Simulation of a Multi-agent Protocol for Task Allocation in Cooperative Design

Kwang Mong Sim, Simon Chi Keung Shiu, and Martin Bun Lam
Department of Computing,
Hong Kong Polytechnic University,
Hung Hom, Kowloon, Hong Kong.
E-mail: {cskmsim, csckshiu}@comp.polyu.edu.hk, martinlam@usa.net
Fax: (852) 2774-0842

ABSTRACT

To achieve high performance in agent-based cooperative design, the effective allocation of design agents to distributed and cooperative design tasks becomes a crucial issue. This research extends the well-known *contract net protocol* for decentralized task allocation in cooperative engineering design. Since the contract net protocol only provides a generic framework for agents to exchange and evaluate information, it does not prescribe any specific coordination policies for cooperative engineering design. By adding a set of agent selection and task selection policies, this research addresses issues in agent-based cooperative engineering design such as: How an agent is selected to carry out a design task? How can an agent select from among a list of design tasks which to bid for? Through a series colored petri-net simulation experiments, the performance of these selection policies are measured and evaluated. Smith has noted that the contract net protocol is particularly suitable for executing different (sub-)tasks simultaneously while ensuring effective resource allocation and focused decision. Results from the coloured petri-net simulations have shown that these properties are preserved in the proposed protocol.

1. INRODUCTION

The information and expertise required in designing large-scale and complex artifacts are usually distributed among different (teams of) engineers throughout and perhaps outside an enterprise. Projects in large-scale design industries often require the cooperation among multi-disciplinary design teams using incompatible software tools to support different aspects (or phases) of a design. To prevent any impediment to cost-effective engineering design due to fragmentation of information and expertise, and incompatibilities of software tools, it is essential to provide an integrated computerized environment to bolster design collaboration. In an earlier work [1], an *agent-based approach* [2] was proposed to control and coordinate the interactions of *heterogeneous* design software and to enable distributed design tasks to be performed by different software. In particular, this research extends the well-known *contract net protocol* [3] for decentralized task allocation in cooperative engineering design (section 2). By adding a set of agent selection and task selection policies (sections 3 and 4) adapted from [4], the proposed protocol that allows design agents to cooperate by requesting and providing services to other agents through mutual selection. Through *colored petri-nets* [5, 6] simulations, the performance of the proposed protocol under different conditions is measured and evaluated in section 5.

2. A MULTI-AGENT PROTOCOL

In this research, an agent is an *encapsulated* design support software that has its own local database, it can communicate with other agents and have some design problem solving expertise. The expertise of an agent is represented by a set of task operators. Each task operator consists of a list of conditions responding to requests and the corresponding methods for executing a task. To facilitate the inter-operations of (design) agents, an *agent-based infrastructure* for design collaboration was defined [1]. The agent-based infrastructure provides an environment that bolsters (i) information exchange among agents and (ii) allocation, control and coordination of decentralized design tasks.

To allow effective coordination and control of design (sub-)tasks, Sim [1] employs an extension of the *contract net* protocol [3] for allocating agents to design tasks. In this approach, agents coordinate their activities through *contracts* to accomplish a design (sub-)goal. Contracting involves an exchange of information between agents, an evaluation of the information and a final agreement based on mutual selection. When an agent needs the services of other agents, it plays the roles of a *manager* and contract out the task to other *contractor* agents. Since several agents may be eligible to carry out a task, they may bid for the same task. Moreover, contractor agents who wish to provide services may choose from among a list of tasks to bid for. The stages of the proposed multi-agent protocol [1] for cooperative design are given as follows:

1. During a cooperative design process, if an agent (the manager) requires the services of other agents, it will formulate a list of tasks that will be allocated to a group of contractor agents.

2. The manager agent announces the list of tasks to all other agents via a *blackboard* [7] (a global database serving as a common repository to all agents). A task

0-7803-5731-0/99/$10.00 ©1999 IEEE

3. announcement consists of descriptions of the task (such as task id and task type), the expertise required to carry out the task, an estimated execution time and a deadline for receiving bids.

4. Each contractor agent evaluates the list of task announcements and determines its *eligibility* for the tasks. Eligibility is computed by taking the ratio of the cardinality of agent's expertise matching the required expertise (to carry out the task) and the cardinality of the required expertise.

