress [11].

In Paxos, an agent can play tree roles:

• A proposer.
• An acceptor.
• A learner.

The Paxos algorithm can operate in two phases. In first phase, a proposer selects a proposal number *n* and sends to a majority of acceptors a prepare request. When an acceptor receives a prepare request with number *n* greater than the last prepare request that it has responded already, it will send a promise to not accept a proposal number less than *n*. In second phase, if a majority of acceptors replied to the proposer, then it will send an accept request to each of this acceptors for the proposal numbered *n* and value *v*, where *v* is the value of the highest-numbered proposal among the responses, or is any value. When an acceptor receives an accept request for a proposal numbered n, it will accept it only if it has not already responded to a prepare request having a number greater than *n* [9].

To learn that a value has been chosen, a learner must know that a proposal has been accepted by a majority of acceptors.

For solving consensus, in Paxos the agents execute multiple rounds. Each round has a unique identifier and a leader. Proposers send their proposed value to the leader that tries to reach consensus on it in a round. Thus, the leader is responsible for that round, being able to decide if any other rounds were successful or not.

B. Hardware architecture

When clients wish to connect to a wireless network, they first scan for available networks. When an available network receives a probe request, they will transmit a probe response packet containing the necessary information required to use the network including channel information. The client then decides which Access Point is the best choice for their connection. After the client has decided which Access Point it wants to connect to, it will transmit a management frame with an authentication sub-type, requesting authentication to the network. In the case of open networks, the Access Point simply responds with an authentication success message and the connection process continues. In the case of a network utilizing WEP, the Access Point will attempt to validate the WEP key of the client that wishes to access the network by sending them an authentication challenge. Once the client gets an authentication success message from the access point, the authentication process is complete. A client may choose to authenticate to multiple Access Points simultaneously. When the client is authenticated, then it will start the association process. A client can associate to only a single Access Point at any given time. For the association process a client will send a management frame with the associate request sub-type to the Access Point. The Access Point will check to make sure the client has already authenticated. If an entry exists indicating the client has already authenticated to the Access Point, it will generate an associate response message to the client. Once authenticated and associated to the Access Point the client needs to acquire an IP address. The client use the DHCP protocol to request a dynamic allocation from a DHCP server, implemented on the Access Point. The client is configured to use Power Save Poll mode. This allows the client to sleep while waiting the DHCP response. Once an IP address is acquired, the client will be able to communicate with the Access Point, using the UDP protocol.

If an Access Point becomes unavailable, the clients associated with the unavailable Access Point will scan again the network and will re-start the authentication and association process with a new available Access Point.

An advantage of this architecture is that Wi-Fi communication is widely spread in buildings or even in city squares and the existent infra-structure can be used without laborious software effort and supplementary costs.

III. EXPERIMENTS

Web Instruments are true virtual instruments because the physical part of the instrument may be located very far away from the instrument panel. The web instrument leverages the network as infrastructure technology.

A Web Instrument is very configurable; each that is displayed can be individually configured. The configuration window is presented in figure 3.

Figures 4 present Paxos Uniform Consensus for four agents. For this scenario, agent 1 is the leader, so only him can propose values. If agent 1 will fail, agent 2 will take its tasks. After all processes accepted write phase is printed decision for the instances. In an instance, only a single can be accepted; for accepting multiples values in Paxos are

used instances. This implementation it was made in Kompics. Kompics [12] is a model for building reconfigurable distributed systems from event-driven components. It has been developed in Sweden at the Royal Institute of Technology (KTH) and the Swedish Institute of Computer Science (SICS).

