

A Framework for Mobile-Service Based Co-ordination of Embedded Web Agents in Intelligent Buildings

Zhi-Gang Deng and Huosheng Hu

Abstract -- Embedded agents can be used as building blocks or components in the construction of an Intelligent Building (IB). However it remains a great challenge on how to implement this idea in a distributed networking environment. In this paper, we present a co-ordination framework for web-enabled embedded agents to optimize the dynamic allocation of resources and the mobility of services in an intelligent building environment to achieve real-time performance. This co-ordination framework is mobile-service based and composed of two parts: a triangle-link algorithm and a double-layer protocol. A case study is presented to show its implementation.

Keywords -- Intelligent Building, Embedded web agents, Multi-agent systems, Mobile services, Co-ordination

I. INTRODUCTION

The motivation for the development of Intelligent Buildings is to apply computer and network technologies to the information management and energy control of the buildings we inhabit. Up to now, the major focus has been placed on how to build a distributed control system that is able to respond to the need of its occupants and to optimize energy-consumption and work efficiency [1].

Multi-agent systems [12] have proved to be a powerful framework for the development of complex distributed systems for IB applications. For instance, individual rooms in an IB can be considered as physical control systems, controlled by embedded agents that are distributed within the building [4]. These embedded agents are designed to process sensory information associated with building services (e.g., temperature, light, air, safety etc) and to provide a level of intelligent control to each room, including reasoning, adaptation and learning. They closely connect sensors to actuators within the room in order to implement the specified tasks. It is a great challenge for embedded agents to handle all the tasks required by IB, such as controlling environment variables, monitoring sensors and other devices, learning occupant behaviors, and communicating with other agents whilst operating at the limited hardware resource [2].

As the number of agents embedded in an IB increases, planning and control of the system becomes increasingly complex. The methods to handle such complexity include a centralized control method and a decentralized control method. More specifically, in a centralized control method all planning and decision-making functions are handled by a single control center [11]. Each agent has the limited computing power for simple operation, a number of sensors

for monitoring, the actuators for operation, and the communication facilities for data exchange with the control center. All the information management and decision making in the system are done by the control center and conflicts among multiple agents are easily solved. This method has been widely adopted in current IB. One major disadvantage of the system is that the whole system will stop functioning immediately if the control center fails.

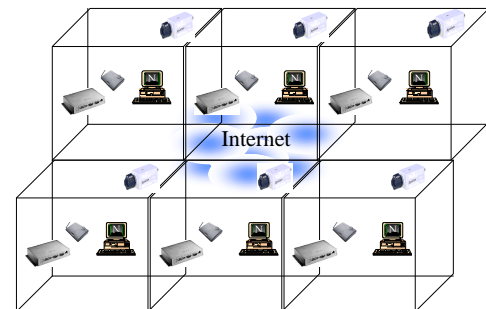


Figure 1 Example of a Decentralized IB System

In contrast, a decentralized control method is to equip each room with multiple sensors and embedded computers in order to sense its environment, build models, and plan actions. Figure 1 presents a simplified example of decentralized IB systems with multiple room-based agents. In any unforeseen situation, the agent is able to find a solution without waiting for commands from the control center. The function of the control center is only limited to the allocation of tasks in the system and the broadcasting of the reference information. Inter-agent communication becomes necessary since competition for resources should be avoided and sharing experience could improve system performance.

In both control methods, to realize the co-operation of multiple IB agents, three main issues need to be addressed: (i) how to appropriately divide the functionality of the system into multiple agents, (ii) how to manage the dynamic configuration of the system in order to realize co-operative behaviors, and (iii) how to achieve co-ordination and learning for a team of IB agents. This paper is to address these issues using a decentralized approach.

The rest of this paper is structured as follows. In section 2, a brief introduction of co-operative behaviors and their implementation is presented. A paradigm of IB embedded web agent is presented in section 3. Section 4 proposes a framework of mobile-service [17] based co-ordination. The design of triangle-link co-ordination algorithm is given in section 5. To avoid the message collision and deadlock, a double-layer co-ordination protocol is proposed in section 6. Section 7 gives a case study to show the implementation of

the proposed framework. Finally, conclusions and future work are summarized in section 8.

