Available online at www.sciencedirect.com
ScienceDirect
Journal of Computer and System Sciences 74 (2008) 255–278
JOURNAL OF COMPUTER AND SYSTEM SCIENCES
www.elsevier.com/locate/jcss

Topological analysis of AOCD-based agent networks and experimental results

Hao Lan Zhang *, Clement H.C. Leung, Gitesh K. Raikundalia

School of Computer Science and Mathematics, Victoria University, PO Box 14428, Melbourne City, MC 8001, Australia

Received 15 June 2006; received in revised form 31 October 2006

Available online 24 April 2007

Abstract

Topological analysis of intelligent agent networks provides crucial information about the structure of agent distribution over a network. Performance analysis of agent network topologies helps multi-agent system developers to understand the impact of topology on system efficiency and effectiveness. Appropriate topology analysis enables the adoption of suitable frameworks for specific multi-agent systems. In this paper, we systematically classify agent network topologies and propose a novel hybrid topology for distributed multi-agent systems. We compare the performance of this topology with two other common agent network topologies—centralised and decentralised topologies—within a new multi-agent framework, called *Agent-based Open Connectivity for DSS* (AOCD). Three major aspects are studied for estimating topology performance, which include (i) transmission time for a set of requests; (ii) waiting time for processing requests; and (iii) memory consumption for storing agent information. We also conduct a set of AOCD topological experiments to compare the performance of hybrid and centralised agent network topologies and illustrate our experimental results in this paper.

Keywords: Agent network topology; Agent network performance; Intelligent agents; AOCD

1. Introduction

Applications of multi-agent systems have been arising in many areas. Agent-based systems provide a way of conceptualising complex software applications. These applications often face problems involving multiple and distributed sources of knowledge [1]. However, current research work with respect to topological theory in the area of intelligent agents is inadequate. This situation has led to a set of important research problems concerning how an agent network should be designed to perform efficiently and effectively. The agent network topology issue becomes one of the key issues in solving problems of agent cooperation and communication. An appropriate agent network topology design helps to enhance agent communication efficiency and network mobility.

An *agent network topology* represents the information of agent distribution over an agent network, which incorporates agent mobility and intelligence aspects into the process of arranging and configuring an agent network. The term,

* Corresponding author.
E-mail addresses: haolan@sci.vu.edu.au (H.L. Zhang), clement.leung@vu.edu.au (C.H.C. Leung), gitesh.raikundalia@vu.edu.au (G.K. Raikundalia).

0022-0000/$ – see front matter
doi:10.1016/j.jcss.2007.04.006

"agent network topology," is derived from mathematical topological theory. This concept overlaps with topological theory in data communications and distributed systems areas.

Existing topological theories in the computing field have been mainly applied to data communications and distributed systems areas. These theories have made some extraordinary contributions. However, as an emerging discipline, topological theory in multi-agent systems is inadequate. Existing topological theory cannot fulfil the needs of an agent network because an agent network has its specific characteristics, which include: (i) mobility, (ii) intelligence, and (iii) flexibility. Agent network topologies take not only agent distribution into consideration but also consider agent mobility in a network. Most of the current research work in the agent network topology area adopts topological theory from the distributed system and computing network fields without considering mobility and intelligence aspects.

Current research work in the agent network topology area is also not systematic and relies on graph-based methodology. Graph-based topological analysis of a network topology is often based on the network graph provided and sometimes lacks precise measurements of each agent. Moreover, existing agent network topologies are incapable of providing much detailed information of each agent and its relationship with other agents on a network. This increases the difficulty of agent communication and cooperation, such as agent searching or matching, over a network. Thus, the research direction of agent network topologies needs to follow these three characteristics based on some concrete network performance analysis. Therefore, we propose a classification of the current agent network topologies. We also provide theoretical analysis and experimental results to compare the performance of three common agent network topologies.

Extensive research has been carried out to analyse the performance of network topologies in the Distributed Artificial Intelligence (DAI) and network areas [2,3]. However, performance analysis of agent network topologies has been inadequate in the multi-agent systems area as it is an emerging discipline. In this paper, we carry out performance analysis of three major agent topologies: (i) centralised, (ii) decentralised, and (iii) hybrid topology (mesh + centralised) based on the AOCD architecture. The performance analysis presented in this paper provides a concrete and innovative methodology to analyse agent network topologies, especially within the AOCD framework.