5. Contractor agents select from a list of tasks to submit bids. Tasks are selected based on the criteria described in section 3.
6. The manager agent evaluates the bids put up by all eligible contractor agents. The criteria for evaluating bids are explicated in section 4.

7. Using a set of selection policies the manager agent assigns the task to the most appropriate contractor agent. Details are given in section 4.

8. While the manager agent waits for the results, the agent that is assigned the task matches the request to the list of conditions of its task operators to search for the appropriate methods for carrying out the task. Based on the task specifications given in the task assignment, the task is executed using the relevant methods in the agent's tasks operator.

Generic (domain independent) algorithms for the manager and contractor agents to exchange and evaluate information at the various stages of the proposed protocol closely resemble those of the contract net protocol [3]. The algorithms for stages 4, 5 and 6 such as the task selection and agent selection algorithm are domain specific. While space limitation precludes the inclusion of domain independent algorithms, expositions of the task and agent selection policies together with the corresponding algorithms are given in sections 3 and 4.

Desirable Properties: In [3], it was noted that the contract net protocol is particularly suitable for executing different (sub-)tasks simultaneously while ensuring *effective resource allocation* and *focused decisions*. While effective resource allocation is essential for achieving a balancing of computational load among agents, focused decision ensures that the most appropriate agent is selected to perform a given (sub-)task. These two properties are essential in ensuring high performance in distributed problem solving. In this research, having a *balanced of loading* among a society of agents as well as achieving *high throughput* are viewed as achieving high performance. The rationale is to accomplish a large-scale design task at the quickest possible time by effectively allocating sub-tasks to the most appropriate set of agents. It is argued that the effectiveness of coordination among a society of agents for cooperative engineering design can be studied by observing and measuring the throughput and the balanced of loading. This claim is supported by a series of colored petri-net simulations presented in section 5.

3. TASK SELECTION

Design agents who wish to provide services to other agents can choose among a list of tasks to bid for. The task selection policies prefer tasks that (i) the agent has expertise to carry out (ii) require the least amount of computational time. While the first criteria help determine whether an agent can perform a task, the second criteria considers whether it is desirable to bid for a task. If a task requires considerable amount of computation time, an agent may be prevented from executing other tasks. To ensure high throughput, an agent prefers tasks that consume less of its computational time. Given a list of requests (tasks), a contractor agent uses algorithm 1 to determine which task(s) to place bids. A bid consists of the expertise and experience of an agent together with its *loading factor* (number of tasks assigned to an agent). The **Evaluate Task** procedure in algorithm 1 determines the eligibility of an agent to carry out a task as described in stage 3 of the protocol. **E** is an arbitrary number determined through a series of colored petri-net simulations.

Receive a list of task requests $\mathbf{R} = (req_1, req_2, \ldots, req_n)$;
Set response_list = $\varnothing$;
For each request $req_i \in \mathbf{R}$ do
    Evaluate Task(req_i, eligibility);
    If eligibility > **E** then
      response_list = response_list $\cup$ req_i;
End_for;
Sort the response_list in ascending order by *estimated_execution_time*;
For every $req_i \in$ response_list do
    Generate a bid b_i;
    Transmit b_i to req_i.manager via the out_port;
End_for;

---

*Algorithm 1. The **Bidding** algorithm*

4. AGENT SELECTION

When an agent requires a service from others, the agent selection policies determines how to assign the task to the most appropriate contractor agent. The agent selection policies (adapted from [4]), prefer agents that (1) have the expertise to perform the task, (2) have previously performed a similar type of task and (3) have been allocated the least number of tasks. Both the first and the second criteria accentuate the statement "design problem solving often requires many skills and much knowledge" [8]. While the first criteria searches for agents that has the most matching expertise, the second criteria focuses the search by preferring agents that have experience in

carrying out similar type of tasks. Both the first and the second criteria provide *focused decisions* [3] to select agents with the most appropriate skills and experience to perform the task. The third criteria is to ensure a balanced of loading (effective allocation of computational resources) in the task distribution and hence it prefers agents with the least number of tasks. As mentioned in section 2, to solve the *connection problem* [3] (allocating agents to tasks), both focused decision and effective resource allocation are essential in ensuring high performance of the protocol.