II. CO-OPERATIVE BEHAVIORS IN IB EMBEDDED AGENTS

A primary aim in the development of teams of co-operative embedded agents in IB is to synthesize their co-operative behaviors, which are to accomplish missions that cannot easily be achieved with individual agent [12]. For many application tasks, even in some environments that appear simple enough, it is still very difficult to determine the behavior of a multi-agent system at the time of its design [6][7]. This would require, for instance, that the designer knows in advance which environmental requirements will occur in the future, which agents will be available at the considered time, in order to plan their interaction in response to these requirements. Enumeration of all possible states of a multi-agent system would inevitably cause a combinatorial explosion.

We propose a modular approach to solve this difficulty. In other words, we firstly synthesize basic co-operative behaviors required for the co-ordination of multiple IB agents based on the reactive architecture proposed by Brooks [5]. Then, other co-operative behaviors can be gradually added during the future stages of our research. This bottom-up approach will make IB tasks easy and feasible.

Currently, several co-operative behaviors are identified for room-based agents as follows:

- Safeguard Behavior – to keep a safe environment for occupants in a room being controlled, and to protect any unexpected intruders.
- Communication Behavior – to communicate with other agents or exchange information with a control center in order to make co-operation of multiple agents feasible.
- Energy-saving Behavior – to control the energy consumption and achieve utility efficiency such as gas, water and electricity.
- Comfort Behavior – to control lighting, ventilation, humidity, fresh air, audio, and video automatically in order to make occupants feel comfortable.
- Modeling Behavior – to collect data from different sensors, and build environment model continuously.
- Learning Behavior – to learning occupant behaviors and to be adaptive to occupant's needs dynamically.

The key challenge issue in IB is how to deal with both the uncertainties in a dynamically changing environment and non-deterministic reaction from occupants. For example, heating and lighting could go wrong. Smoke alarm may be malfunction. Window glass may be broken. Occupants' demand may change from time to time. Therefore, the proposed co-operative behaviors are intended to handle these by reacting quickly from the stimuli gathered from the IB environment by distributed sensors.

Some problems in the development resulting from complexity of a multi-agent system can be avoided, or at least reduced by providing the agents with the ability to adapt and to learn. In our case, learning information comes

from occupant reactions and other agents. Co-ordination among agents forms an important foundation for learning.

III. IB EMBEDDED WEB AGENTS

The Internet appears to be the best choice for the communication infrastructure in IB applications due to its low cost and worldwide availability [5]. All embedded agents distributed throughout a building can be connected via such a network with IP (Internet Protocol) based connection and co-ordination or information sharing. In other words, IB is a typical embedded networking application.

Currently, there are a number of embedded networking devices, such as SSV NetPC [18] and Dallas TINI [19] boards, which are available for connecting distributed sensors and actuators to Ethernet and allowing them to be monitored, controlled and managed through standard network applications such as web browsers. The combination of board-based I/O capability, a TCP/IP network protocol stack and a powerful object-oriented programming environment enables programmers to quickly create applications that provide not only local control, but also the ability to remotely manage IB devices. Based on these embedded networking devices, IB embedded web agents (i.e., web-enabled IB embedded agents) can be easily built, as shown in Figure 2.

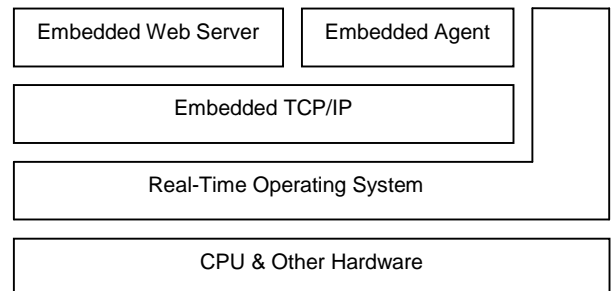


Figure 2 IB Embedded Web Agent

In our case, IB embedded web agents have three characteristics as follows.

- Embeddable -- They are capable of being built into resource-constrained embedded networking devices (i.e., small computational footprint and high integration).
- Web-Enabled -- They are so functioned by integrating with a micro web server that it is possible for web-based (i.e., IP-based) co-ordination and even control.
- Multi-Role Based -- Each IB agent can be either room-based or domain-based. On the other side, it can also play a part in either requesting services as a client-agent or in providing services as a server-agent. An IB embedded web agent can serve as one or more roles simultaneously during the course of co-ordination.