This paper is organised as follows. The next section introduces related work that has been done in agent network topology and distributed systems areas. Section 3 describes the concepts of AOCD and Matrix design. Section 4 classifies agent network topologies and provides mobility analysis based on simple agent network topologies. In Section 5, we conduct a theoretical performance analysis of three common agent network topologies based on the AOCD architecture. Section 6 introduces the experimental design of the AOCD topological experiment and Section 7 provides our experimental results. In the final section, we conclude our research work and consider future issues that need to be addressed.

2. Related work

The AOCD architecture is an agent-based decision support system (DSS). The work presented in this paper is the intersection of network topology theory and topological application in DSS. Therefore, the related work of our research can be divided into two major areas. The first area relates to network topological theory, and the second area relates to topological application in DSS.

2.1. Network topological theory

The related work in this area mainly focuses on the development of distributed system theory and complex network theory.

Intelligent agent technology such as a multi-agent application is often considered as a sub-discipline of DAI. Research work of topological theory applied to digital networks and distributed systems areas, including DAI, has been carried out (e.g. [2,3]). Much work in distributed systems and digital networks areas can be applied to the intelligent agent area, including topological theory.

Minar [2] classifies distributed system topologies into several basic categories based on some common experience. In Minar's hybrid topology, there are three major forms: (i) a centralised + centralised, (ii) centralised + ring, and (iii) centralised + decentralised (partial connection). The hybrid topology in AOCD design is different from Minar's hybrid theory and adopts centralised + mesh topology (fully connected). The centralised + mesh topology is an efficient but

costly solution; nevertheless, existing technologies are able to reduce costs by using message passing or Supernode [4] mechanisms. The topological theory developed in Minar's work is not systematic and lacks comprehensive analysis of topological theory. Nevertheless, it provides some basic ideas about the classification of simple distributed networks and is helpful for the development of topological theory in multi-agent systems.

After the proposal of Small-world theory and its application to complex networks [5–7], the development of topological theory soon spreads to complex agent networks [8,9]. Nevertheless, topological theory in the multi-agent field is still preliminary. Most of the related work in the field emphasises the application of a specific topology, such as Small-world topology or Scale-free topology, to agent networks. This work does not raise the issue of classification of agent networks, which limits the research in gaining a comprehensive understanding of agent networks. Thus, we contribute this paper to clarify current agent network topological theory.

Apart from the above theoretical work of analysing network topologies, some concrete methodologies have been provided to evaluate network performance and efficiency, which are useful to apply to topological research in agent networks. For instance, (i) cost calculation methodologies for wide area network design [3] have been used for network design estimation; (ii) performance analysis of distributed systems has been carried out by Androutsellis-Theotokis and Spinellis [10] and Helsinger et al. [11]; (iii) analysis of network topology impact on dependability offers an evaluation method for estimating the effect of topology on dependability [12].

2.2. Topological application in DSS

The related work in this area mainly emphasises using multiagent technology to develop an efficient architecture for DSS.

The traditional design methodologies of DSS are facing challenges from multi-agent technology. For instance, Gachet and Haettenschwiler [13] proposed a decentralised approach to DSS. This approach overcomes the disadvantages of existing DSS and offers flexible, extendable, and mobile features to DSS. However, the major disadvantage in Gachet and Haettenschwiler's framework is the lack of manageability, which is caused by removing central control [14]. Concurrent control and synchronicity problems are the major difficulties that hold back decentralised systems' manageability. Moreover, the modules used in their proposed architecture lack sufficient intelligence and mobility. As a result, their system cannot be easily transplanted and it is also inefficient in system reusability and independency.

In recent years, the hybrid topology has been used in distributed systems like those described in Groove [15], KaZaa [16], and Morpheus [17]. The successful performance of these systems inspires the use of hybrid topology in distributed DSS, and as a result the AOCD architecture is proposed. The concept of AOCD architecture design has been introduced in our previous work [18]. The AOCD architecture makes use of a unique component, the *Matrix*, to enhance the central control capability. AOCD design eliminates the concurrent control and synchronicity problems that plague many decentralised systems.