In this research the connection problem for cooperative design is addressed by algorithm 2, the **Evaluate Bids** (Agent Selection) algorithm. It compiles a list of bids and determines how to assign the task to the most appropriate contractor agent. In algorithm 2, criteria 1 and 2 are determined using the **Compute Skill Utility** and **Evaluate Experience** procedure respectively (these procedures are omitted due to space limitation). The **Compute Skill Utility** procedure determines the *utilities* of agents bidding for a task based on the number of their expertise matching the required expertise (utility is similar to the eligibility in section 3). The **Evaluate Experience** procedure verifies if an agent has previously performed a similar task. Criteria 1, 2 and 3 are integrated into algorithm 2 with the following interpretations:

1. If an agent has a utility that is clearly higher than *all* other agents bidding for a task, then it is assigned the task
2. If there are more than one agent having the highest utility then agents with experience in performing similar task are given higher consideration
3. If there are more than one agent with the highest utility and possessing experience in carrying out similar task then the agent with the lowest loading factor is assigned the task

---

Let req_i be a task request, u_a, u_j, and u_k be utilities and b_a, b_j, b_k, and b_m be bids.
Let $bid\_list_i = (bid_1, bid_2, \ldots, bid_n)$ and $skill\_utility\_list_i$ be a list of bids and a list of utilities.
Set $bid\_list_i = \varnothing$;
Set $skill\_utility\_list_i = \varnothing$;
While system_time < req_i.deadline do
  Receive a bid b_j from input_port;
  Set $bid\_list_i = bid\_list_i \cup b_j$;
  Compute Skill Utility(req_i, b_j, u_j);
  Set $skill\_utility\_list_i = skill\_utility\_list_i \cup u_j$;
End_while;
Sort $skill\_utility\_list_i$ in descending order;
Sort bid_list_i according to the order of $skill\_utility\_list_i$;
If $\{u_a > u_k \mid u_a$ is the first element of bid_list_i and $u_k \in bid\_list_i - \{u_a\}\}$,
  Then **Assign Task**(req_i, b_a);
Else
  Set $selected\_bid\_list_i = \varnothing$;
  Set $unselected\_bid\_list_i = \varnothing$;
  For $\{u_a = u_k \mid u_k \in bid\_list_i - \{u_a\}\}$ do
    If **Evaluate Experience**(b_k.agent_id.experience, req_i.task_type) = true
    Then $selected\_bid\_list_i = selected\_bid\_list_i \cup b_k$;
    Else $unselected\_bid\_list_i = unselected\_bid\_list_i \cup b_k$;
    End_if;
  End_for;
  If $selected\_bid\_list_i = \varnothing$ Then
    Sort $\{unselected\_bid\_list_i \cup b_a\}$ in ascending order by loading;
    Select b_m such that $b_m \in \{unselected\_bid\_list_i \cup b_a\}$;
    Assign Task(req_i, b_m);
  Else
    Sort $selected\_bid\_list_i$ in ascending order by loading;
    Select b_m such that $b_m \in selected\_bid\_list_i$;
    Assign Task(req_i, b_m);
  End_if;
End_if;