The flexibility and versatility of IB embedded web agents can make significant contributions to our mobile-service based co-ordination mechanisms as described in details below.

Co-ordination is an essential component of multi-agent systems [8][9]. Contention over shared resources and the desire to employ remote information may require some sort of information transfer among agents to be resolved. Co-ordination in multiple IB embedded web agents takes place when one IB embedded web agent acting as a client (i.e., client-agent) to issue request exchanges information with another IB embedded web agent acting as a server (i.e., server-agent) to provide information services. A request is generated by the co-ordinator of client-agent and a message packet sent to the server-agent in the form of the communication language [7]. At the server-agent side, the receipt of a request for assistance causes server-agent's co-ordinator to invoke related functions to process the services.

For instance, when the client-agent has no knowledge about a certain environmental requirement and sends a request for new behavior (i.e., rule related to environmental variables, devices and room occupant), which conforms to one of safeguard, communication, energy-saving, comfort, modeling and learning types, to the server-agent, the server-agent will take an active part in co-operative service for helping client-agent get the effective behavior to be involved. All the activities for inter-agent co-ordination need to be effectively and efficiently organized and controlled. We thus address the design of a co-ordination framework for multi-web-agent based IB to organize and control multi-web-agent co-ordination in an optimum way in a distributed IB environment.

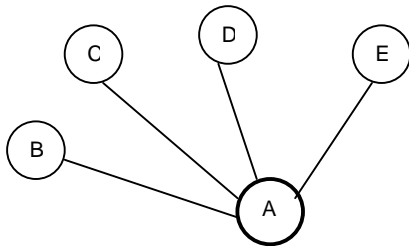


Figure 3 A Coarse-Grained Framework of IB Multi-Web-Agent Co-ordination

Figure 3 shows a framework of multi-web-agent co-ordination. In this design, when agent A encounters an inexperienced environmental requirement, it would send a help request to its neighbor agents B, C, D and E. Although it is based on decentralized autonomous co-ordination, this framework would still cause some problems. Assuming that agent A asks an agent which also has no knowledge about the same environmental requirement and then this agent has to ask for help from other agents nearby. In this case, real-time performance could be very difficult to guarantee especially when the response-agent is physically and logically far away from the request-agent.

On the other hand, it would be difficult to reduce the overhead of the whole process when the request-agent coordinates with several other agents simultaneously. It is not difficult to imagine what would happen when the resource-constrained embedded-internet device will not only allocate part of resources to handle the internal processes but also guarantee the external processes of co-ordination. So this

framework will result in expensive overheads of multi-web-agent co-ordination.

IV. A FRAMEWORK FOR MOBILE-SERVICE BASED CO-ORDINATION

In view of the requirements of the co-ordination in multiple embedded-web-agent based IB, the following operations have been taken into account.

- Service Mobility

Due to the fact that human behaviors are not restricted within only one room, building-wide services to different occupants, such as temperature, light, are required to be adaptive. By deploying mobile IP [14], it is possible to make the services provided by room-based agents virtually mobile.

- Responsibility Distribution

In order to distribute the responsibility of agents to be involved to several really inevitable agents, according to their respective roles, all the IB embedded web agents can be classified into three groups – room-based client-agent, room-based server-agent and domain agent. The server-agent, in most cases, can be considered as the first agent involved to a specific occupant originally. When the occupant changes his/her activity from one room to another one in IB, one more or two more agents (i.e., case 1 and case 2 in Figure 4) will get involved depending on whether the new activity environment of the occupant belongs to the same pre-divided domain which is composed of a set of agents with most-closely physical (i.e., neighboring to each other) and logical (i.e., common properties) relationship to each other.

