J. Sens. Actuator Netw. **2013**, *2*, 557-588; doi:10.3390/jsan2030557

OPEN ACCESS

Journal of Sensor and Actuator Networks

ISSN 2224-2708

www.mdpi.com/journal/jsan

Article

A Multi-Agent-Based Intelligent Sensor and Actuator Network Design for Smart House and Home Automation

Qingquan Sun [1], Weihong Yu [2], Nikolai Kochurov [1], Qi Hao [1,*] and Fei Hu [1]

[1] Department of Electrical and Computer Engineering, University of Alabama, Tuscaloosa, AL 35487, USA; E-Mails: qsun3@crimson.ua.edu (Q.S.); nick.kochurov@gmail.com (N.K.); fei@eng.ua.edu (F.H.)

[2] College of Transportation and Management, Dalian Maritime University, Dalian 116026, China; E-Mail: yuwhlx@163.com

* Author to whom correspondence should be addressed; E-Mail: qh@eng.ua.edu; Tel.: +1-205-348-2618; Fax: +1-205-348-6959.

Received: 9 June 2013; in revised form: 2 August 2013 / Accepted: 6 August 2013 / Published: 19 August 2013

Abstract: The smart-house technology aims to increase home automation and security with reduced energy consumption. A smart house consists of various intelligent sensors and actuators operating on different platforms with conflicting objectives. This paper proposes a multi-agent system (MAS) design framework to achieve smart house automation. The novelties of this work include the developments of (1) belief, desire and intention (BDI) agent behavior models; (2) a regulation policy-based multi-agent collaboration mechanism; and (3) a set of metrics for MAS performance evaluation. Simulations of case studies are performed using the Java Agent Development Environment (JADE) to demonstrate the advantages of the proposed method.

Keywords: multi-agent system; smart house; intelligent sensor-actuator networks; JADE

1. Introduction

A smart house consists of a variety of embedded sensors/actuators, distributed/mobile computing units and energy harvesting devices. The smart house technology enables home automation functions, such as lighting/venting/air conditioning control, multimedia, security, healthcare, *etc*. A smart house can understand environmental contexts, such as weather, time and location, as well as human contexts, such as identity, activity and behavior. Based on the contextual information, the smart house can provide services, such as energy efficiency control, security monitoring and medical assistance, as shown in Table 1.

Table 1. The functions and devices in smart house systems.

Functions	temperature/illumination/ventilation control, healthcare assistance, security management
Sensors	thermal, light, temperature, acoustic, photo, pressure, medical
Actuators	switch, heater, air-conditioner, lamp, speaker, TV
Operating systems	TinyOS, SunSPOT, Android
Computing units	smart phone, tablet, microcontroller

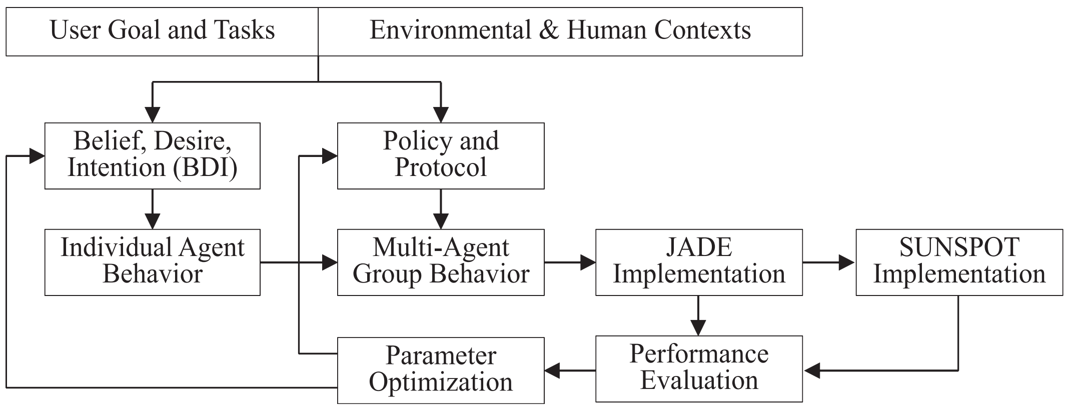
However, an integration of such a large number of distributed devices will result in a complicated system, in which different components may have conflicting objectives. Besides, multiple sensing/computing platforms, such as Android, TinyOS and SunSPOT, could be used. A promising solution is to utilize multi-agent technology for system design and management. An agent is an independent hardware/software co-operation unit, which can understand the situation and respond to stimuli according to predefined individual behaviors [1]. Each agent has its own special functionalities: such as sensing, action, decision and database. Multiple agents, when deployed together, can automatically share information with each other and form collaborative group behaviors to achieve the common goal [2]. However, in order to develop smart houses based on multi-agent systems (MASs), the following technical challenges have to be resolved:

1. development of a generic approach that can systematically generate individual behavior for different agents;
2. development of a framework that can control group behavior of multiple agents;
3. development of a set of metrics to evaluate the performance of individual and group behaviors of agents;

Conventional approaches defining individual agent behavior are based on each agent's functionalities and roles. The lack of a unified methodology usually results in *ad hoc* procedures for agent system design. We have been developing a belief, desire and intention (BDI)-based agent behavior generation method. The user's goal, tasks and contextual information can be converted into a set of belief, desire and intention models. The individual agent behavior is then systematically developed based on BDI models. Group interaction protocols and resource management policies are also generated to control group behavior of multiple agents. For performance analysis and optimal design purposes, individual and group behaviors can be modeled as finite-state machines, and their efficacy and efficiency can be analyzed and evaluated through Petri-net methods.

In this paper, we present a multi-agent design framework for smart house applications, as shown in Figure 1. A BDI-based agent model is proposed for individual behavior formulation. A regulation policy-based group behavior control is proposed based on Petri-net analysis. A set of performance metrics is presented to evaluate individual and group behaviors for system optimization. Case studies are performed using the Java Agent Development Environment (JADE) tools.

Figure 1. The proposed design and evaluation procedure of multi-agent systems for smart house technology.



This paper is organized as follows: Section 2 reviews related work. Section 3 describes the system architecture and problem statement. Section 4 summarizes the individual and group behavior generation and control methods. Section 5 presents the JADE implementation. Section 6 discusses the evaluation metrics. Section 7 provides related results and discussions. Section 8 concludes the paper and outlines future work.

2. Related Works

Smart Home technology aims to provide a flexible, comfortable and energy-efficient home environment to improve upon the quality of life for residents through the use of sensor-actuator networks and information processing techniques. Most smart home designs emphasize the use of artificial intelligence algorithms [3–8]. Sensor networks [9–11] and multi-modal information fusion techniques [12] have also been utilized to perceive situations and achieve automatic control. However, these systems usually involve large amounts of information processing and centralized implementation schemes. Their mechanisms are not optimal to achieve distributed, scalable and robust smart house automation.

On the other hand, distributed data acquisition and in-network processing mechanisms have been proposed to improve sensor-actuator networks' efficiency and speed in information acquisition and situation perception [13,14]. Techniques, such as adaptive sampling, in-network aggregations, runtime reconfiguration and multi-task query are used to enhance query performance with reduced data throughput and power consumption. Furthermore, [15] provides a building management framework for multiple node platforms, which can dynamically capture the morphology of the building and management multiple node groups with different functionalities. These technologies assume that: (1) only simple operations are involved for data acquisition and actuation; (2) only basic computing algorithms are used for in-network processing; and (3) the whole system can be homogeneously developed, except for the base station. However, many sensors/actuators for home automation require complicated procedures of calibration, configuration and cooperation. Besides, sensors and actuators impose different requirements upon hardware and computing capabilities. Machine learning techniques are required for in-network processing to achieve context and situation perceptions. Learned context and knowledge models need to be stored in databases for easy access. Therefore, these distributed units have to be enhanced with different computing, communication, configuration and storage capabilities, without losing the system scalability and robustness against local failures.

Agent-based approaches have been used to develop scalable smart house and home automation technology [16–18]. A typical smart house includes sensing agents (e.g., temperature, floor sensors), administration and decision agents (e.g., interface, butler and reasoning units), action agents (e.g., effector, actuator and housekeeper), and database agents (e.g, context and knowledge bases). Sensing agents are used to collect information about the environment, resources and human activity. Administration and decision agents manage interactions among agents, dispatch events, perform reasoning and assign tasks. Action agents execute the tasks, reconfigure the system and provide services. Database agents accumulate information and knowledge from other agents' experiences. In the existing implementations, agents are designed according to their roles; there is no unified procedure for agent behavior design nor a mathematical model for agent performance analysis.

Forming and managing collaborations among multiple agents is a challenging problem. Both centralized and distributed methods have been developed [19–21]. In the centralized scheme, there is a control agent, which manages the information exchange among agents, coordinates their actions and resolves conflicts. However, centralized schemes are not scalable, and a failure of some agents will result in dysfunctions of the whole system. In the distributed scheme, there is no central control agent, but there is an agent management platform, which maintains task stacks and message queues and resolves the conflicts among resource requests. The whole platform is scalable to the number of agents. This platform is implemented among distributed computing units. Still, such a platform cannot solve the intrinsic conflicts among agent actions. Therefore, it is necessary to develop a set of mathematical models and tools for multi-agent performance analysis.

Petri-net technology has been developed to analyze the functionality and evaluate the performance of asynchronous, distributed, discrete systems [22]. It provides a set of graph based tools under a unified framework that can be used to study the reachability, liveness and boundedness of a system. Petri-net based strategies have been proposed for multi-agent scheduling [22–24]. A property-preserved Petri-net method has been developed to analyze the BDI model [25].

However, Petri-net-based analysis tools are only suitable to validate logic models of group behavior. A more detailed study of multi-agent group behavior, with physical models of sensors and actuators, needs a proper simulation/implementation platform. Among existing implementation platforms of multi-agent systems, such as Zeus, TAOM4E [26], JADE is the most popular one due to the use of the Java programming language and its compliance with IEEEstandards on Foundation for Intelligent Physical Agents (FIPA) [27]. Furthermore, its mobile versions (JADE Leap, SubSense) can be implemented on Android devices and Java-based SunSPOT sensor motes [28]. JADE has also been used in smart house and healthcare applications [29,30].

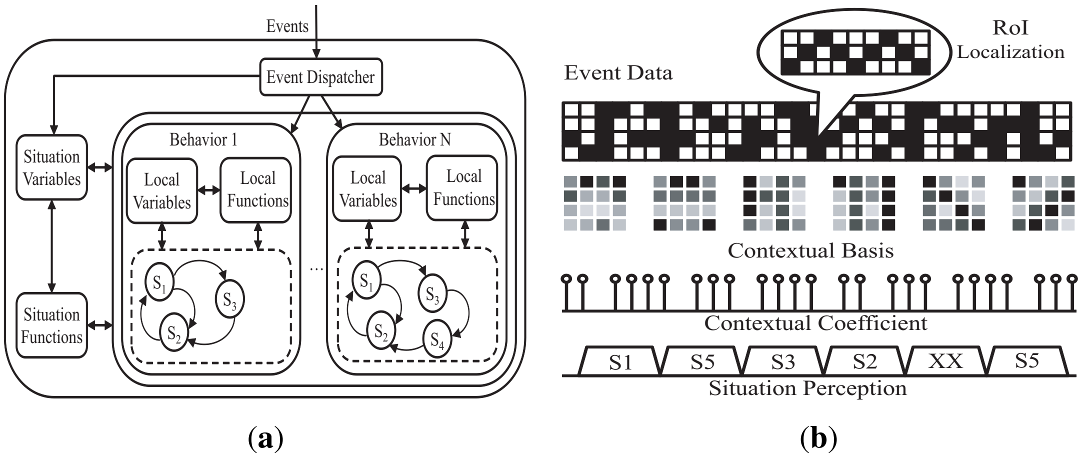
Our research aims to develop a unified framework for behavior design and control for different types of agents using belief, desire and intention models. We develop a set of methods for group behavior design and control based on regulation policies and Petri-net analysis. JADE tools are used to analyze the computational complexity, communication throughput and energy consumption of the proposed multi-agent system.

3. System Setup and Problem Statement

3.1. Intelligent Agents

An agent is an independent hardware/software co-operation unit with the following characteristics: goal-oriented, adaptive, mobile, social and self-reconfigurable. Each agent is capable of understanding its situation and adapts to changing environments through self-configuration, as shown in Figure 2a. The situation perception is achieved through learning and contextual modeling of event data, as shown in Figure 2b. After a set of contextual bases are learned from the high-dimensional event data, different scenarios can be represented by the clustered contextual coefficients. The agents are then able to percept the situation and localize regions of interest (RoIs) through identified scenarios [31–33]. Each agent has a behavior state machine and a behavior library; it chooses a certain behavior according to individual goals and other agents' behaviors.