3. AOCD and Matrix

The AOCD architecture is supported by a set of methodologies to ensure its efficiency and effectiveness. Using the AOCD architecture, as Fig. 1 shows, an external agent can be plugged into an existing DSS as a component without disturbing the existing system structure.

In this architecture, a DSS is divided into a set of subsystems and each subsystem will be represented by at least one agent. A central control panel, called the *Matrix*, consists of four layers. The Matrix is deployed to connect the different agents. The Matrix in an AOCD framework is standardised and independent of the environment. In other words, an AOCD system may vary according to the different business processes used by different organisations; however the Matrices used in these different systems are standardised.

The Matrix is the essential design in the AOCD architecture, which distinguishes the AOCD from other multi-agent based systems. A Matrix has three major parts including: (i) agent society that provides a virtual space to agents, (ii) *information acquisition section* that acquires knowledge from the external environment, and (iii) *Matrix control panel* that is the core part mainly handling agent matching, request processing, and resource allocation. Figure 2 shows a conceptual view of the Matrix. The Matrix does not act as an agent facilitator or mediator that mainly deals with agent coordination. In contrast, the Matrix incorporates agents in its virtual space in which agents can obtain information, self-upgrade, and be reused.

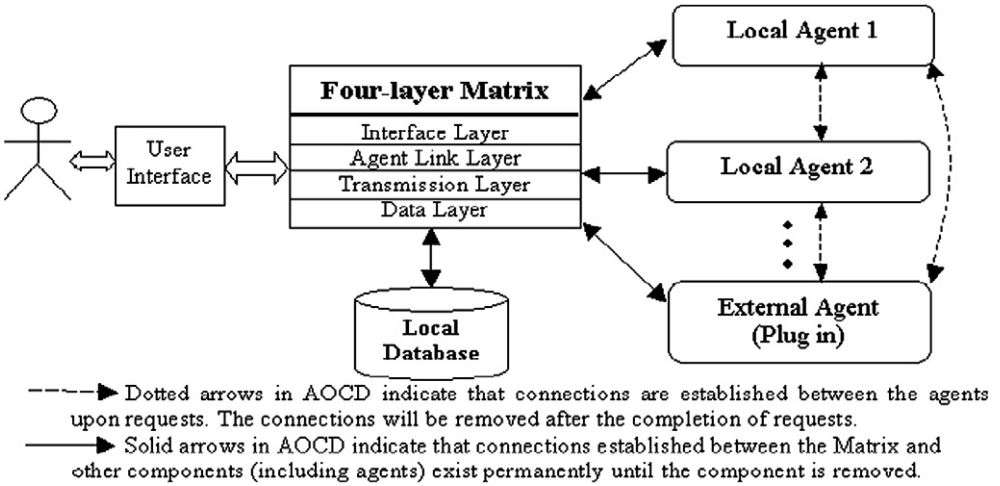

Fig. 1. Conceptual view of AOCD architecture.

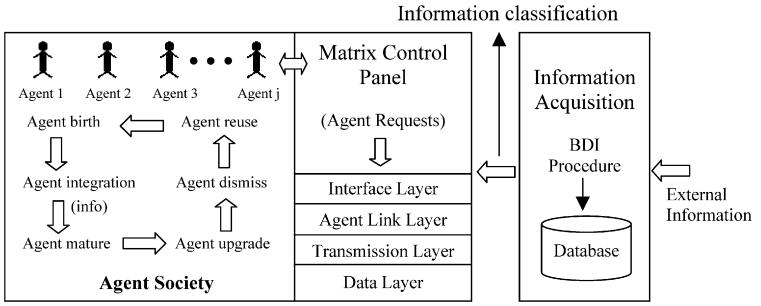

Fig. 2. Conceptual view of Matrix.

The Matrix control panel manages the agent society through establishing relationships with agents. An agent society can be regarded as an aggregation of a set of agent relationships. The set of relationships between agents is the most important issue in the Matrix concept. The other facts, such as actual physical space (or memory consumption), geographic distance, and dimensional issue of the agent society are not major concerns for the Matrix concept. As shown in Fig. 3, the Matrix adopts a hybrid agent network topology. The hybrid agent network topology is the most efficient topology for the AOCD architecture. In the hybrid topology, agents are connected directly to each other and they are all connected to the Matrix. This design improves the communication efficiency between agents and reduces the workload of the Matrix to avoid a bottleneck problem, which plagues many centralised systems.