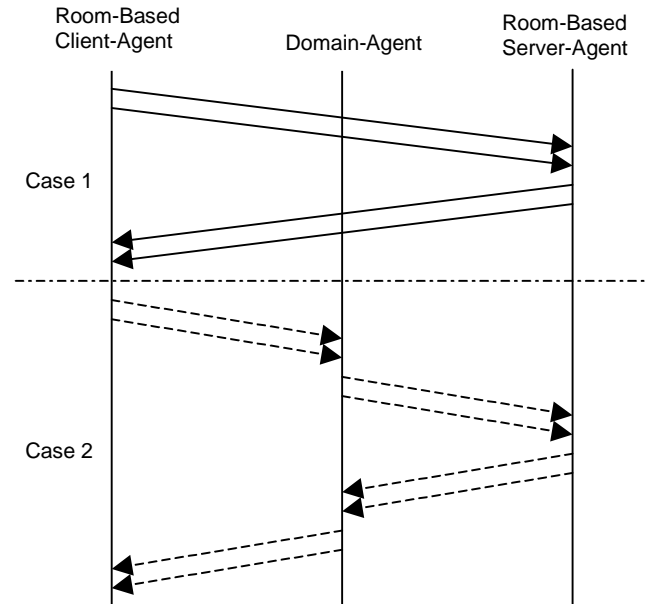


Figure 4 A Framework for Mobile-Service Based Co-ordination of IB Multi Web Agents

- Triangle-Link Co-ordination

When the occupant is located in a domain which is different from its original one, three IB embedded web agents is supposed to get involved in the whole co-ordination process.

The domain agent, serving as an intermediate agent (i.e., a proxy) with both roles of client-agent and server-agent. It serves as a server-agent to the agent in a room where human behaviors happen, whilst acting as a client-agent to the room-based agent to which the occupant's original personalized services have been registered.

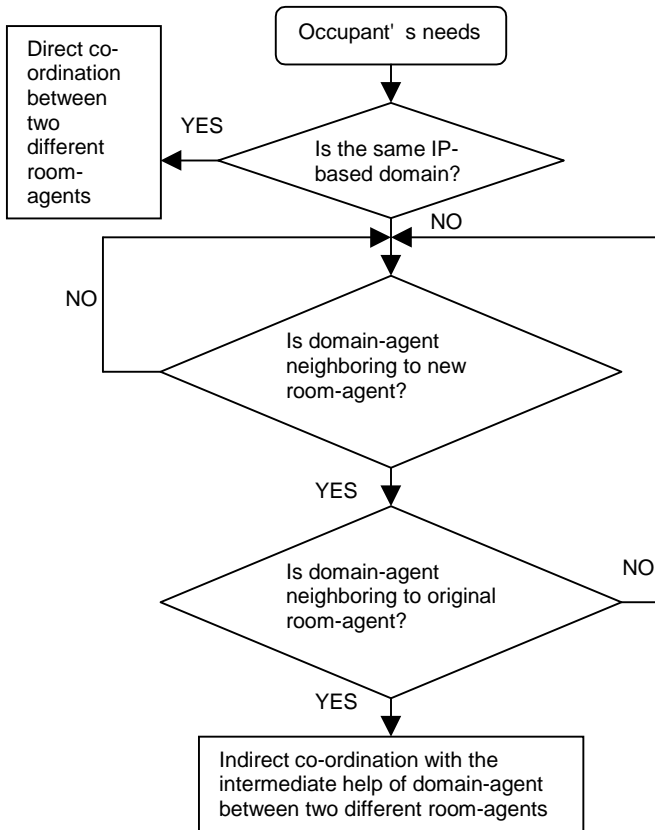
Our co-ordination framework is deployed practically based on a triangle-link algorithm and double-layer protocol [11][12][13], both of which will be described in detail in the following sections.

V. TRIANGLE-LINK CO-ORDINATION ALGORITHM

The control algorithm should take the following requirements into consideration for the purpose of optimizing the use of system resources.

- Due to the fact that occupant moves around yet each agent is fixed in a room, the services specific for each occupant are inevitably desired to be mobile in a distributed IB environment.
- Communication agents contained in a specific domain in a distributed IB environment should have common properties based on which the division of domain has been performed. This division is involved in network management which can form another independent research topic. Whatever the division rule is chosen, a communication agent is allocated to each domain, which is responsible for the registration of the service request that is available in a remote agent residing somewhere else beyond the local domain.

The algorithm can be described as follows.



The whole algorithm can also be represented as the following arithmetic description.