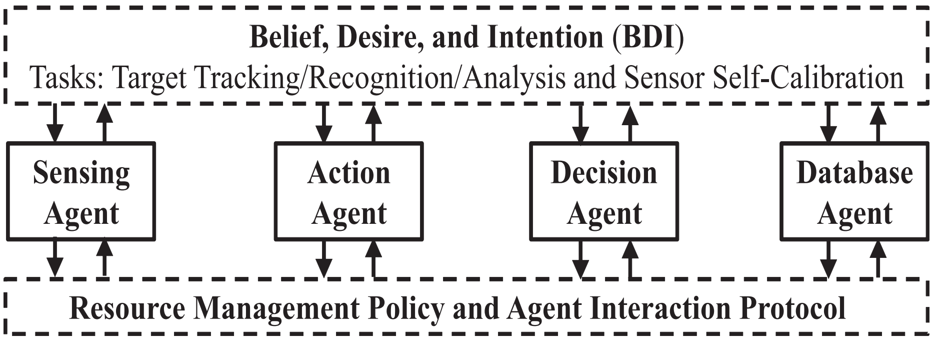
Figure 2. Illustration of (**a**) agent architecture; (**b**) situation perception from event sequences.



3.2. Multi-Agent Interactions and Collaborations

Multi-agent-based smart house technology aims at providing environmental control, security, entertainment and healthcare services for users with high energy efficiency. The system consists of four major types of agents: sensing, action, decision and database as shown in Figure 3. Such a multi-agent architecture will enable efficient, distributed information collection and processing, as well as system adaptation. Each agent has a set of beliefs, desires and intentions. All agents share beliefs through inter-agent communication. Given a set of beliefs, each agent can plan its short-time behavior, according to its understanding of the situation and recent events, to achieve the desired goal. The multi-agent platform provides an agent execution engine, as well as other related services, such as communication, naming, timer and resource management. There is a library for communication protocols, collaboration mechanisms and resource management schemes. Given a regulation policy and the user's goal, a communication protocol, a collaboration scheme and a resource management policy will be selected from the library.

Figure 3. Illustration of multi-agent collaboration.


Belief, Desire, and Intention (BDI)
Tasks: Target Tracking/Recognition/Analysis and Sensor Self-Calibration
Sensing Agent
Action Agent
Decision Agent
Database Agent
Resource Management Policy and Agent Interaction Protocol

3.3. User Interface and Event Dispatching

The user interface has two functions: (1) convert user's goal and environment and human context into a set of beliefs, desires and intentions for each agent; and (2) select a communication protocol, a collaboration mechanism and a resource management scheme based on the regulation policy provided by the user. For example, goal: house security; constraints: one month with an operation of 100 mW power consumption; tasks: measuring the gait biometrics of subjects inside the house. These inputs will be converted into selections of sensor modalities, algorithms/protocols, context/behavior templates and a resource management policy. There are two types of events: (1) external and (2) internal. External events represent different states of the environment and human subjects' behavior. Internal events represent different states of agents' behavior. These events will be dispatched to operating agents, and in each agent, events will trigger behaviors under certain situations.

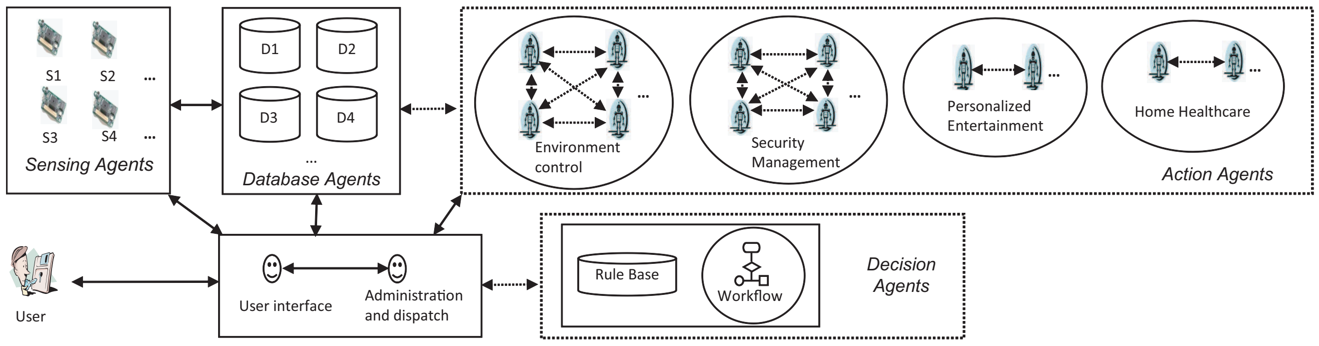
3.4. Problem Statement

The goal of this study is to develop a MAS framework with a set of design tools for smart house and home automation applications, which can:

1. design and control individual agent behaviors based on a belief, desire and intention model;
2. design and control multi-agent group behaviors based on a regulation policy; and
3. evaluate system performance and optimize design parameters based on a set of metrics.

The system diagram is illustrated in Figure 4. It can be seen that the operation of the whole system relies on the interaction and collaboration among various agents: sensing, action, decision and database. The individual and group behaviors of these agents are formulated by agent models and regulation policies. The design of agent models and regulation policies should be a strict procedure instead of an *ad hoc* one. Therefore, it is an important issue to develop a set of mathematical models that can describe the individual and group behaviors of agents. Based on these mathematical models, the collaboration performance of agents can be analyzed, and design parameters for the whole system can be optimized.

Figure 4. The proposed multi-agent system (MAS) architecture for smart house technology.



4. Agent Behavior Design and Evaluation

Our system design is mainly focused on three topics: (1) individual agent behavior; (2) multi-agent group behavior; and (3) agent behavior analysis. The developed MAS will enable the functionalities of a smart house technology, as listed in Table 1, through multi-agent collaboration.

4.1. BDI Model-Based Individual Agent Behavior

An agent is a system that is situated in a changing environment and chooses autonomously among various options available. Each agent needs to have an appropriate behavior (*i.e.*, actions or procedures to execute in response to events) based on a belief, desire and intention (BDI) model.

1. Beliefs represent the information the agent has about itself, other agents and environments.
2. Desires store the information on the goals to be achieved, as well as properties and costs associated with each goal.
3. Intentions represent action plans to achieve certain desires.

Beliefs of an agent are derived from its perception of situations (*i.e.*, the environment, itself and other agents). For example, an agent may have the following beliefs: (1) its own position, state, capability; (2) time, ambiance, temperature, location; (3) user's activity disposition, preference; and (4) other agents' state, conditions, capability. The desires of an agent are generated by user's input and its own beliefs. Compared with the user's goals, an agent's desires are more practical and achievable. Intentions consist of feasible action plans. Each set of plans is formed from beliefs, desires and old plans. An intention depends on the current situation and updates events and user goals. Given an intention, the agent will choose a certain behavior model from a library.

Figure 5 shows the generic BDI model for individual agent behavior design. The beliefs are modeled as a set of Bayesian networks to represent the relationships among random variables of the environment, user and agents. The belief updating function can recursively update the conditional probability functions of random variables according to up-to-date evidence. The option generation function can map the user's goals to a set of feasible goals through existing beliefs. This is equivalent to adding constraints to a set of objective functions. The feasibility filtering function can generate an action sequence through dynamic programming of the constrained objective functions. Based on the possible action sequence, one of the behavior models from the behavior model library will be selected.

Figure 5. BDI model based individual agent behavior design.

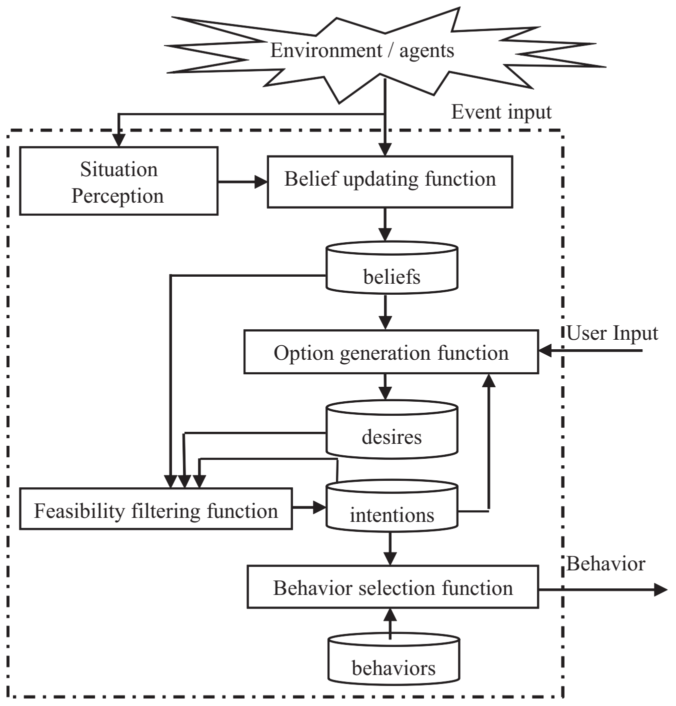


Table 2. belief, desire and intention (BDI) model-based sensing agent individual behavior for lighting control.

User input	**Illumination Control**
Belief	location: living room time: night number of humans: two subjects energy: moderate
Desire	Set up proper illumination conditions for each human activity with the goal of reducing power consumption
Intention	(1) use thermal sensor to identify human activity (2) use light sensor to detect the current illumination level (3) adjust illumination conditions to a proper level for that activity (4) choose the energy-efficient behavior mode
Behavior	(1) Sensor agent: (low resolution) sensing → (simple) processing → (low data throughput) transmission → (parameter) configuration (2) Action agent: (less frequent) communication → (parameter) configuration → (less frequent) command

Table 2 gives an example to illustrate the procedure of generating the behavior model. The user input is illumination control. The environmental beliefs include time, location, number of humans and energy conditions. Based on the beliefs, the desire is to setup proper illumination conditions with the goal of reducing power consumption. The intention plan includes four actions: (1) use thermal sensors to identify the subjects' activities; (2) use light sensors to detect the current illumination conditions; (3) change the illumination conditions to accommodate the current activity; and (4) choose an agent behavior model that saves energy. As a result, energy-efficient behavior models for sensing and decision agents are selected to perform this illumination control task.

4.2. Regulation Policy-Based Multi-Agent Group Behavior

When a group of agents work together, three major issues need to be addressed: (1) communication protocol; (2) collaboration scheme; and (3) resource management. A group of collaborative agents exchange various types of information, including service requests, service reports and the states of each agent. Therefore, a communication protocol should be formulated to ensure high efficiency in information exchange among agents. The collaborative activities of a group of agents may result in conflicts on schedule, resource and causality. Deadlocks, oscillation, unreachable tasks should be avoided for group behaviors of agents.

Figure 6 shows the process of generating a multi-agent interaction protocol. Each interaction protocol defines a group behavior. Given a user regulation policy, the multi-agent platform configuration function can generate a set of feasible communication protocols and collaboration schemes based on resource conditions and individual agent behavior models. Each regulation policy contains: (1) priority; (2) a task list; (3) a resource list; and (4) power consumption restrictions. Given a group behavior of multiple agents, possible conflicts can be checked and evaluated by using Petri-net methods. Among those protocols without any operational conflicts, the one with the highest perform-to-cost ratio will be chosen. A corresponding resource management policy will be adopted by the multi-agent platform.

Figure 6. Regulation policy-based multi-agent group behavior design.