4. Classification of agent network topologies and mobility analysis

In the real world, a network for an industry organisation is complex and specific to that organisation. Therefore, it is not realistic to establish a general model for many different agent networks. However, a complex network can be divided into several simple topologies. For instance, Local Area Network (LAN) theory generally defines four basic topologies [3,19], which include star topology, bus topology, ring topology, and tree topology. In practise, we often use these topologies by combining them into one network, which is called a hybrid topology. This phenomenon also has been found in distributed systems such as multiagent systems. Based on the traditional topological theory of the LAN, Minar [2] suggests four basic simple topologies and two hybrid topologies in evaluating distributed systems topologies. These two hybrid topologies can be viewed as the combinations of the four basic simple topologies.

As we mentioned in the previous sections, the topological theory in multiagent systems area is very preliminary. A systematic study on agent network topologies is urgently required. In this paper, we summarise our previous work [24] and classify the agent network topologies into two main categories, including (i) simple agent network topology,

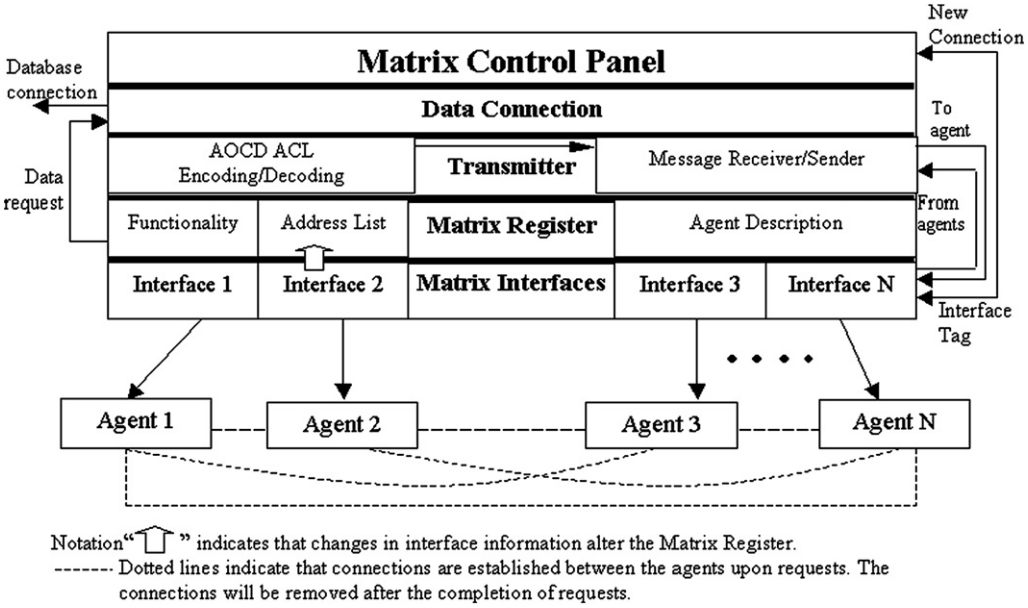

Fig. 3. Matrix control panel design.

and (ii) complex agent network topology. Although, the complex agent network topologies can be divided eventually into a number of simple agent network topologies. We argue that it is very necessary to develop the topological theory for complex agent networks. This is because complex network theory provides a conceptual and abstract view of an overall network especially for a large-scale agent network.

4.1. Simple agent network topologies

In the multiagent area, agent network topologies not only take agent distribution into consideration but also consider agent mobility and intelligence in a network. The intelligence factor is one of the key factors that distinguish agent network topology theory from existing topological theory. The individual nodes in existing networks, such as LAN or distributed systems, do not pay much attention to intelligence issues. In an agent-based network, intelligence is a key issue as it is one of the most important characteristics of an intelligent agent. Our recently proposed language [24] for agent network topologies, called *Topological Description Language for Agent networks* (TDLA), reflects the intelligence factor in agent network topology theory. TDLA enables an individual agent to perceive other agents' information and the information of the network that the agent resides in. Therefore, an agent can be proactive before performing any actions, such as moving or matching, in an agent network. In this paper, we mainly focus on the agent mobility factor as the TDLA has been discussed in our previous work [24]. The mobility factor is another key issue in agent network topology research. Agent mobility is affected by the distribution of agents over a network and the connections between agents. These facts that affect agent mobility are basically topological issues. We classify the simple agent network into six basic topologies:

(1) Centralised agent network topology.
(2) Peer-to-peer agent network topology.
(3) Broadcasting agent network topology.
(4) Closed-loop agent network topology.
(5) Linear agent network topology.
(6) Hierarchical agent network topology.

Mobility analysis of the agent network topology is the key issue as it directly impacts on the overall network performance. Therefore, in this section we emphasise topological classification and mobility analysis of each simple agent network topology.

4.1.1. Centralised agent network topology

Our definition of the centralised agent network topology is: the topology has a central agent and only this central agent is connected with other agents over the network. There is no direct connection between any others agents except with the central agent.

In the centralised agent network topology, the central agent is vital to the network. However, the central agent's mobility is very inefficient. Total connection of a topology is one of the key facts that affect agent network mobility. Based on our previous agent mobility analysis [24], we develop the following equation to define the connection-based coefficient of agent network mobility:

$$m_{\mathrm{co}} = \frac{1}{\sum_i (\text{Totalconnection/import}) \times m_i}, \tag{1}$$

where m_{co} is the connection-degree coefficient of mobility. m_i is the mobility of an individual agent i, which indicates the degree of flexibility that an agent detaches from an agent network. *Import* means the importance of an agent that is measured by the number of connections the agent has. These variables also will be used in the remaining topologies in this section. Based on Eq. (1), the connection-based coefficient of centralised agent network mobility is:

$$m_{\mathrm{co}} = \frac{1}{\sum_{i=1}^{v} ((v-1)/a_i) \times m_i}, \tag{2}$$

where a_i is the number of connections for agent i, v denotes the total number of agents over a network including the central agent. The variables v and a_i will also be used in the following topologies in this section.

The centralised agent network topology has a relatively stable mobility coefficient in terms of increasing agent numbers. The terminal agents of the centralised topology have high mobility as they only have one connection to the network centre. However, the whole network relies heavily on the central agent that makes the mobility of the central agent very inefficient.

4.1.2. Peer-to-peer agent network topology

In the peer-to-peer agent network topology, each agent has direct connection(s) with other node(s) over a network. There is no restriction on the connections between agents. There are two categories in the peer-to-peer agent network topology, one is the fully connected peer-to-peer topology and another is the partially connected peer-to-peer topology. In the fully connected peer-to-peer topology, each agent has connections with all the other agents over a network. In the partially connected peer-to-peer topology, each agent has at least one connection with another agent over a network and the maximum connections is smaller than c. In the fully connected peer-to-peer topology, if v denotes the number of agents over a network, then c is given by exhausting all combinations from choosing any two agents:

$$c = \binom{v}{2} = \frac{v(v-1)}{2}, \quad 1 \leqslant v < \infty. \tag{3}$$

Thus, the connection-based coefficient of the fully connected peer-to-peer agent network mobility is:

$$m_{\mathrm{co}} = \frac{1}{\sum_{i=1}^{v} (c/a_i) \times m_i}. \tag{4}$$

In the fully connected peer-to-peer agent network, the agent network mobility decreases rapidly when the total agent number increases.

4.1.3. Broadcasting agent network topology

We define the broadcasting agent network topology as follows: all the agents are connected through a common media and there is no direct connection between any of the two agents. A bus topology is a typical broadcasting agent network topology.

In the broadcasting agent network topology, every agent has the same role in the network (unlike in the centralised agent network topology, the central agent has a more important role than other agents) and there is no direct connection between any two agents. The connection-based coefficient of the broadcasting agent network mobility is:

$$m_{\mathrm{co}} = \frac{1}{\sum_{i=1}^{v} (v/a_i) \times m_i}. \tag{5}$$

The network mobility of the broadcasting agent network topology is stable and efficient. The importance of agents over the network is the same. Each agent has only one connection to the media, which results in the mobility of agents over the network being even.