N_a denotes the sets of room-agent A's neighbors
 N_b denotes the sets of room-agent B's neighbors
 N_c denotes the sets of agents neighboring to both room-agent A and B
 C denotes a domain-agent
 $\forall \text{ agents} = \{A, B, C, D, \dots\}$
 $N_a = \{B, C, D, \dots\}$
 $N_b = \{C, \dots\}$
 $C \subset N_c = (N_a \cap N_b)$

If

$$\lim_{n \rightarrow +1} \left(\sum_{i=0}^n r_i(a, c) + \sum_{j=0}^n r_j(b, c) \right) = \sum_{k=0}^n r_k(a, b)$$

where r denotes the sum of multi-paths between two nodes in a distributed network then, the co-ordination between a-c-b can be considered shortest triangle-link based. Whether or not the actual co-ordination link is short enough depends on how close it is to the virtual direct co-ordination link between request-agent and response-agent without domain-agent to be involved.

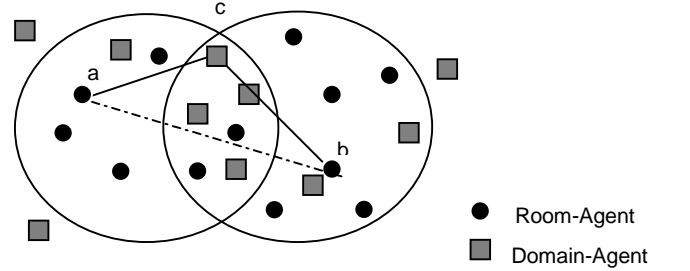


Figure 5 The Schema of Co-ordination Algorithm

VI. DOUBLE-LAYER CO-ORDINATION PROTOCOL

The protocol provides the ability to manage the whole process of co-ordination in order to avoid the message collisions, deadlocks, etc. In order to be suitable for a wide application, the following design criteria of inter-agent co-ordination protocol should be taken into account.

- **Portable and Extensible**
This co-ordination protocol should not be concerned exclusively with the specific domain of Intelligent Building. Instead it should be available to a whole range of environments including home, car, space vehicles and other forms of "building space". On the other hand, the protocol should be independent of a specific communication language with the definition of widely compatible specification.
- **IP-Based**
IP-based intelligent building is coming as a new trend, which can provide an extensible application, even wireless connection. Designing this protocol based on IP can facilitate the distributed application development using a wide range of technologies such as RMI, CORBA, Jini, and so on.
- **Stateless**
Each client-agent request is serviced by a server-agent independently of any other requests, and no information about a client-agent is maintained at server-agent after a

connection has closed. This is based on the consideration of effective usage of system resources.

- Tunneled

An exclusive “tunnel” will be generated between two web-agents to be involved in order to guarantee the security of message transmission.

The message packet consists of the following information.

- ID: the home address of agent based on mobile IP
- Location: the current address in the range of a network
- Type: message type, such as request or response
- Behavior: the detailed service information attached corresponding to the message type
- Priority: This number is concerned with indicating the priority of the message, especially related to emergency processing

The protocol is presented as a double-layer model consisting of request layer and response layer, which is responsible for performing all of different activities, resulted from inter-agent co-ordination.

The request layer is concerned with request-related actions by client-agent and request-related actions by server-agent. The response layer is involved in response-related actions by server-agent and response-related actions by client-agent. Both layers specify in detail the action management specification of inter-agent message-passing co-ordination in order to guarantee not only the whole course of co-ordination can be performed effectively and efficiently but also the message can be transmitted appropriately and safely.

- Request Layer

Request is a process by which IB client-agents are able to acquire information about behaviors available from IB server-agent, including their identity, message type, behaviors (i.e., services available to providing) and privileges. The behaviors an agent provides are decomposed into six functions: safeguard, communication, energy-saving, comfort, modeling and learning behaviors (i.e., sets of rules). A request specifies a series of behaviors, which are described in the behavior part of a message packet to be transmitted.

At the client-agent side, actions involved in a request fall into three categories: *RequestStart*, *RequestCancel* and *RequestUpdate*. *RequestStart* is used for starting a request. *RequestCancel* is employed for canceling the previous request in the case of, for example, client-agent has made some mistakes in the request sent to server-agent. *RequestUpdate* is utilized for telling server-agent the changes of the request for behaviors.

At the server-agent side, actions concerned with a request can be divided into three types: *RequestRegister*, *RequestDecline* and *RequestForward*. *RequestRegister* is used by server-agent to notify client-agent of the registration of the request. *RequestDecline* is used for telling client-agent that the server-agent gets no spare time to handle its request. When server-agent has no ability to handle the request from the client-agent, *RequestForward* will be used by the server-agent to notify the client-agent of the request forwarding so

that it can construct a new connection with another server-agent and break the connection with the old one.