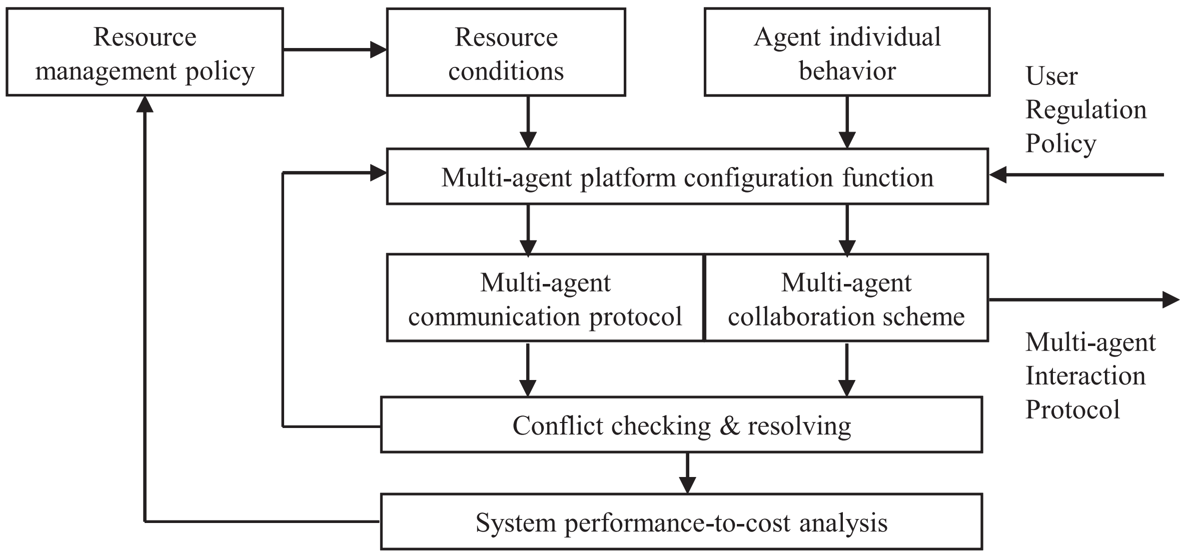

Resource management policy
Resource conditions
Agent individual behavior
User Regulation Policy
Multi-agent platform configuration function
Multi-agent communication protocol
Multi-agent collaboration scheme
Multi-agent Interaction Protocol
Conflict checking & resolving
System performance-to-cost analysis

Table 3 shows three typical regulation policies with different priorities: (1) response time; (2) quality of service (QoS); and (3) energy efficiency. For response time-oriented regulation policies, fast sensing agents will be chosen, and no database agent will be used; decision agents will use simple algorithms to make fast decisions. For QoS-oriented regulation policies, high resolution sensing agents will be chosen, and a database agent will be used to enhance the situation/context awareness of the sensing process, which can improve the information fidelity; more complicated action planning will be used for action agents to achieve better control quality. For energy efficiency-oriented regulation policies, low resolution sensing agents will be chosen, and a database agent will be used to enrich the information content of the sparse measurements; more computations will be performed within each agent to reduce communication throughput; more planning activity within the decision agents will be performed to improve the efficiency of action agents.

Table 3. Regulation policies and agent interaction protocol designs. QoS, quality of service.

Regulation Policy Priority	**Interaction Protocol**
policy 1: response time	high-speed sensing → database agent → action agent
policy 2: QoS	high-resolution sensing → decision agent → database agent → decision agent → action agent
policy 3: energy efficiency	low-resolution sensing → decision agent → action agent

4.3. Agent Behavior Model and Petri-Net-Based Analysis

Both individual and group behaviors can be modeled as finite-state machines (FSM). A finite state machine consists of a number of states. For an agent, each state represents the status of an agent and is associated with certain functions. The transitions among states are also associated with certain operations. For example, as shown in Figure 7, the database agent has three states: waiting, query, done. When the database agent receives a request from the decision agent, it will change from the first state to the second state, and after the query is done, it will change to the third state and send the result to the decision agent. At the same time, after the decision agent sent a request to the database agent, the decision agent will change to the waiting state. When the decision agent receives a reply from the database agent, it will change to the computing state. Once finished computing, it will change to another state for making the decision. The final decision will be sent to the action agent.

For a group of agents, each state represents the status of agent collaboration. Figure 7 shows a composite FSM model of a multi-agent system consisting of four agents. The FSM models of multi-agent behaviors enable a mathematical analysis of their feasibility and stability. One reason is that FSM is a mathematical model of computation, and it can be used in distributed systems to implement system automation. The other reason is that we proposed to use Petri-net as the mathematical evaluation tool to analyze and test our multi-agent model. Moreover, the Petri-net is generated based on the FSM in our design.

The biggest challenge for agent collaboration is resolving possible conflicts in scheduling and resource allocation. Petri-net (PN) models have emerged as very promising performance modeling tools for systems that exhibit concurrency, synchronization and randomness. Petri-nets have been used as one of the mathematical models to describe the execution process of distributed systems. In this work, we utilize Petri-net approaches to study the reachability, consistency of the multi-agent system and evaluate its group behavior.

Figure 7. The architecture of a multi-agent system and the finite states of each agent.

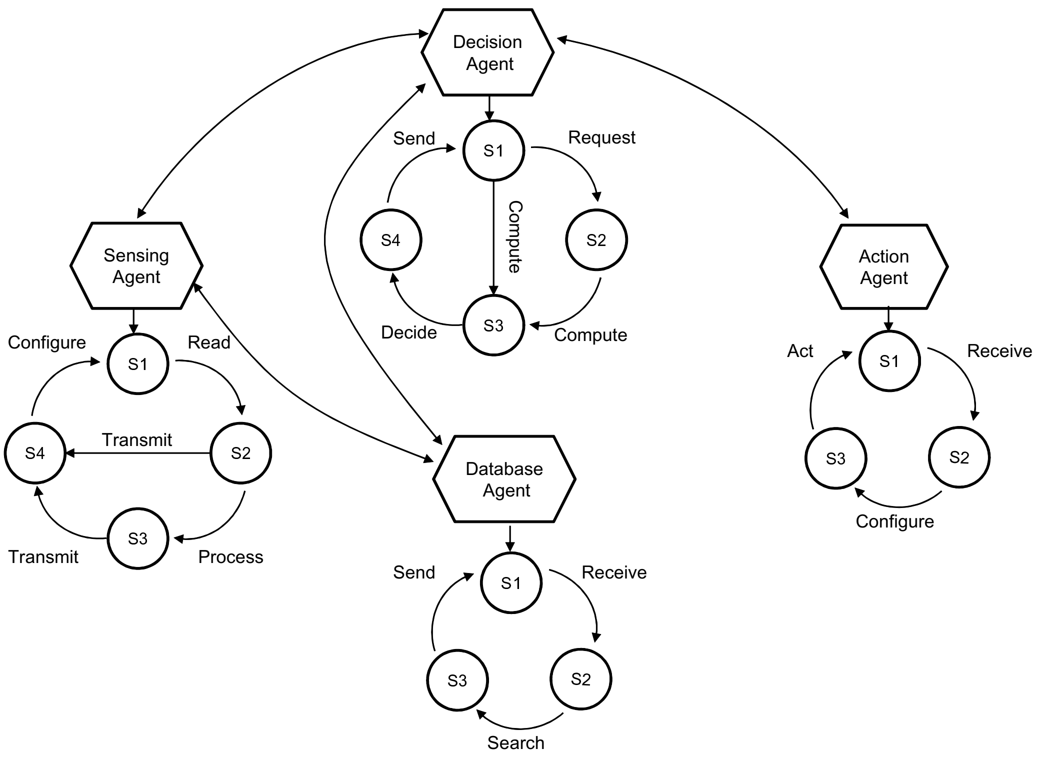


Figure 8 shows Petri-net graphs of three collaborative agents. Each PN consists of positions, transitions and input and output functions. A PN is said to be safe for an initial state if all states are reachable. Figure 8a shows a reachable PN model. The states of a PN evolve by the firing of transitions. A transition is alive for an initial state if there exists a firing sequence for that initial state to reach the next state. Figure 8b shows a state reachability graph of the valid model in (a). When certain transitions are no longer available or when all or part of the PN no longer functions, there will be mistakes in the system design. Figure 8c shows an unreachable PN model. A PN is alive for an initial state if all transitions are live for that initial state. A deadlock is a state in which no transition can be fired. Liveness of a PN implies the degree of absence of potential deadlock states. Based on these concepts, the feasibility and performance of multi-agent collaboration can be evaluated [34].

To check the security and reachability of a multi-agent collaboration scheme using Petri-net methods, we define a PN as a four-tuple combination, $\{P, T, IN, OUT\}$, where:

$$P = \{P_1,\ P_2,\ P_3, \cdots P_n\} \tag{1}$$

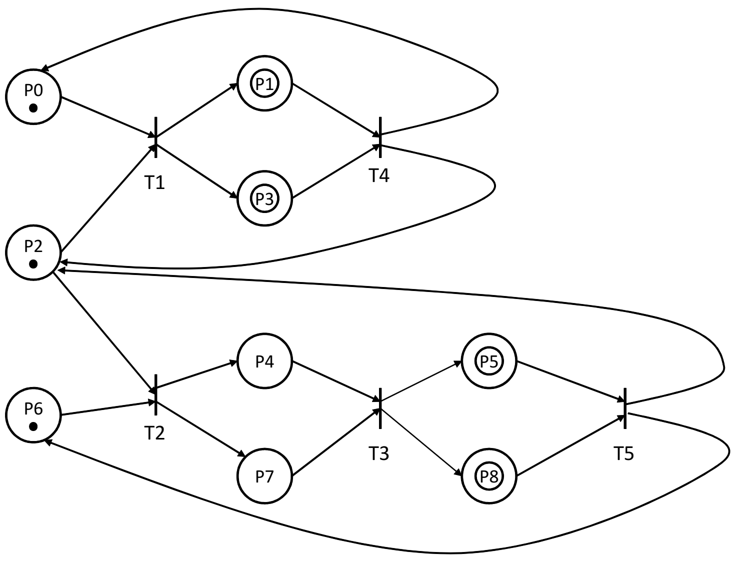
is a set of states, and:

$$T = \{T_1,\ T_2,\ T_3, \cdots T_n\} \tag{2}$$

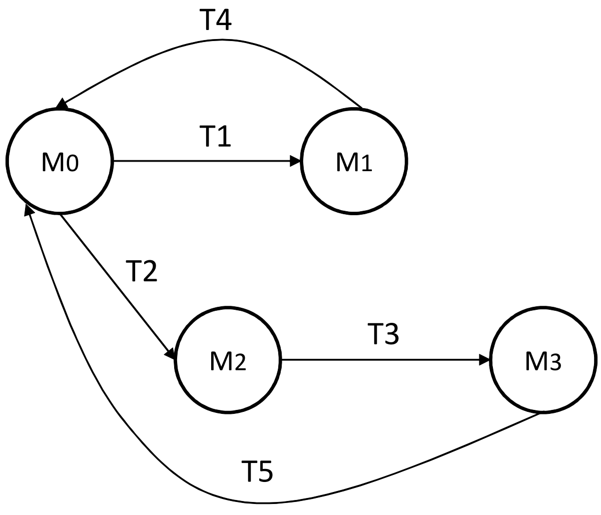
is a set of transitions subject to:

$$P \cup T \neq 0, P \cap T = 0 \tag{3}$$

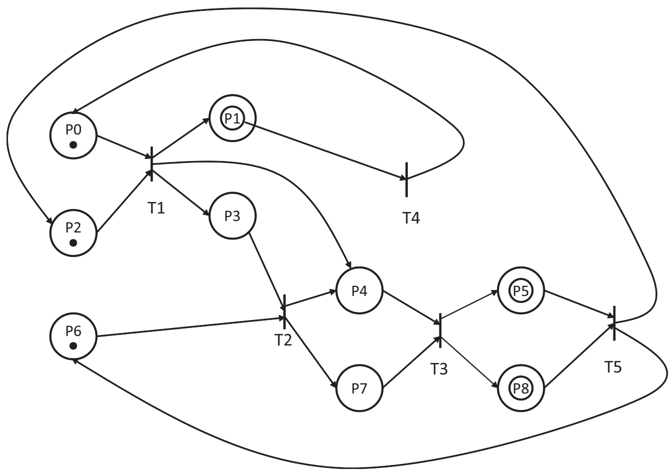
Figure 8. Collaboration scheme design using a Petri-net (PN) graph for three agents. (**a**) a reachable PN model; (**b**) the state reachability graph of a valid collaboration model; (**c**) an unreachable PN model.