4.1.4. Closed-loop agent network topology

In the closed-loop agent network topology, a network forms a loop and each agent connects exactly to two other agents. In the case of removing connections between any two agents, the topology will turn into a linear topology.

Simple ring and dual ring topologies are the typical closed-loop topology. The connection-based coefficient of closed-loop agent network mobility is:

$$m_{\mathrm{co}} = \frac{1}{\sum_{i=1}^{v} (v/a_i) \times m_i}. \tag{6}$$

The coefficient calculation of network mobility is based on the total connections over the network. Therefore, the closed-loop agent network topology has the same coefficient as the broadcasting topology as they have the same connections when they have the same number of agents over a network. However, for an individual agent, the closed-loop agent network topology is less efficient than the broadcasting agent network topology because there are two connections for each agent in the closed-loop topology.

4.1.5. Linear agent network topology

In the linear agent network topology, all agents are distributed in linear form in sequential order and there is no loop in the network. Each agent is connected to two neighbour agents except two end agents that are connected to only one neighbour agent. In many cases, this topology is regarded as inefficient because the communication between two end agents is extremely inefficient. However, in some cases it appears to be efficient such as in a pipelining process.

The connection-based coefficient of linear agent network mobility is:

$$m_{\mathrm{co}} = \frac{1}{\sum_{i=1}^{v} ((v-1)/a_i) \times m_i}. \tag{7}$$

Basically, the linear agent network topology can be considered as a ramification of closed-loop topology. The coefficient calculation of network mobility based on the total connections is slightly smaller than for the closed-loop topology. In practise, we regard the coefficients of the network mobility of these two agent network topologies as the same, especially when the number of agents is large.

4.1.6. Hierarchical agent network topology

In the hierarchical agent network topology, an agent is a basic unit and a number of agents form a group, which is connected to an upper level agent. In the hierarchical agent network topology, an agent is not connected to other agents except with its upper level agent.

A recursive method can be used to determine if a topology is a hierarchical topology by starting from the end points, which has no lower level agent connected. We also define a very important characteristic of hierarchical topology: there is no loop in a hierarchical topology. The connection-based coefficient of the hierarchical agent network mobility is:

$$m_{\mathrm{co}} = \frac{1}{\sum_{i=1}^{v} ((v-1)/a_i) \times m_i}. \tag{8}$$

The total numbers of network connections for hierarchical and linear topologies are the same when their agent numbers are the same. However, the mobility of an individual agent in a hierarchical topology is more complex because the importance of each individual agent in a hierarchical network is different. This fact results in the removal of the important agents (those that have more connections), which are inefficient and it consequently decreases the overall network mobility.

4.2. Complex agent network topologies

Traditional topological theory is insufficient to describe a complex agent network such as a multi-agent system. This is because traditional topological theory is unable to define the relationships between nodes and describe an overall view of the network efficiently. In these circumstances, the topological theory for a complex network is required to provide more abstract descriptions. Current topological theory classifies networks into four major categories:

(1) Regular Network Topology.
(2) Random Network Topology.
(3) Small-World Network Topology.
(4) Scale-free Network Topology.

In a regular network, nodes (agents) are distributed orderly and the connections between nodes are based on certain constraints, for example, the wiring process is based on finding neighbour agents the shortest distance from each other [5].

In a random network, the connections between two agents are generated randomly [20]. Random network is more realistic than regular network in describing real-world complex networks such as multi-agent systems. However, the limitation of the random network topology is the difficulty of predicting, monitoring and controlling the network.

In the real world, a network topology is assumed to be either completely regular or completely random. The actual topology is somewhere between these two extreme cases and it is defined by Watts and Strogatz as the *Small-world* topology [5]. Small-world network theory enables possible control or monitoring over the network especially in some critical areas of a network through observing and adjusting the probability p. p denotes the probability of randomness when an agent is connecting to other agents. When p is 0, a network is wired completely in order. When p is 1, a network is wired completely randomly. For $0 < p < 1$, a network is in small-world topology scope.

The three topologies we discussed in the previous sections are basically static and homogeneous, with peak at an average value and decay. Such networks are called *exponential networks* [21]. However, recent research in the field of complex networks indicates that a number of large-scale complex networks, including the Internet, World Wide Web, and metabolic networks, are scale-free [22] and the vertices over such a network are not homogeneous. The scale-free network topology is considered the most suitable topology for multi-agent based systems taking high-degree mobility of an agent network into account.