- Response Layer

Response is a process by which IB server-agents are able to return the results after handling the related services to IB client-agents. The results depend on two different cases that server-agent is succeeded in providing the related services and that server-agent has no helpful provision. In order to correspond to request layer, response layer is defined separately according to client side and server side. A response specifies a series of behaviors, which are described in the behavior part of a message packet to be transmitted.

At the server-agent side, actions concerned with a response are divided into three types: *ResponseStart*, *ResponseCancel* and *ResponseComplete*. *ResponseStart* is used for starting a response and telling client-agent that there is available provision from server-agent. *ResponseCancel* is responsible for asking client-agent to cancel the last message since there are some errors in the results, which have been sent out. *ResponseComplete* is use for telling client-agent that all the results have been sent and they should finish the whole course of co-ordination by breaking the connection between client-agent and server-agent.

At the client-agent side, actions involved in a response fall into two categories: *ResponseReceive* and *ResponseRepeat*. *ResponseReceive* is employed to notify server-agent that client-agent has been succeeded in receiving message so as to deregister itself. *ResponseRepeat* is responsible for telling server-agent to send the results again because of, for example, incomplete message or incorrect message.

VII. A CASE STUDY

This case study is to show an example of practical application of the proposed co-ordination framework. The scenarios are assumed to happen in a hospital, as shown in Figure 6.

- A patient is moved from room 201 to room 101. Room 101 is a physical and logical neighbor of room 201 based on our IB embedded-web-agent co-ordination framework. Because there is no recorded experience related to this patient in agent 101, it needs to ask other agents for appropriate environmental parameters in order to provide comfort behaviors specific to this patient.
- Based on co-ordination protocol, agent 101 sends a *RequestStart* message to neighbor agent 102 in order to initialize the co-ordination with agent 102.
- Since agent 102 is co-ordinating with another agent, it has to send back *RequestDecline* message to agent 101 to tell agent 101 that it is too busy to help.
- Agent 101 then tries requesting one another neighbor -- agent 202 with *RequestStart* message.
- Fortunately, agent 202 is free and sends *RequestRegister* message back to notify agent 101 that its request has been registered successfully.

- However, agent 202 has also no knowledge about the related environmental requirements specific to this patient. It will then forward agent 101's request to agent 201.
- After agent 202 receives *RequestRegister* message from agent 201, it sends *RequestForward* message to agent 101 to notify the change of the co-ordination partner so that agent 101 can construct a new connection with agent 201.

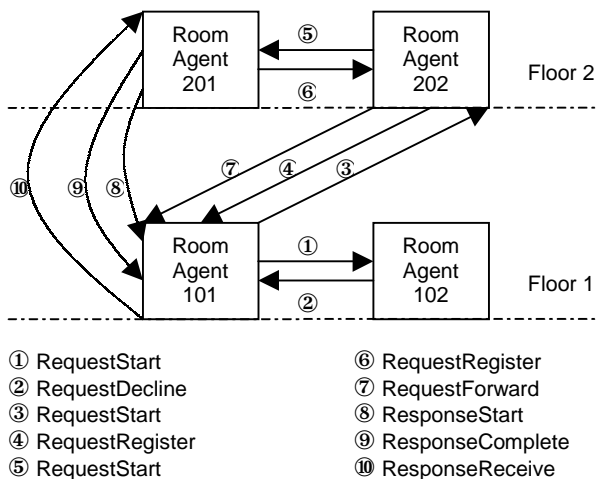


Figure 6 A Case Study of Co-ordination Framework

- After agent 201 finishes handling the request, it sends agent 101 the *ResponseStart* message to notify it to receive the required information.
- After agent 201 finishes sending all the information to agent 101, it sends *ResponseComplete* message to agent 101 to end the co-ordination. And agent 101 will send *ResponseReceive* message back to agent 201 as a de-registering notification.

VIII. CONCLUSIONS AND FUTURE WORK

Since the proposed IB embedded web agents will be implemented in commercially available embedded networking devices with the constrained resource, the co-ordination among these embedded agents becomes extremely important for the real-world applications. A framework for such co-ordination is proposed in this paper and a case study shows its implementation.