(**a**)



(**b**)



(**c**)

Assume IN is an input function that defines directed arcs from states to transitions, and OUT is an output function that defines directed arcs from transitions to states. The process of testing security and reachability of our proposed collaboration model is demonstrated by the following algorithm:

Algorithm 1: Security and reachability test algorithm.

```
Input: positions: $P$, transitions: $T$, position to transition function: $IN$, transition to position
       function: $OUT$
Output: state transition matrix: $M$
$i \leftarrow 0$;
$M_0 \leftarrow initial\ state$;
$W \leftarrow OUT - IN$;
$s \leftarrow sizeof(T)$;
while $i < s$ do
    $i \leftarrow i + 1$;
    $M_i \leftarrow M_0 + T_i * W$;
    if the value of each element in $M_i$ is not larger than 1 then
        $M_i\ is\ reachable$;
        $continue$;
    else
        $M_i\ is\ unreachable$;
        $break$;
    end
end
return $M$
```

Example I: We use an example to verify the safety and reachability of multi-agent collaboration schemes. For the model shown in Figure 8a, the input function is given by:

$$IN = \begin{bmatrix} T \setminus P & P_0 & P_1 & P_2 & P_3 & P_4 & P_5 & P_6 & P_7 & P_8 \\ T_1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ T_2 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ T_3 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ T_4 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ T_5 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

The output function is given by:

$$OUT = \begin{bmatrix} T \setminus P & P_0 & P_1 & P_2 & P_3 & P_4 & P_5 & P_6 & P_7 & P_8 \\ T_1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ T_2 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ T_3 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ T_4 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ T_5 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

According to the definition, an incidence matrix is given by:

$$W = OUT - IN \tag{4}$$

We then obtain:

$$W = \begin{bmatrix} T\setminus P & P_0 & P_1 & P_2 & P_3 & P_4 & P_5 & P_6 & P_7 & P_8 \\ T_1 & -1 & 1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 \\ T_2 & 0 & 0 & -1 & 0 & 1 & 0 & -1 & 1 & 0 \\ T_3 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & -1 & 1 \\ T_4 & 1 & -1 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ T_5 & 0 & 0 & 1 & 0 & 0 & -1 & 1 & 0 & -1 \end{bmatrix}$$

The incidence matrix is used to find out the changes in a Petri-net upon firing a given transition. The characteristic equation of a state transition is given by:

$$M_i = M_0 + T_i \times W \tag{5}$$

where M_0 is the initial state of the state transition equation.

For this collaboration model, the initial state, M_0, should be:

$$M_0 = \begin{bmatrix} P_0 & P_1 & P_2 & P_3 & P_4 & P_5 & P_6 & P_7 & P_8 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

The points in P_0, P_4, P_6 are tokens, which represent the current state of the agent. We choose to fire transition 1; we can get the result of the next state:

$$M_1 = \begin{bmatrix} P_0 & P_1 & P_2 & P_3 & P_4 & P_5 & P_6 & P_7 & P_8 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

We then continue to trigger all the transitions along the path shown in Figure 8. The transition matrix is obtained as:

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

The complete state transition matrix will be:

$$M = \begin{bmatrix} P_0 & P_1 & P_2 & P_3 & P_4 & P_5 & P_6 & P_7 & P_8 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

It can be seen that in the state transition matrix, the number of tokens in any state is not larger than one. This means that there are no conflicts. In other words, the proposed collaboration model is safe and reachable. The state reachability graph is shown in Figure 8b.

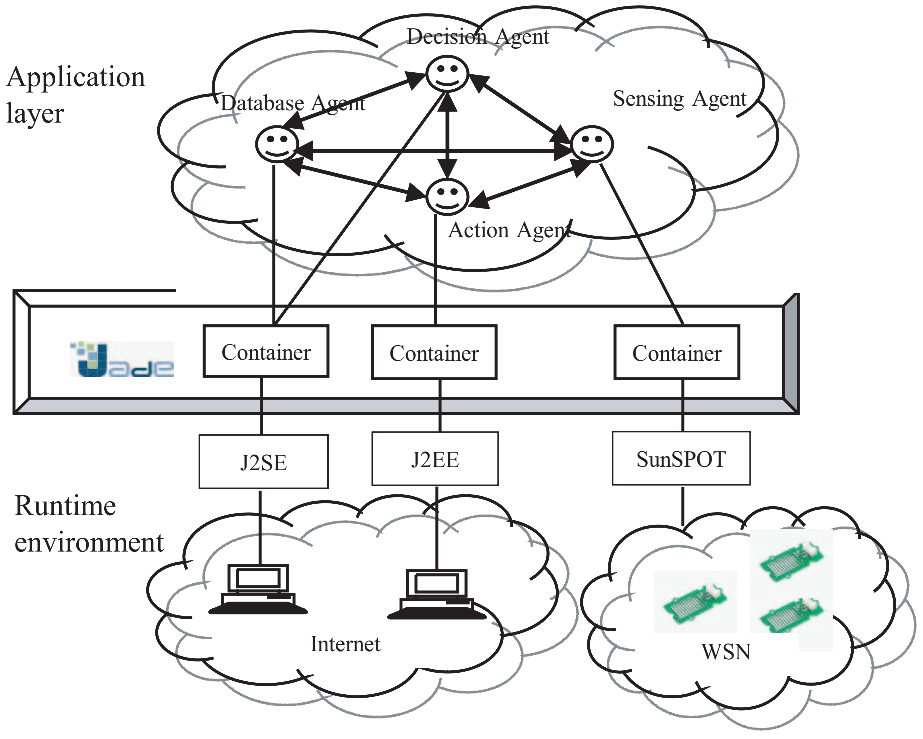
Example II: In contrast, for the PN model shown in Figure 8c, the incidence matrix, W, is:

$$W = \begin{bmatrix} T \setminus P & P_0 & P_1 & P_2 & P_3 & P_4 & P_5 & P_6 & P_7 & P_8 \\ T_1 & -1 & 1 & -1 & 1 & 1 & 0 & 0 & 0 & 0 \\ T_2 & 0 & 0 & 0 & -1 & 1 & 0 & -1 & 1 & 0 \\ T_3 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & -1 & 1 \\ T_4 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ T_5 & 0 & 0 & 1 & 0 & 0 & -1 & 1 & 0 & -1 \end{bmatrix}$$

The transition matrix, T, is the same, and the state matrix is:

$$M = \begin{bmatrix} P_0 & P_1 & P_2 & P_3 & P_4 & P_5 & P_6 & P_7 & P_8 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 2 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \end{bmatrix}$$

Figure 9. Java Agent Development Environment (JADE) multi-agent implementation.



The Petri-net analysis method is based on the FSM model. In an FSM, any agent only can stay in one state at each moment. When the state transition matrix of Figure 8c has a value of two, this means that at a certain moment, an agent needs to stay in two states. Obviously, such a status is hardly reachable during a collaboration. According to [35], such a state transition matrix does not satisfy the safeness metric. Therefore, there is a conflict in the collaboration design shown in Figure 8c, that is, such a collaboration is neither safe nor reachable.

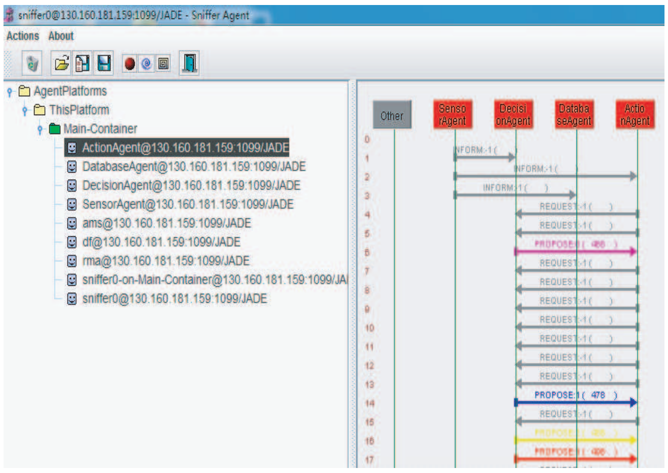
5. JADE Implementation

The proposed multi-agent system is implemented using the Java Agent Development Environment (JADE). JADE is a Java-based open source software framework for developing multi-agent systems. The JADE architecture is built on peer-to-peer modality. Intelligence, initiative, information, resources and control can be fully distributed across a group of heterogeneous hosts, including mobile terminals and devices, through wireless or wired networks. Each agent can communicate and negotiate with its peers to reach mutually acceptable agreements for cooperative problem solving.

5.1. JADE Framework

In our system implementation, we have chosen a three-layer architecture, as shown in Figure 9. The lowest layer is the runtime environment, such as j2se/j2ee on local computing devices or Squawk VM on SunSPOT mobile sensor nodes. The middle layer is the JADE platform, which consists of a number of containers, which provide services for multi-agent operations. The upper one is the application layer in which agents (sensing, decision, database and action) perform collaborations to accomplish required tasks. Figure 10 shows a snapshot of the JADE IDE.

Figure 10. Snapshot of the JADE development environment.


sniffer0@130.160.181.159:1099/JADE - Sniffer Agent
Actions
About
AgentPlatforms
ThisPlatform
Main-Container
ActionAgent@130.160.181.159:1099/JADE
DatabaseAgent@130.160.181.159:1099/JADE
DecisionAgent@130.160.181.159:1099/JADE
SensorAgent@130.160.181.159:1099/JADE
ams@130.160.181.159:1099/JADE
df@130.160.181.159:1099/JADE
rma@130.160.181.159:1099/JADE
sniffer0-on-Main-Container@130.160.181.159:1099/JA
sniffer0@130.160.181.159:1099/JADE
Other
SensorAgent
DecisionAgent
DatabaseAgent
ActionAgent
INFORM:-1 ( )
INFORM:-1 ( )
INFORM:-1 ( )
REQUEST:-1 ( )
REQUEST:-1 ( )
PROPOSE:1 (486)
REQUEST:-1 ( )
REQUEST:-1 ( )
REQUEST:-1 ( )
REQUEST:-1 ( )
REQUEST:-1 ( )
REQUEST:-1 ( )
REQUEST:-1 ( )
PROPOSE:1 (478)
REQUEST:-1 ( )
PROPOSE:1 (486)
PROPOSE:1 (486)

5.2. Multi-Agent Implementation

Each JADE application consists of a set of agents with unique names and IDs. Agents execute tasks and interact with each other by exchanging messages and beliefs. Agents run on a platform that provides services, such as message delivery, agent migration and resource management. A platform is composed of one or multiple containers, which can be implemented among distributed hosts, such as mobile devices, the base station and sensor/actuator nodes. Each container can host multiple agents. A typical agent program has three components: (1) behavior; (2) communication protocol; and (3) graphical user interface (GUI).

Each agent has a set of active behaviors; each behavior should achieve a single task or sub-task. The behavior can be defined as one-shot, cyclical or conditional. For each behavior, the execution method needs to be implemented. Multiple behaviors in each agent have to be scheduled properly. Each communication protocol contains four components: (1) definition of the message structure; (2) discovery of agents for communication; (3) message sending and reception mechanism; and (4) message filtering function. Each agent GUI has three components: (1) a standard graphical user interface for user interaction; (2) a jFrame interface containing visual components (e.g., textbox, button, checkbox); (3) a set of events, events handlers and event listeners.

In our study, there are four types of agents: sensing, action, decision and database. The detailed behavior of each agent can be summarized as the following algorithms. Algorithm 2 gives an example of a simple behavior for sensing agents. n sensor nodes can fuse their sensory data, based on context information, to form a set of scenario data. Algorithm 3 gives an example of a simple behavior for decision agents. The input for the decision agents is scenario data from sensing agents. Decision agents will interact with database agents to access the knowledge database and inference rules.

Algorithm 2: Sensing agent behavior.

```
Input: number of sensing agents: $n$; context message: $m$
Output: scenario data (a data set of human subjects, light, temperature, etc.): $s$
$i \leftarrow 1$;
while $i \leq n$ do
    $check\ state$;
    if $n_i$ $is\ not\ suspended$ then
        $receive\ m$;
        $s \leftarrow observation\ signal$;
    else
        $check\ next\ one$;
    end
    $i \leftarrow i + 1$;
end
return $s$
```

Algorithm 4 gives an example of a simple behavior for database agents. The input for database agents is a request message. Database agents can provide inference rules based on context variables. Algorithm 5 gives an example of a simple behavior for action agents. Action agents perform some actions according to commands from decision agents. The detailed behaviors of these agents under different policies are also provided in the FSMs shown in Figures 11–13.

Algorithm 3: Decision agent behavior.

```
Input: scenario data from all working sensing agents: $s$; number of sensing agents: $n$
Output: decision
1  $data \leftarrow 0$; $\triangleright$ $the\ fusion\ of\ scenario\ data$
2  for $i \leftarrow 1$ to $n$ do
3      $data \leftarrow data + s_i$
4  end
5  $process\ data$;
6  $refer\ decision\ rules : dr$;
7  if $dr == null$ then
8      $create\ request\ message :\ rMessage$;
9      $send\ rMessage\ to\ database\ agent$;
10     $receive\ response$;
11     $make\ decision$;
12 else
13     $make\ decision$
14 end
15 $send\ decision\ to\ action\ agent$;
```

Algorithm 4: Database agent behavior.

```
Input: request message from decision agent: $rMessage$
Output: reference: $r$
1  while $rMessage$ do
2      $search\ related\ rules$;
3      if rules found then
4          $r \leftarrow rules$
5      else
6          $r \leftarrow null$
7      end
8  end
9  return $r$
```

Algorithm 5: Action agent behavior.

```
Input: decision command from decision agent: $cmd$
Output: feedback message: $fMessage$
1  while $cmd$ do
2      $act$;
3      $generatefMessage$;
4      if act then
5          $fMessage \leftarrow success$
6      else
7          $fMessage \leftarrow failure$
8      end
9  end
10 return fMessage
```

Figure 11. The finite state machines for each agent under the response-time-oriented policy.