Mobility calculations for complex agent network topologies are based on that agent network's specific facts. We argue that there is no general and precise equation (or a set of equations) for a complex agent network's mobility calculation because every complex agent network has a very different structure to other complex agent networks. Unlike the simple agent network topology, the classification of complex agent network topologies is conceptual and abstract. Therefore, the mobility calculation for a complex agent network has to be specific to that network. We recommend using a macro-statistics methodology to calculate the mobility of complex agent network instead of dividing the complex network into numerous simple agent networks and calculating their values individually. We predict that the calculated result of a complex network topology should be close to the combination of numerous simple agent networks' calculated results.

4.3. Hybrid agent network topology

The hybrid agent network topology is an important concept in traditional topological theory, which combines two or more simple agent network topologies. Since the hybrid agent network topology is relatively more complex than the other six basic simple agent network topologies, it is often regarded as a complex network. Technically speaking, the hybrid agent network topology is still a simple agent network topology because it operates on a small scale compared to the complex network topologies from a large-scale point of view. However, hybrid agent network topology has both characteristics of simple and complex agent network topologies.

A hybrid agent network cannot describe an overall complex network in a specific and efficient way. However, it can explain a complex network in a simple way and it is efficient to a limited scale. It can solve some complex problems that the simple agent network topology cannot solve appropriately. For instance, two different agent networks can be joined together to form a hybrid agent network without changing their current structures. A hybrid agent network

topology can be a combination of any two or more simple agent network topologies such as a closed-loop topology and a centralised topology. The hybrid network topology eliminates the difficulty of concurrent control, which mainly plagues the centralised topology. Our study shows that the hybrid topology offers superior performance in agent-based systems [8].

Nevertheless, simple agent network topological theory (including hybrid topology) can only describe a limited scale of agent networks in a simple way. Hence, current topological theory in the multi-agent field adopts more complex topological theory such as small-world topology and scale-free topology to describe a large-scale complex network.

Similar to that of the complex agent network, the mobility of a hybrid agent network topology is the combination of a number of simple agent networks. Unlike the complex agent network topology, in a hybrid agent network we can divide a network into a number of simple agent networks because we define a hybrid agent network as being much smaller than a complex agent network, which is formed by a couple of simple agent networks. For an AOCD-based network, the coefficient of the network mobility is the same as that of the centralised agent network topology, which is:

$$m_{\mathrm{co}} = \frac{1}{\sum_{i=1}^{v}((v-1)/a_i) \times m_i}. \tag{9}$$

The reason for Eq. (9) is that: in an AOCD-based system, the connections between the Matrix and agents exist permanently, which are $(v-1)$ connections (v is the total number of nodes on the network). But the connections between agents exist temporarily. Therefore, we will not take the temporary connections between agents into account although they might slightly decrease the network's mobility.

5. Theoretical performance analysis based on three common agent network topologies

The agent framework in AOCD architecture is a hybrid topology combining centralised and decentralised topologies. Three common network topologies are suggested for the agent framework [13]: centralised topology, decentralised topology and hybrid topology. The hybrid topology is likely to be the most efficient framework for agent communication in the AOCD architecture. In the AOCD architecture, the Matrix plays as a centralised coordinator that mainly dealing with agent relationship management and the communications between agents are decentralised. The following analysis quantifies the performance of implementing each topology in AOCD. Our analysis adopts the following notations:

- i: the position of a request set in a queue,
- J: total number of agents in an AOCD system,
- M: the number of requests that the Matrix can handle at any one time,
- N: the total number of n requests that are sent by a number of agents in a short period of time,
- R: the average size of a record in the agent information list,
- T: the average transmission time between two nodes,
- TR: the total transmission time for N requests,
- W: the total waiting time for all the agents.

The precondition of this analysis is that all the sampling requests are sent synchronously. In addition, we assume that a waiting queue is deployed to store the agents that have not been processed and will be processed in the future. It excludes the waiting time while two agents are connecting and serving. We present the following calculations for the three different agent network topologies.