This mobile-service based co-ordination framework support both wired and wireless networking LAN in which the served objects (human or other mobile devices) are distributed. Any existing space can be also viewed as being composed of one or more real or virtual intelligent environments. Both the commercial and research potentials are obviously massive and thus there is no shortage of impetus to support the further work that will be needed to turn these exciting ideas into reality. This research work is at its design stage and the implementation is underway.

Acknowledgements: We would like to thank the members of the Intelligent Buildings Group at Essex University, especially Victor Callaghan and Graham Clarke for their comments. Our thanks also go to Sam Steel for his criticism.

References

- [1] M. Boman, P. Davidsson, N. Sharmas, K. Clark and R. Gustavsson, Energy Saving and Added Customer Value in Intelligent Buildings, Proc. PAAM' 98, Editor H.S. Nwana, pages 505-516, 1998
- [2] V. Callaghan, G. Clarke, M. Colley, H. Hagra, "A Soft-Computing DAI Architecture for Intelligent Buildings", Journal of Studies in Fuzziness and Soft Computing on Soft Computing Agents, Physica-Verlag-Springer, July 2000.
- [3] L. M. Camarinha-Matos and W. Vieira, "Intelligent Mobile Agents in Elderly Care", International Journal of Robotics and Autonomous Systems, Vol. 27, pages 59-75, 1999.
- [4] M. H. Coen, "Design Principles for Intelligent Environments", Proceedings of the 1998 National Conference on Artificial Intelligence, 1998.
- [5] B. Dalton and K. Taylor, "A Framework for Internet Robotics", IROS'98, Victoria, B.C. Canada, Oct. 12-17 1998.
- [6] M. D'Inverno, D. Kinny, M. Luck, "Interaction Protocols in Agentis", Proceedings of the Third International Conference on Multi-Agent Systems, 112-119, IEEE Press, 1998.
- [7] F. Cayci, V. Callaghan, G. Clarke, A Distributed Intelligent Building Agent Language (DIBAL), Proceedings of the 6th International Conference on Information Systems, Analysis and Synthesis, Orlando, Florida, July 2000
- [8] M. Fukuda, L. F. Bic, M. B. Dillencourt, F. Merchant, "Distributed Coordination with MESSENGERS", July 1997.
- [9] B. Horling and V. Lesser, "Using Diagnosis to Learn Contextual Coordination Rules", Umass Computer Science Technical Report, 1999.
- [10] H. Hu and M. Brady, Towards Advanced Mobile Robots for Manufacturing, a Chapter in the Book "Artificial Intelligence and mobile robots", Editors D. Kortenkamp, P. Bonasso, and R. Murphy, pages 297-321, MIT Press, March 1998
- [11] H. Hu, D. Gu, M. Brady, A Modular Computing Architecture for Autonomous Robots, *Int. Journal of Microprocessors and Microsystems*, Vol. 21, No. 6, pages 349-362, March 1998
- [12] N. R. Jennings, K. Sycara, M. Wooldridge, "A Roadmap of Agent Research and Development", Journal of Autonomous Agents and Multi-Agent Systems, Vol. 1, pp. 7-38, 1998.
- [13] S. Kristoffersen and F. Ljungberg, "The Architecture and Protocol of MOSCOW: Mobile Sharing and Co-ordination Of Work", 1999.
- [14] E. Oliveira, K. Fischer, O. Pankova, "Multi-Agent System: Which Research for Which Applications", International Journal of Robotics and Autonomous Systems 27, pp. 91-106, 1999.
- [15] C. Perkins and A. Myles, "Mobile IP", SBT/IEEE International Telecommunications Symposium, Rio De Janeiro, Brazil, 22-25 August 1994.
- [16] S. Sharples, V. Callaghan, G. Clarke, "A Multi-Agent Architecture for Intelligent Building Sensing and Control", International Journal of Sensor Review, May 1999.
- [17] T. D. Hodes, R. Katz, E. Servan-Schreiber, L. A. Rowe, "Composable Ad-hoc Mobile Services for Universal Interaction", Proceedings of the 3rd ACM/IEEE Int. Conf. on Mobile Computing, Budapest, Hungary, September 1997.
- [18] SSV, <http://www.ssv-embedded.de/ssv/pc104/p158.htm>
- [19] TINI board, <http://www.ibutton.com/TINI/index.html>