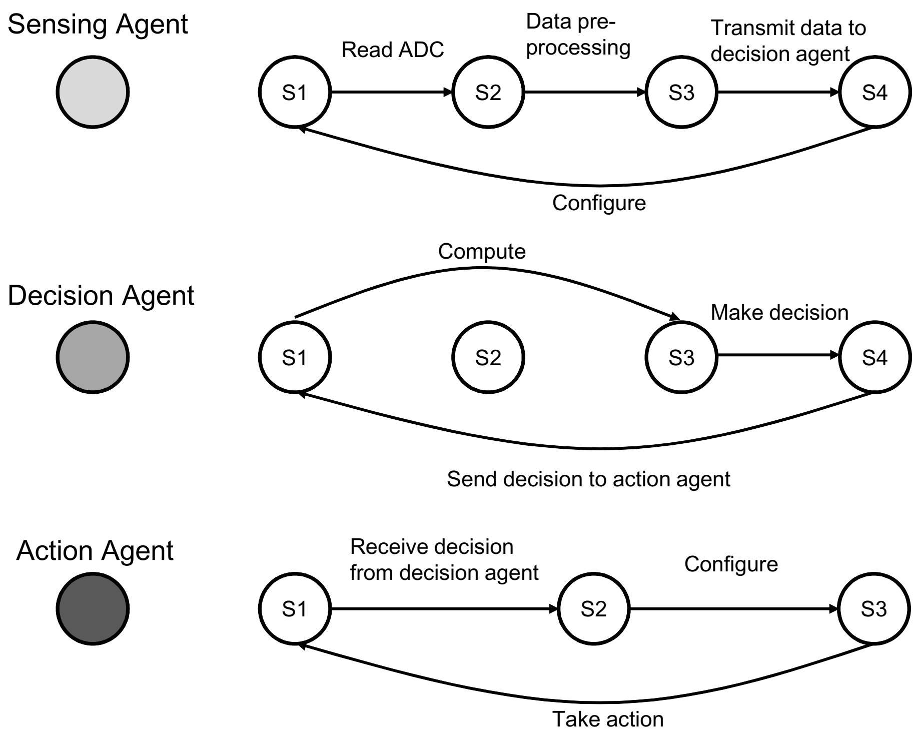


Figure 12. The finite state machines for each agent under the energy-efficient-oriented policy.

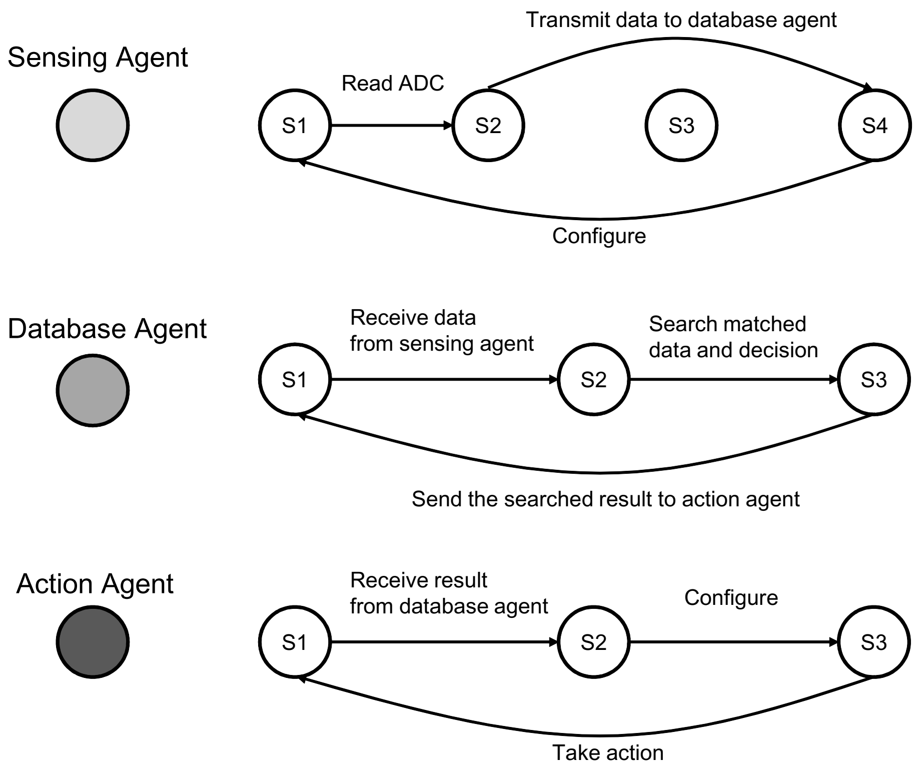
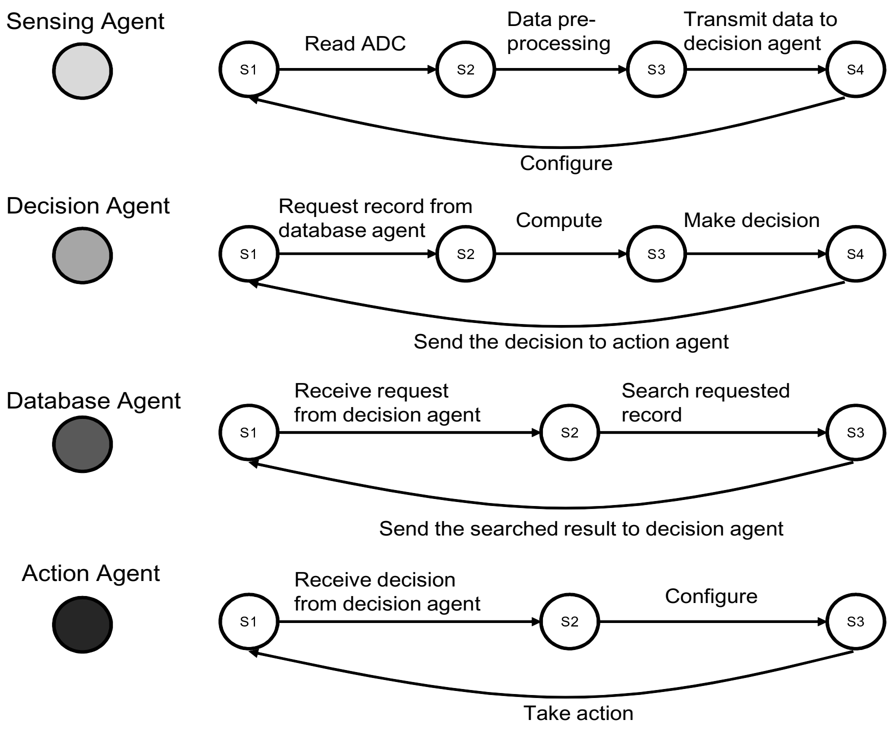

Sensing Agent
Transmit data to database agent
Read ADC
S1
S2
S3
S4
Configure
Database Agent
Receive data from sensing agent
Search matched data and decision
S1
S2
S3
Send the searched result to action agent
Action Agent
Receive result from database agent
Configure
S1
S2
S3
Take action

Figure 13. The finite state machines for each agent under the QoS oriented policy.


Sensing Agent
Read ADC
Data pre-processing
Transmit data to decision agent
Configure
S1
S2
S3
S4
Decision Agent
Request record from database agent
Compute
Make decision
Send the decision to action agent
Database Agent
Receive request from decision agent
Search requested record
Send the searched result to decision agent
Action Agent
Receive decision from decision agent
Configure
Take action

6. Evaluation Metrics

A number of metrics can be used for performance evaluation of multi-agent systems. These evaluation metrics can fall into two categories: logic and quantitative evaluation metrics. Logic evaluation metrics validate logic functions and conflicts of a multi-agent system. Quantitative evaluation metrics measure the quality of service, power consumption, computing complexity and communication throughput of the system.

6.1. Logic Evaluation Metrics

We develop logic evaluation metrics based on Petri-net models. For a Petri-net, reachability, boundedness and liveness are common metrics for evaluation. Reachability refers to all Petri-net states having the ability to be transformed from an initial state through a sequence of transition firing. Boundedness refers to the possibility of all the states being reached in a finite sequence. Liveness refers to the degree of absence of deadlocks. *Multi-agent collaboration is based on hierarchical finite state machines. Given environmental and system uncertainties, deadlocks and large bounds are inevitable in collaboration schemes. Besides, the evaluation of liveness and boundedness of a Petri-net involves higher computational complexity.* Therefore, in this paper, we only choose state reachability (SR) as the logic metric to validate a MAS design.

6.2. Quantitative Evaluation Metrics

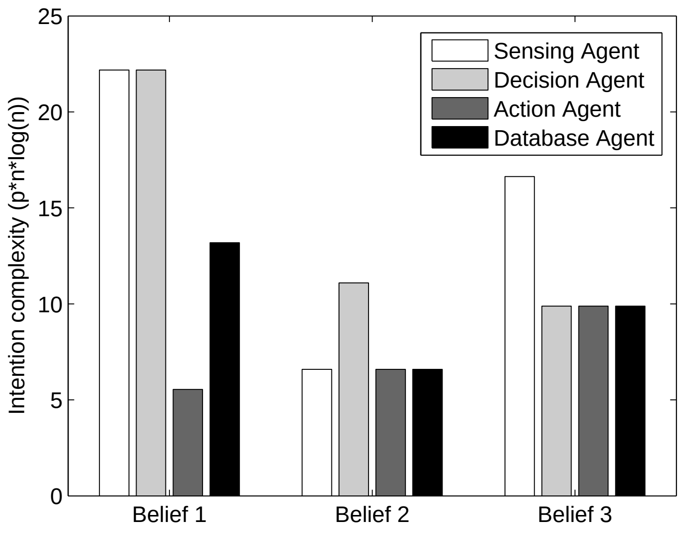
Quantitative evaluation metrics can be used to measure the QoS of a multi-agent system, as well as the resource consumption. In the context of MAS, QoS refers to a ratio between real performance and ideal performance of the system. It is a real number between zero and one. Resource consumption indexes include communication throughput, memory occupancy, computation complexity and power consumption. We can use above indexes normalized by the power consumption to evaluate the effectiveness of a multi-agent system.

7. Results and Discussions

7.1. Individual Agent Behavior Evaluation

The individual agent behavior design is based on BDI models. For a given user goal and existing beliefs, a set of feasible desires and intentions (*i.e.*, action plan) will be generated to select suitable agent behaviors. Low complexity intentions correspond to simple agent behaviors (e.g., low sensing resolution, low complexity estimation and control algorithms). High complexity intentions correspond to complicated agent behaviors (e.g., interaction with database agents for estimation, control and decision). We use the computational complexity of intentions to represent the overall intention complexity. Figure 14 illustrates the behavior design result.

Figure 14. BDI models for four types of agents.

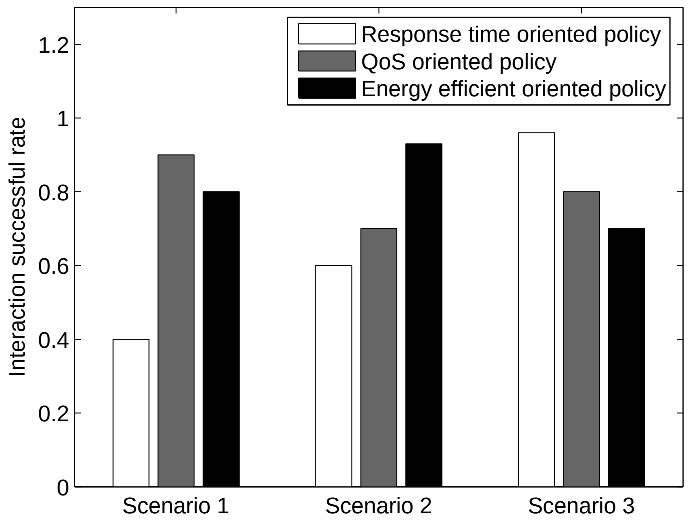


Generally, in a BDI model-based multi-agent system, beliefs include an understanding of the condition of agents, the situation of the environment and the demands of users. In order to reduce the cost of the whole sensing system and to give an intuitive description of agent behaviors, in this part, we simplify the beliefs to be only about the illumination condition in a room (the situation of the environment). In Figure 14, belief1, belief2 and belief3 represent varying, high and low light intensity, respectively. The coefficient, p, represents the computation complexity for one operation, the coefficient, n, stands for the size of agent states. The coefficient, n, can be obtained from the specific agent FSM design. The factor, $\log n$, which represents a certain degree of parallelism in the computing, is introduced in this paper to calculate the intention complexity motivated by the complexity analysis of multi-agent system [36].