---

*Algorithm 2. The **Evaluate Bids** (Agent Selection) algorithm*

5. COLORED PETRI-NET SIMULATION

To capture the behavior of the multi-agent task allocation protocol, color perti net (DESIGN/CPN©) is used to construct the simulation experiment. The overall architecture of the protocol process is shown in Figure 1, and the major variables and color sets used are as follows:

```
(* GLobal Value *)
val EligilityPara = 0.5;
val NumTask = 5;
val NumGoal = 5;
val NumSkill = 5;
val NumExpe = 5
val NumOfContractor = 1;
val NumOfManager = 1;
val MaxLoadFactor = 10;

(* Color Set *)
color ID = int;
color AgentID = int;
color TaskID = int;
color ManagerID = int;
color ContractorID = int;
color Role = with MANAGER | CONTRACTOR;
color SkillType = with s1 | s2 | s3 | s4 | s5 | s6;
color TaskType = with t1 | t2 | t3 | t4 | t5;
color GoalType = with g1 | g2 | g3 | g4 | g5;
color Condition = with c1 | c2 | c3 | c4 | c5;
color Action = with a1 | a2 | a3 | a4 | a5;
color ResultType = with r1 | r2 | r3 | r4 | r5;
color Message = with ACCEPT | REJECT;
color ExperienceType = TaskType;
color Loading = int;
color Throughput = int;
color EstExeTime = int;
color FinishTime = TIME;
color Deadline = int;
color RandomNumber = int;
color Eligility = real;
```

```
color ListSize = int;
color Dummy = int;
color Flag = int;
color Counter = int;
color RealNum = real;
color Number = int;
color GotExperience = bool;
color Uskill = int;
color NumOfTask = int with 1..NumTask;
color NumOfSkill = int with 1..NumSkill;
color NumOfGoal = int with 1..NumGoal;
color NumOfExpe = int with 1..NumExpe;
color Finish = product AgentID * TaskID * FinishTime;
color SkillList = list SkillType;
color GoalList = list GoalType;
color Experience = list ExperienceType;
color Task = product TaskID * TaskType * GoalList *
SkillList * EstExeTime * ManagerID;
color ContractedTask = Task;
```

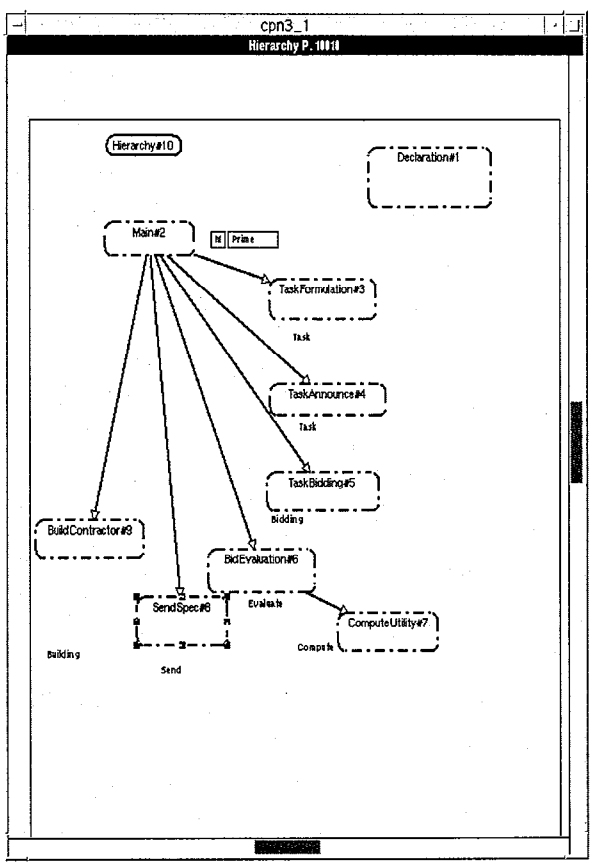

Figure 1 Overall Architecture

The overall architecture consists of one main CPN page and seven sub-pages. The main page consists of the overall major protocol, and this is decomposed into sub-pages for detail executions. The main page is shown in Figure 2. Due to space limitation, only the bidding and the evaluate bid sub-page are shown.