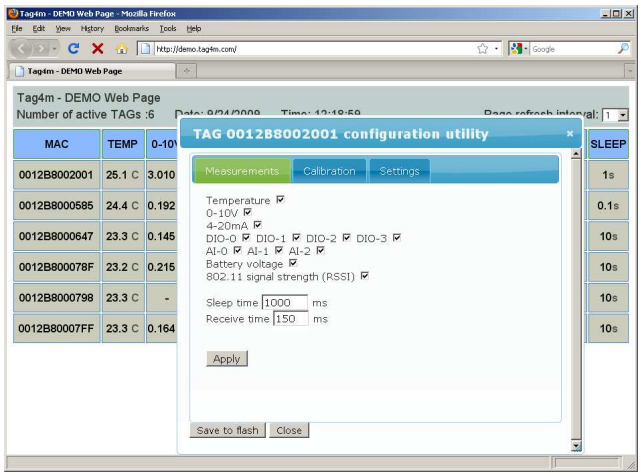

Fig. 3. Configuration Panel presented by the Web Instrument Example

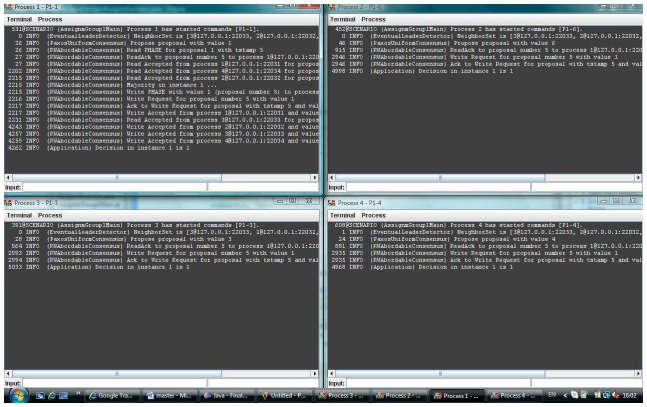

Fig. 4. Paxos for a topology with 4 processes, 1 instance

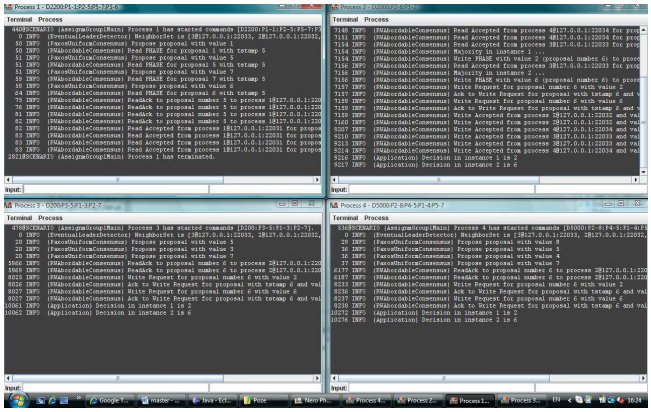

Fig.5. Paxos for a topology with 4 processes, multiple instances, and failure of leader before decided

Figure 5 present a topology with for agents. This scenario is an experiment when the leader fails. As it can be seen, after failure second agent take the task and it will propose in the future.

This experiment shows that Paxos is fault-tolerant because when the leader fails, the agents choose another leader which can start to propose values. In Paxos if an acceptor fails and after that it will recover, it have to remember the highest number proposal that it has ever accepted and the number of the highest - numbered prepare request to which is responded.

IV. Conclusion

In the paper, a distributed reconfigurable monitoring system is presented. The communication between the parts of the system is Wi-Fi. The allocation of the clients to the routers is dynamic, controlled by a software agent's society. In this way the system implements virtual redundancy, a low cost reliable configuration: if a part of the system fails, other parts will assume its tasks. Therefore, when a part (router) of the system is in failure, the agents will use consensus in order to reallocate the tags to the rest of the routers, with respect of the communication quality requirements. In failure state, it is preferably to achieve an unbalanced load of the AP's instead of a physical redundancy o a balanced system. The system can be connected and communicate with other similar systems, therefore it is possible to obtain wide area monitoring systems.

Acknowledgment

The authors would like to thank Cosmin Arad and Tallat M. Shafaat for their professional support.

References

[1] Stuart Russell and Peter Norvig, 2002. "Artificial Intelligence: A Modern Approach", Englewood Cliffs, NJ: Prentice Hall, pp: 375-410.

[2] Hyacinth S. Nwana, "Software agents: An Overview", Knowledge Engineering Review, at Cambridge University Press, 1996, Vol.11, No.3, pp. 1-40.

[3] Maes and Pattie, 2005. "Designing Autonomous Agents", Cambridge, MA: MIT Press, pp: 102-110.

[4] Jeffrey M. Bradshaw, "Software Agents", MIT Press Cambridge, MA, USA, 1997.

[5] Zhong Zhang and McCalley, 2004. "Multiagent System solutions for distributed computing, communications , and data integration needs in the power industry", Power Engineering Society General Meeting, IEEE, 1: 45 - 49.

[6] Adina Morariu, Silviu Folea, Honoriu Valean, "Reliable Agent Based Monitoring System", 24th International Conference on Computers and Their Applications, April 8-10, 2009 Holiday Inn Downtown-Superdome New Orleans, Louisiana, USA, pp. 99-105, INSPEC and DBLP.

[7] Klusch, M, Sycara, K., "Brokering and Matchmaking for Coordination of Agent Societies: A Survey. In: Coordination of Internet Agents, A. Omicini et al. (eds.), Springer, 2001.

[8] Leslie Lamport, „The part-time parliament", ACM Transactions on Computer System, 16(2):133-169, May 1998;

[9] Leslie Lamport, "Paxos made simple". SIGACTN: SIGACT News (ACM Special Interest Group on Automata and Computability Theory), 32:1825, December 2001;

[10] Butler W. Lampson, "The ABCD's of Paxos", Presented at Principles of Distributed Computing, 2001

[11] Marco Primi, "Paxos made code, Implementing a high throughput Atomic Broadcast", Master's Thesis, Faculty of Informatics of the University of Lugano

[12] C. Arad, J. Dowling & S. Haridi, "Developing, simulating, and deploying peer-to-peer systems using the Kompics component model", in COMSWARE'09.