7.2. Multi-Agent Group Behavior Evaluation

Multi-agent group behavior design is based on a regulation policy and Petri-net analysis. For a given user's regulation policy of tasks and resources, a set of agent collaboration protocols is generated based on individual agent behaviors and resource conditions. Petri-net analysis methods are then used to validate and analyze candidates for group behaviors. For each valid group behavior, quality of service performance and related computing/communication/energy costs are used to select the final group collaboration protocol. Figure 15 shows the successful rate of multi-agent collaboration for three regulation policies (response-time-oriented, QoS-oriented, and energy-efficiency-oriented) under different scenarios (complex, medium and simple).

Figure 15. Performance of multi-agent collaborations under three different policies.



In Figure 15, scenario 1, scenario 2 and scenario 3 correspond to complex, medium and simple scenarios, respectively. More specifically, the complex scenario includes the light intensity of the environment, the power condition of agents and the demands from the user. The medium scenario includes the light intensity of the environment and the power condition of agents. The simple scenario includes the user demands only. Because the complex scenario contains three constraints, the agents need to perform more analysis to meet the demands. Obviously, the high-speed sensing and low-resolution sensing operation (defined in Table 3), under the response-time-oriented and energy-efficiency-oriented policies, will lead to poorer analysis/decision results than the QoS-oriented policy does. Therefore, the QoS-oriented policy has the highest successful interaction rate. Likewise, for the simple scenario, only the fast response requirement is considered. Under the imposed time constraint, obviously the response-time-oriented policy will yield the highest successful interaction rate.

7.3. System Performance Evaluation

We have investigated three system designs to test the proposed multi-agent-based sensor-actuator system for the house illumination control problem. The design priorities are (1) response time, (2) quality of service (QoS) and (3) power consumption. As shown in Table 3, the first design utilizes high-speed sensing, simple decision algorithms, large control signals and the least database information to achieve fast system response to the changing environment and situations. The second design utilizes high-resolution sensing, complicated decision algorithms, large control signals and a lot of database information to achieve a QoS to meet users' expectation. The third design utilizes low-resolution sensing, complicated decision algorithms, small control signals and a large amount of contextual and situational information from the database to minimize power consumption. The detailed operation procedure of these designed policies and the FSMs are given by Figures 11–13.

Figure 16 shows changing environment conditions, number of users and available energy storage, which is supplied by energy harvesting devices. Figure 17 shows a comparison of QoS among three regulation policies. Figures 18–20 illustrate the histograms of the QoS performance under three policies given high-power and low-power storage levels, correspondingly. It can be seen that: (1) the energy-efficiency-oriented policy yields unstable QoS and, sometimes, makes wrong decisions; (2) when the power storage is sufficient, there is not much difference in the QoS performance between the response-time-oriented policy and the quality of service-oriented policy; (3) when the power storage level is low, there is not much difference in the expected QoS performance between the response-time-oriented policy and the energy-efficiency-oriented policy, but the energy-efficiency-oriented policy's performance yields the largest deviation.

Figure 16. Environment, user and system information: (**first row**) light intensity; (**second row**) number of humans; (**third row**) power condition.

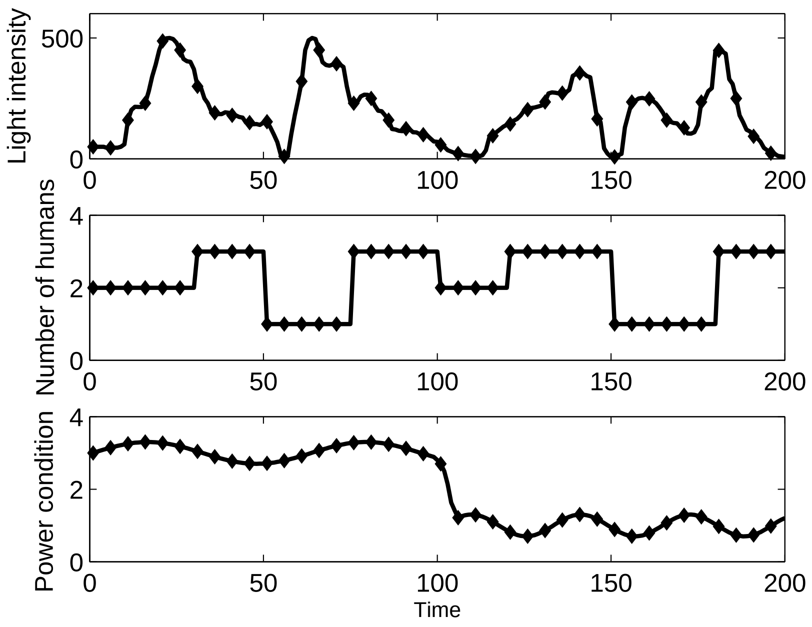


Figure 17. Comparison of the quality of service (QoS).

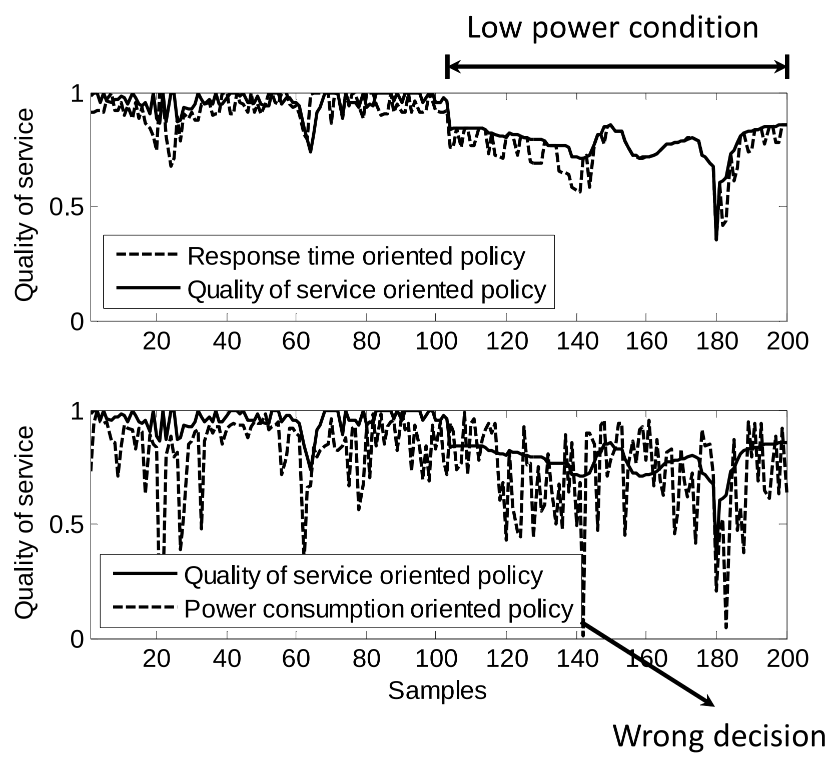


Figure 18. Histogram of the QoS performance under the response-time-oriented policy for (**left**) high and (**right**) low power storage levels.

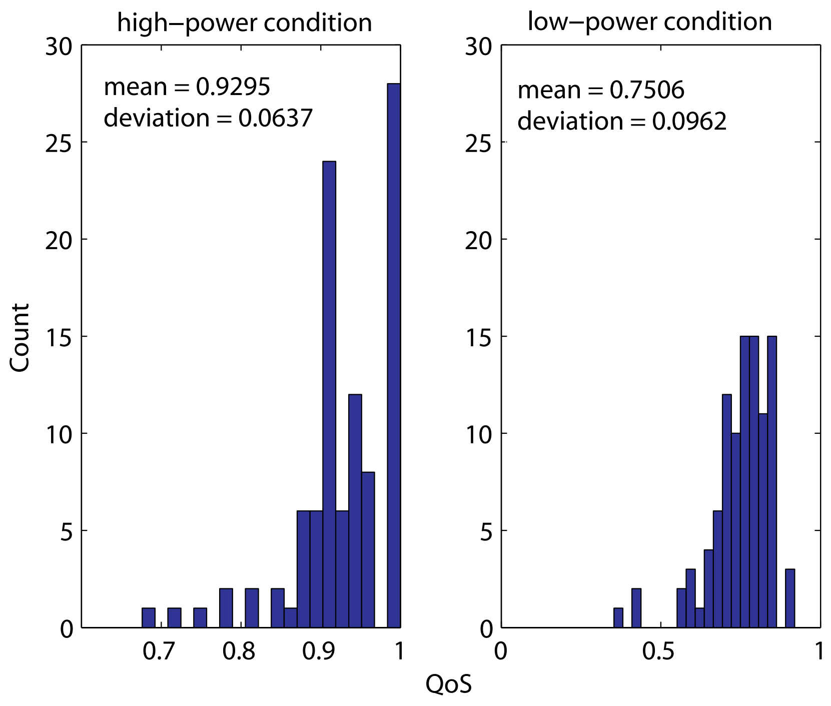


Figure 19. Histogram of the QoS performance under the quality of service-oriented policy for (**left**) high and (**right**) low power storage levels.

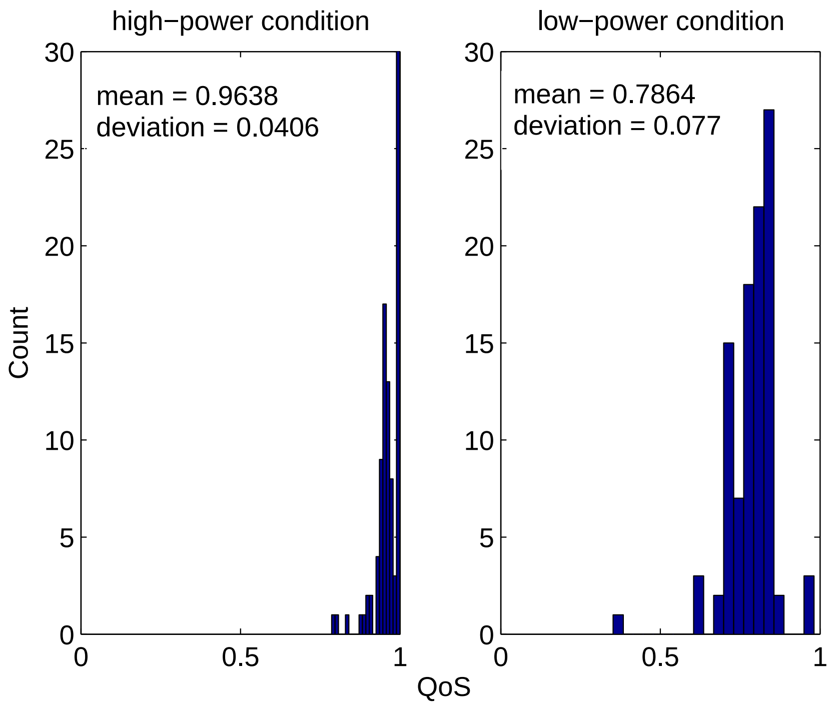


Figure 20. Histogram of the QoS performance under the energy-efficiency-oriented policy for (**left**) high and (**right**) low power storage levels.