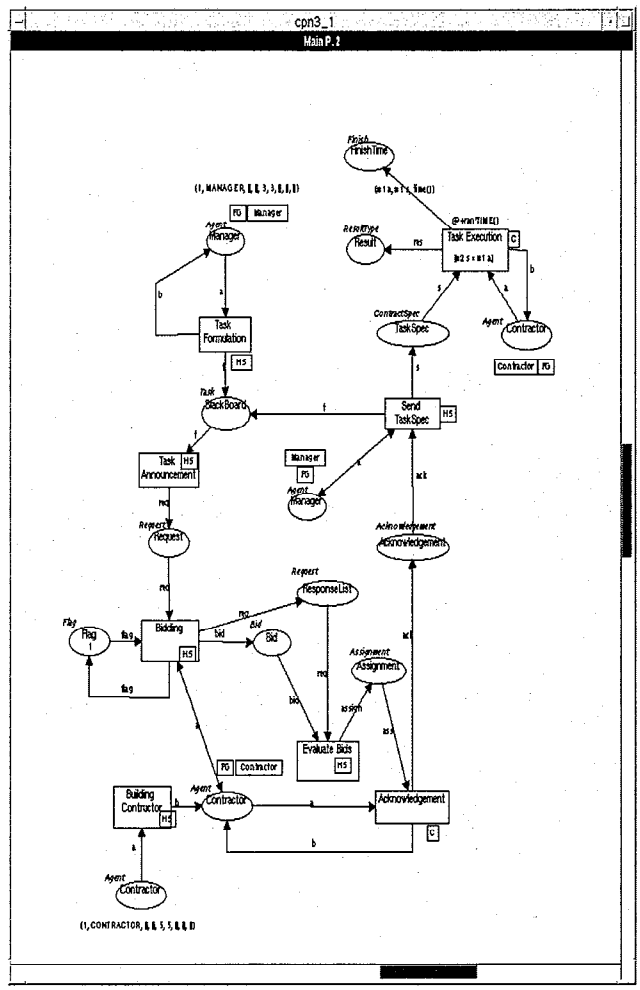

Figure 2 Main page of the multi-agent task allocation process

The major transition in the Bidding CPN sub-page is Generate Bid which is activated by the response list and contractor agent. CPN coding is also attached in this transition based on Algorithm 1, described in section 3. In Figures 4a and 4b, the major transitions declared in the sub page consist of Select Bid, Compute Skill Utility, Assign Task and Evaluate Experience. Algorithm 2, described in section 4 was implemented in this CPN through the construction of various transitions and places declared in this sub-page.

A total of 8 CPN pages, 52 color sets, 25 types of variables, 5 global functions, 30 embedded functions, 150 places and 70 transitions have been constructed. The experiments were conducted using a Sun Ultra machine with Solaris OS 2.5 and DESIGN/CPN version 3.1. The programming effort is around 500 man hours. The results are shown in Table 1.