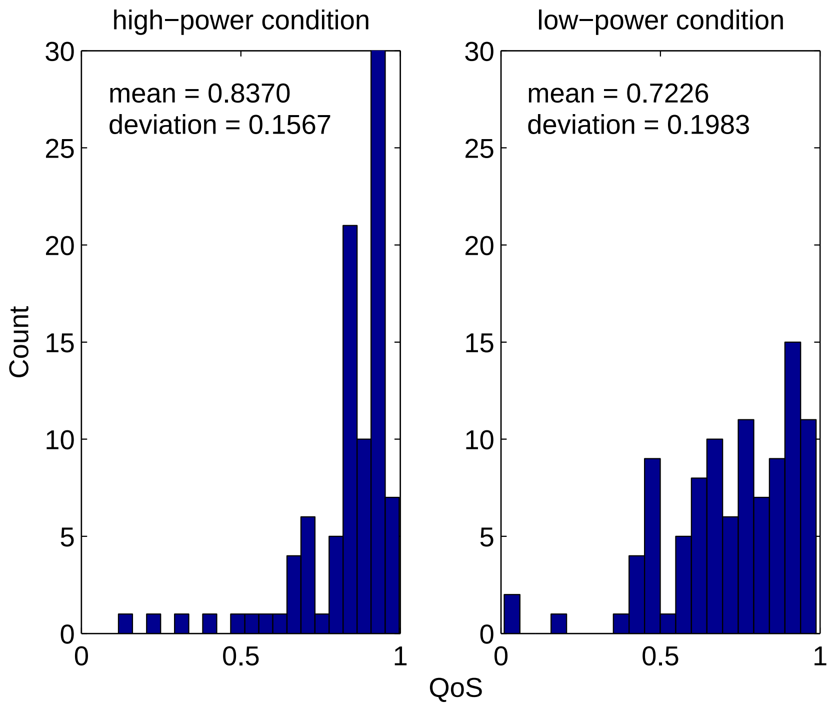


Figure 21 shows the comparison of computational time for these three system designs. Figures 22–24 illustrate the histograms of the computation time under three policies given high-power and low-power storage levels, correspondingly. It can be seen that (1) QoS-oriented policies lead to the highest computational costs, because they involve the use of databases and complicated decision algorithms; (2) there is not much difference in computational time under response-time-oriented policies for both power storage levels; (3) when the power storage level is low, all the policies will reduce both the data volume and algorithm complexity, resulting in lower computation costs.

Figure 21. Comparison of computation (response) time.

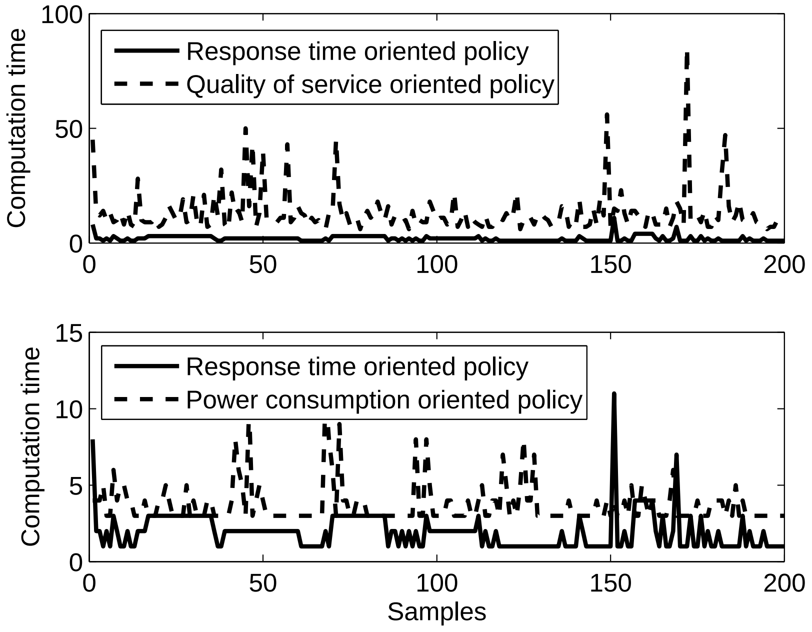


Figure 22. Histogram of the computation time under the response-time-oriented policy for (**left**) high and (**right**) low power storage levels.

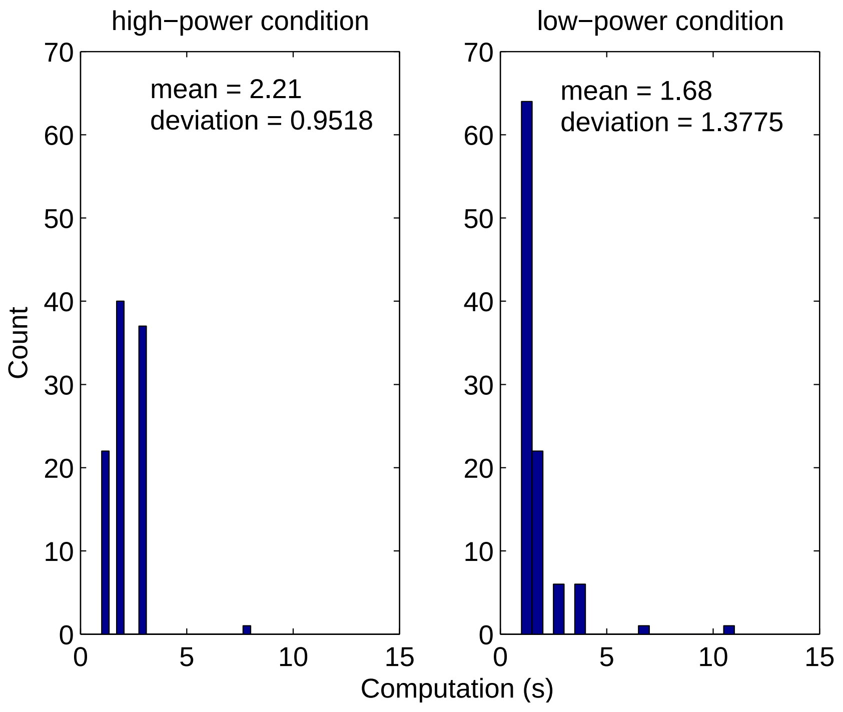


Figure 23. Histogram of the computation time under the quality of service-oriented policy for (**left**) high and (**right**) low power storage levels.

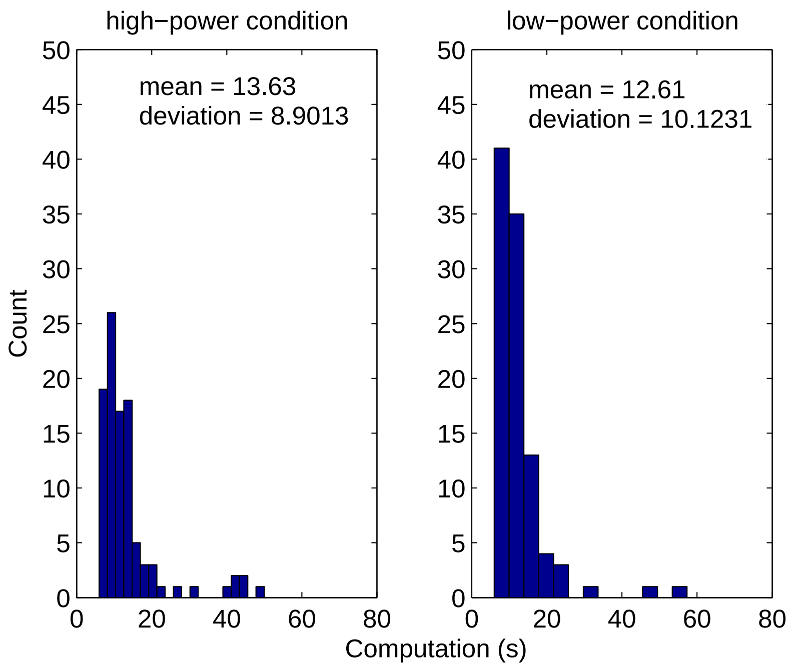
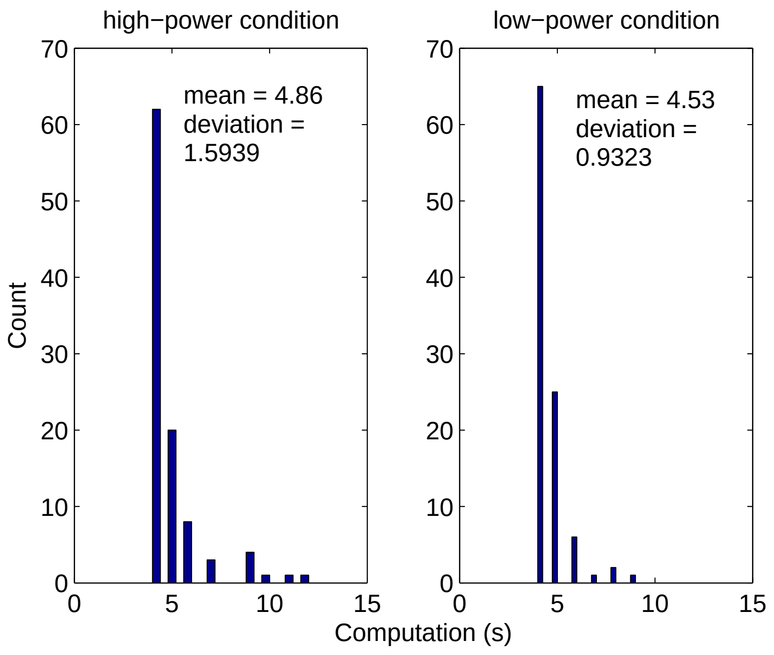


Figure 24. Histogram of the computation time under the energy-efficiency-oriented policy for (**left**) high and (**right**) low power storage levels.



If we have a weighted sum of computation complexity, sensing usage, control usage and communication throughput, we can roughly estimate the system cost, which is equivalent to the power consumption, for each system. Figure 25 shows a comparison of the system cost for the three systems. The normalized system performance can be calculated as the performance-to-system cost ratio. Then, we can use the normalized system performance to evaluate three system designs, as shown in

Figure 26. It can be seen that: (1) the QoS-oriented system provides the best service, performs the most computations, has the highest communication throughput and yields the least decision errors; (2) the power consumption-oriented system yields the most decision errors and provides the poorest service. In real applications, these three system operation modes can be interchanged to adapt to changing situations.

Figure 25. Comparison of system cost.

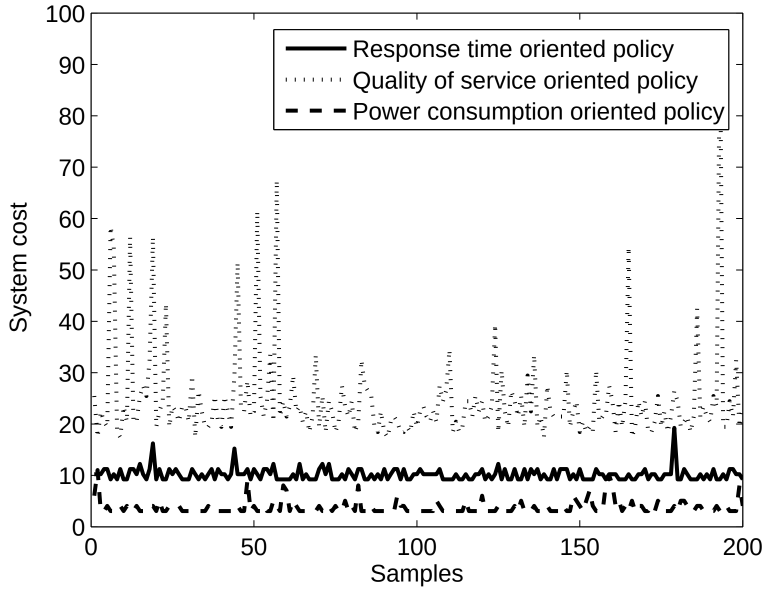
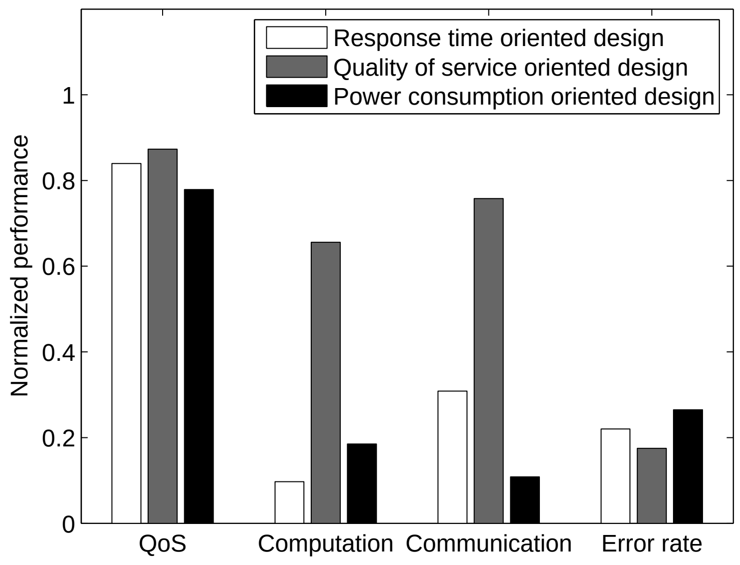


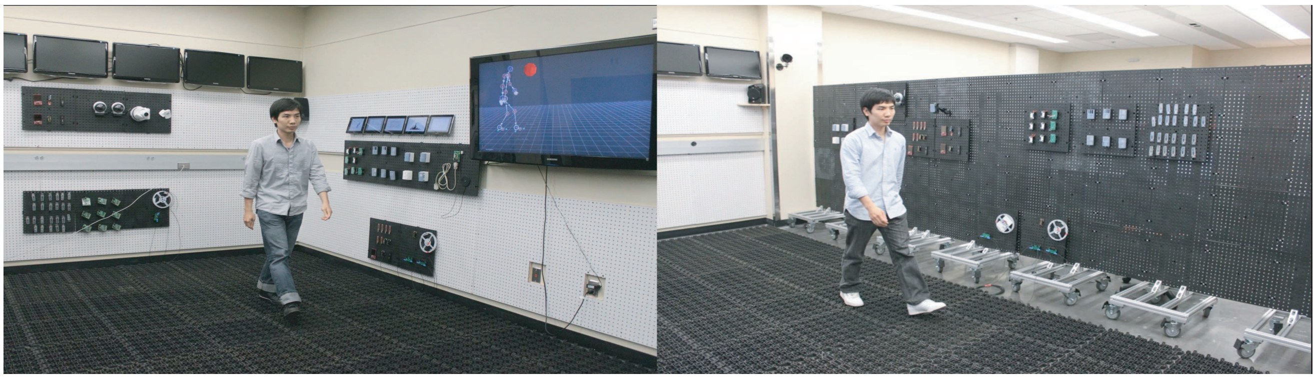
Figure 26. Performance of multi-agent systems under three different policies.