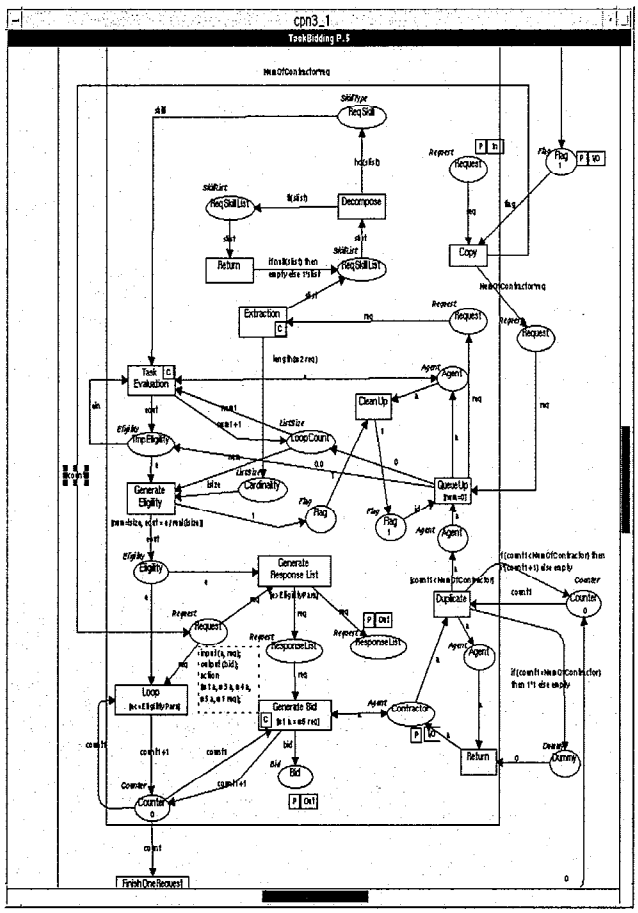

Figure 3 Bidding

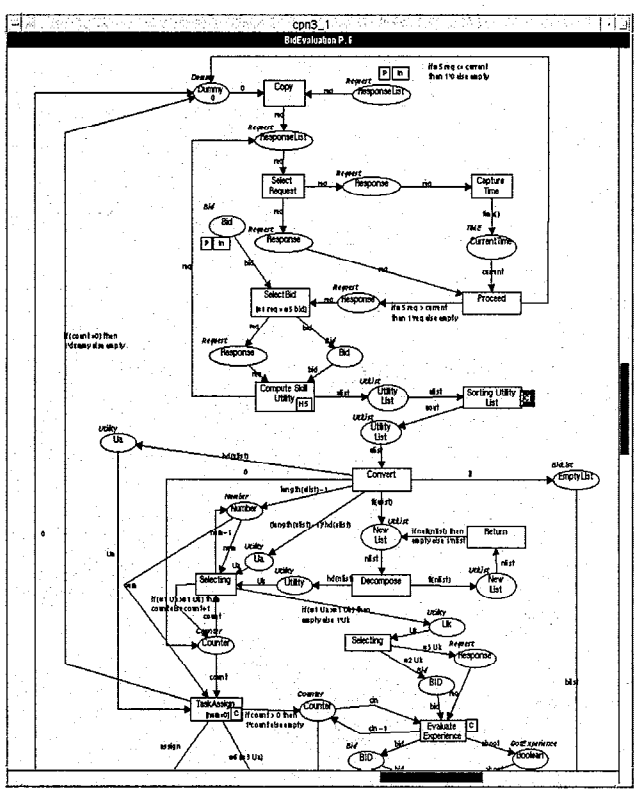

Figure 4a Evaluate Bid

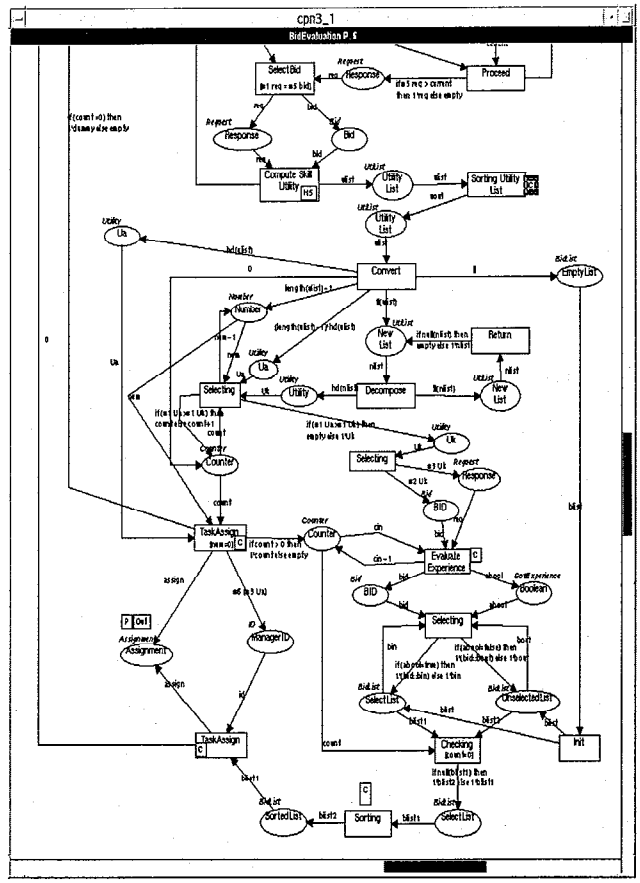

Figure 4b Evaluate Bid

The simulation result is shown in Table 1. By varying the eligibility, a series of experimental results were obtained. Based on the result, it seems that an eligibility of 0.8 leads to a relatively balanced of task assignments among the contractor agents. The experimental result shown in Table 1 was obtained based on the selection policies in sections 3 and 4. While the selection policies were implemented using CPN in Figures 3, 4a and 4b, the balacing of loading in the protocol was determined in CPN simulation.

6. CONCLUSION

With the current proliferation of agent technology [2], it seems natural to adopt multi-agent techniques to bolster cooperative engineering design activities and other applications. In this paper, a protocol for task allocation in cooperative engineering design is proposed. The proposed protocol extends the well-known contract net protocol with a prescribed set of agent selection and task selection policies suitable for cooperative design. Since the contract net protocol only provides a generic framework for agents to exchange information, it does not prescribe any specific coordination policies for agents. While the contract net has been extended to deal with task allocation in manufacturing [9] and job shop scheduling [10], the criteria for task allocation in cooperative engineering design is different. What distinguishes this research from previous approaches based on contract net is that by

simulating the interaction protocol using colored petri net, the performance and properties of the protocol can be measured and verified without actual implementations. In large scale software systems involving multiple agents, considerable amount of time and money can be saved if design flaws can be detected early.

In summary, the contributions of this research are listed as follows:

1. By extending the contract net protocol, a multi-agent protocol for task allocation in cooperative engineering design was devised.
2. Through colored petri-net simulations, it was shown that desirable properties in the contract net protocol such as effective resource allocation and focus attention were preserved in the proposed protocol.

Although there has been previous work [11] in using colored petri net as a tool for specifying and simulating multi-agent systems, research in this area is still in its infancy. It is hoped that the approach presented can shed new light in designing and engineering multi-agent systems.

	Agent 1 (Manager)		Agent 2 (Manager)		Agent 3 (Contractor)			Agent 4 (Contractor)			Agent 5 (Contractor)		
E = eligibility	Task Generate	Task Request	Task Generate	Task Request	Bid Generate	Task Assign	Finish Time	Bid Generate	Task Assign	Finish Time	Bid Generate	Task Assign	Finish Time
0.1	4	4	2	2	6	3	3791	5	1	621	6	2	1420
0.2	3	3	2	2	4	2	1123	3	1	762	4	2	1332
0.3	5	5	1	1	4	2	1367	2	0	0	6	4	3231
0.4	2	2	2	2	4	2	1442	3	1	821	2	1	562
0.5	1	1	1	1	1	1	578	1	1	634	1	0	0
0.6	4	4	5	5	2	2	1294	1	1	734	0	0	0
0.7	2	2	4	4	1	1	533	1	1	645	2	2	1479
0.8	5	5	5	5	1	1	479	1	1	843	1	1	753
0.9	2	2	2	2	0	0	0	0	0	0	0	0	0

Table 1 Simulation Results

ACKNOWLEDGEMENT

This research is funded by a grant (grant number S844) from the Department of Computing at the Hong Kong Polytechnic University. The authors gratefully acknowledge the financial support and research facilities from the Department.

REFERENCES

[1] K. M. Sim. A Multi-agent Testbed for Cooperative Design. AI Technical Report AI-TR-98-1, ISIR, Osaka University, Japan.
[2] M.N. Huhns and M.P. Singh. Readings in Agents. Morgan Kauffman Publisher, 1997.
[3] R. G. Smith. The Contract Net Protocol: High-level Communication and Control in Distributed Problem Solver. IEEE Transaction on Computers, volume C-29, number 12, pages 1104-1113, 1980.
[4] G. Kappel, S. Rausch-Schott, and W. Retschitzegger. Coordination in Workflow Management Systems: A Rule-Based Approach. In W. Conen and G. Neumann (Eds), Coordination Technology for Collaborative Applications: Organization, Processes and Agents, pages 99 - 119, Springer-verlag, 1998.
[5] K. Jensen. Colored Petri Nets: Basic Concepts, Analysis Methods and Practical Use, Volume 1 in 1992. Volume 2 in 1995 and Volume 3 in 1997. Springer-Verlag.
[6] Simon C.K. Shiu, Eric C.C. Tsang, Daniel S. Yeung and Martin B. Lam, "Evaluation of Printed Circuit Board Assembly Manufacturing Systems Using Fuzzy Colored Petri Nets," in Proceedings of 1998 IEEE International Conference on Systems, Man, and Cybernetics, Hyatt Regency La Jolla, San Diego, California, USA, Oct 11-14, Vol. 2, pp. 1506-1511, 1998.
[7] D. Corkill. Blackboard Systems. AI Expert, September 1991, pages 41-47.
[8] S. Shapiro. Encyclopedia of artificial intelligence, 2nd ed., page 331, Wiley, New York,1992, p 331.
[9] H. van Dyke Parunak. Manufacturing Experience with the Contract Net. In Research Notes in Artificial Intelligence: Distributed Artificial Intelligence, Vol. 1, pages 285 - 310, Morgan Kaufmann Publishers, 1987.
[10] Liu and Sycara. Multi-agent Coordination in Tightly Coupled Task Scheduling. In [2], pages 164 – 171.
[11] Moldt and Wienberg. Multi-agent Systems Based on Colored Petri Nets. Application and Theory of Petri Nets, Springer-verlag, 1997.