7.4. Testbed Setup and Implementation Plan

Figure 27 illustrates the experimental setup of a multi-agent system testbed for smart house applications. The testbed consists of thermal, pressure, laser and photonic sensor arrays based on reconfigurable hardware platforms. With this testbed, human subject activities can be measured through a multi-agent distributed sensor network. We have chosen SunSPOT as the wireless sensor network platform. Each SPOTmote includes a CPU, memory and a wireless radio. SunSPOTs run a native small-footprint Java virtual machine that can host multiple applications concurrently. SunSPOTs offer a fully capable Java ME (JME) environment that supports CLDC1.1, and MIDP1.0. stackable boards can be used to expand the abilities of a SPOT and can include application-specific sensors and actuators. Our future work includes the development of a multi-agent-based smart home sensor-actuator network on the SunSPOT platform by using an improved version of JADE-LEAP.

Figure 27. A testbed for MAS-based smart house technology.



8. Conclusions

In this paper, we have presented a multi-agent design framework for smart house and home automation applications. A set of techniques have been proposed to develop multi-agent-based distributed sensor/actuator networks. A BDI model is developed for agent individual behavior design. A regulation policy-based method is developed for multi-agent group behavior design. A Petri-net based method is developed for system evaluation and analysis. A testbed has been constructed for further experimental verification. The initial results have demonstrated that the proposed methods can be used for multi-agent system design and performance evaluation. Our future work includes implementing the multi-agent software in the developed sensor/actuator network and performing extensive experiments on the testbed.

Acknowledgements

Partial support of this research was provided by the U.S. National Science Foundation (NSF) CNS-1059212. All results presented do not necessarily reflect the opinions of NSF.

Conflict of Interest

The authors declare no conflict of interest.

References

1. Wang, M.; Wang, H. Intelligent agent supported flexible workflow monitoring system. *Adv. Inf. Syst. Eng.* **2006**, 787–791.

2. Olfati-Saber, R.; Fax, J.A.; Murray, R.M. Consensus and cooperation in networked multi-agent systems. *Proc. IEEE* **2007**, *95*, 215–233.
3. Qela, B.; Mouftah, H.T. Observe, Learn, and Adapt (OLA)—An Algorithm for Energy Management in Smart Homes Using Wireless Sensors and Artificial Intelligence. *IEEE Trans. Smart Grid* **2012**, *3*, 2262–2272.
4. Duong, T.V.; Phung, D.Q.; Bui, H.H.; Venkatesh, S. Efficient Coxian Duration Modelling for Activity Recognition in Smart Environments with the Hidden Semi-Markov Model. In Proceedings of 2005 Intelligent Sensors, Sensor Networks and Information Processing Conference, Melbourne, Australia, 5–8 December 2005; pp. 2262–2272.
5. Gaddam, A.; Mukhopadhyay, S.C.; Gupta, G.S. Trial & Experimentation of a Smart Home Monitoring System for Elderly. In Proceedings of IEEE Instrumentation and Measurement Technology Conference (I2MTC), Hangzhou, China, 10–12 May 2011; pp. 1–6.
6. Sun, Q.; Wu P.; Wu Y. Unsupervised Multi-Level Non-Negative Matrix Factorization Model: Binary Data Case. *J. Information Security* **2012**, *3*, 245–250.
7. Gaddam, A.; Mukhopadhyay, S.C.; Gupta, G.S. Elder care based on cognitive sensor network. *IEEE Sens. J.* **2011**, *11*, 574–581.
8. Stefanov, D.H; Bien, Z.; Bang, W.-C. The smart house for older persons and persons with physical disabilities: Structure, technology arrangements, and perspectives. *IEEE Trans. Neural Syst. Rehabil. Eng.* **2004**, *12*, 228–250.
9. Sun, Q.; Hu F.; Hao Q. Human Activity Modelling and Situation Perception Based on Fiber-optic Sensing System. *IEEE Trans. Human Mach. Syst.* **2013**, in press.
10. Zhong, D.; Ji, W.; Liu, Y.; Han, J.; Li, S. An improved routing algorithm of Zigbee wireless sensor network for smart home system. In Proceedings of 2011 5th International Conference on Automation, Robotics and Applications (ICARA), Wellington, New Zealand, 6–8 December 2011; pp. 346–350.
11. Tsou, Y.-P.; Hsieh, J.-W.; Lin, C.-T.; Chen, C.-Y. Building a remote supervisory control network system for smart home applications. In Proceedings of 2006 IEEE International Conference on Systems, Man and Cybernetics, SMC'06, Taipei, Taiwai, 8–11 October 2006; Volume 3, pp. 1826–1830.
12. Zhang, L.; Leung, H.; Chan, K. Information fusion based smart home control system and its application. *IEEE Trans. Consum. Electron.* **2008**, *54*, 1157–1165.
13. Madden, S.R.; Franklin, M.J.; Hellerstein, J.M.; Hong, W. TinyDB: An acquisitional query processing system for sensor networks, *ACM Trans. Database Syst.* **2005**, *30*, 122–173.
14. Mueller, R.; Alonso, G.; Kossmann, D. SwissQM: Next generation data processing in sensor networks. *CIDR* **2007**, *7*, 1–9.
15. Fortino, G.; Guerrieri, A.; O'Hare, G.; Ruzzelli, A. A flexible building management framework based on wireless sensor and actuator networks. *J. Netw. Comput. Appl.* **2012**, *35*, 1934–1952.
16. Knishiyama, T.; Sawaragi, T. A decision-making modeling for home ambient intelligent system and its extension to multi-agent modeling. In Proceedings of 2011 IEEE/SICE International Symposium on System Integration (SII), Kyoto, Japan, 20–22 December 2011; pp. 354–357.

17. Cervantes, L.; Lee, Y.-S.; Yang, H.; Ko, S.-H.; Lee, J. Agent-Based Intelligent Decision Support for the Home Healthcare Environment. In *Advances in Hybrid Information Technology*; Springer: Berlin/Heidelberg, Germany, 2007; pp. 414–424.
18. Cook, D.J.; Youngblood, M.; Heierman, E.O. III; Gopalratnam, K.; Rao, S.; Litvin, A.; Khawaja, F. MavHome: An Agent-Based Smart Home. In Proceedings of the First IEEE International Conference on Pervasive Computing and Communications, Fort Worth, TX, USA, 23–26 March 2003; pp. 521–524.
19. Hannon, C.; Burnell, L. A distributed multi-agent framework for intelligent environments. *J. Syst. Cybern. Inform.* **2005**, *3*, 1–6.
20. Jarvis, P.A.; Wolfe, S.R.; Enomoto, F.Y.; Nado, R.A.; Sierhuis, M. A Centralized Multi-Agent Negotiation Approach to Collaborative Air Traffic Resource Management Planning. In Proceedings of the Twenty-Second Conf. on Innovative Applications of Artificial Intelligence, Atlanta, GA, USA, 11–15 July 2010; pp. 1787–1792.
21. Helsinger, A.; Thome, M.; Wright, T. Cougaar: A Scalable, Distributed Multi-agent Architecture. In Proceedings of IEEE International Conference on Systems, Man and Cybernetics, The Hague, The Netherlands, 10–13 October 2004; Volume 2, pp. 1910–1917.
22. Moreno, R.P.; Tardioli, D.; Salcedo, J. Distributed implementation of discrete event control systems based on Petri Nets. In Proceedings of 2008 IEEE International Symposium on Industrial Electronics, Cambridge, UK, 30 June–2 July 2008; pp. 1738–1745.
23. Bai, Q.; Zhang, M.; Zhang, H., A Colored Petri Net Based Strategy for Multi-Agent Scheduling. In Proceedings of Rational, Robust, and Secure Negotiation Mechanisms in Multi-Agent Systems, Amsterdam, The Netherlands, 25 July 2005; pp. 3–10.
24. Kim, Y.W.; Suzuki, T.; Narikiyo, T. FMS scheduling based on timed Petri Net model and reactive graph search. *Appl. Math. Model.* **2007**, *31*, 955–970.
25. Huan, F. Application of property-preservation Petri net operation in Agent BDI inner architecture modeling. *Comput. Eng. Des.* **2009**, *30*, 245–255.
26. Benta, K.-I.; Hoszu, A.; Văcariu, L.; Creţ, O. Agent Based Smart House Platform with Affective Control. In Proceedings of the 2009 Euro American Conference on Telematics and Information Systems: New Opportunities to increase Digital Citizenship, Prague, Czech Republic, 3–5 June 2009; Article No. 18.
27. Pipattanasomporn; M.; Feroze, H.; Rahman S. Multi-Agent Systems in a Distributed Smart Grid: Design and Implementation. In Proceedings of IEEE Power Systems Conference and Exposition, Seattle, WA, USA, 15–18 March 2009; pp. 1–8.
28. Baker, P.C.; Catterson, V.M.; McArthur, S.D.J. Integrating an Agent-Based Wireless Sensor Network within an Existing Multi-Agent Condition Monitoring System. In Proceedings of IEEE International Conference on Intelligent System Applications to Power Systems, Curitiba, Brazil, 8–12 November 2009; pp. 1–6.
29. Wang, K.I.-K.; Abdulla, W.H.; Salcic, Z. Ambient intelligence platform using multi-agent system and mobile ubiquitous hardware. *Pervasive Mob. Comput.* **2009**, *5*, 558–573.
30. Su, C.-J.; Wu, C.-Y. A JADE implemented mobile multi-agent based, distributed information platform for pervasive health care monitoring. *Appl. Soft Comput.* **2011**, *11*, 315–325.

31. Sun, Q.; Hu, F.; Hao, Q. Mobile Targets Region-of-Interest via Distributed Pyroelectric Sensor Network: Towards a Robust, Real-Time Context Reasoning. In Proceedins of IEEE Conference on Sensors, Waikoloa, HI, USA, 1–4 November 2010; pp. 1832–1836.
32. Sun, Q.; Hu, F.; Hao, Q. Context Awareness Emergence for Distributed Binary Pyroelectric Sensors. In Proceedins of 2010 IEEE Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI), Salt Lake City, UT, USA, 5–7 September 2010; pp. 150–155.
33. Sun, Q.; Hu, F.; Hao, Q. Mobile targets scenario recognition via low-cost pyroelectric sensing system: Towards a context-enhanced accurate context identification. *IEEE Trans. Man Syst. Cybern.* **2013**, in press.
34. Hsieh, F.S. Developing cooperation mechanism for multi-agent systems with Petri nets. *Eng. Appl. Artif. Intell.* **2009**, *22*, 616–627.
35. Chauhan, D. JAFMAS: A Java-Based Agent Framework for Multi-Agent Systems Development and Implementation. M.Sc. Thesis, University of Cincinnati, Cincinnati, OH, USA, 1997.
36. Dekhtyar, M.; Dikovsky, A.; Valiev, M. Complexity of multi-agent systems behavior. *Log. Artif. Intell.* **2002**, *2424*, 125–136.