5.1. Centralised agent network topology

In the centralised topology, as Fig. 4 shows, an agent sends requests to the Matrix. The Matrix delivers the requests to the corresponding agents and returns the results to the requesting agent.

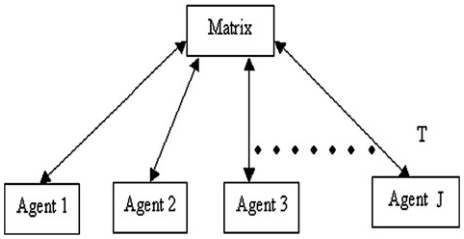

Fig. 4. Centralised transmission framework.

5.1.1. The transmission time for a set of requests in a short period of time

All the communications between agents are transmitted through the Matrix. The total transmission time for N requests is bounded:

$$\lceil 2N/M \rceil \times 2T \leqslant TR \leqslant \lceil N/M \rceil \times 2T + N \times 2T. \tag{10}$$

The reason for $TR \geqslant \lceil 2N/M \rceil \times 2T$ is as follows. In the best case, the Matrix receives N requests from requesting agents then delivers them to the corresponding agents; and ideally the corresponding agents send back the results to the Matrix synchronously. In other words, there are $2N$ requests sent to the Matrix, which includes N requests from the requesting agents and N requests from the corresponding agents. All the results are sent to the Matrix synchronously. Therefore, $2N$ requests are broken into $\lceil 2N/M \rceil$sets. One set of requests contains 1 to M requests and the Matrix can handle one set each time. It takes $2T$ to process one set of requests, which include the requests from the requesting agent costing $1T$ and the results from corresponding agents costing $1T$. The transmission in the centralised topology is a two-way transmission including: sending (or receiving) requests (or results) to (or from) the Matrix and receiving (or sending) results (or requests) from (or to) the Matrix.

The reason for $TR \leqslant \lceil N/M \rceil \times 2T + N \times 2T$ is as follows. The transmission time for delivering the requests from the requesting agents to the Matrix is $\lceil N/M \rceil \times 2T$, which happens synchronously. In the worst case, the results return to the Matrix one-by-one and each request costing $2T$ of transmission time. The transmission time for delivering the results is $N \times 2T$. Therefore, the total transmission time for the requests is $\lceil N/M \rceil \times 2T$ plus $N \times 2T$.

5.1.2. Total waiting time for completing a set of requests

In many cases, the agents are sending requests synchronously. Therefore, a waiting queue is deployed for storing the later-coming requests (see Fig. 5).

For each requests set, the waiting time in a queue is: $(\lceil i/M \rceil - 1) \times 2T$. As mentioned before, each requests set contains 1 to M requests. Our calculation sums up each individual request set as these requests sets may occur in different places; therefore we need to evaluate the total time for each of the requests set.

The total waiting time for all request sets in the queue is:

$$2T \sum_{i=1}^{\lceil 2N/M \rceil} i, \tag{11}$$

where $N > M > 0$. The reason for $N > M$ is that if the queue is not empty then this implies that there must have been more than M requests sent to the Matrix. Otherwise, the waiting queue will be empty which means that there is no waiting time during transmission processes.

In (11), we find that the average transmission time for a set of successful requests in the same period is increased when the total number of requests is increased. The unsuccessful transmission time will not be included in this model because the Matrix eliminates most of the conflict requests between agents.

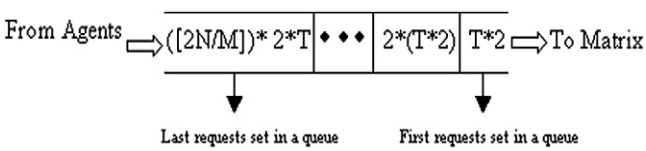

Fig. 5. Waiting time in a queue.

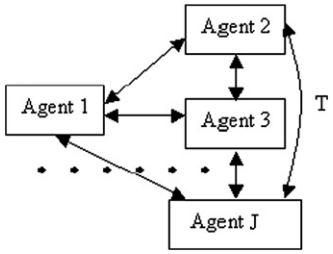

Fig. 6. Decentralised transmission framework.

5.1.3. Memory consumption for storing agent information

In the AOCD framework, the memory allocated to store agent information by using the centralised topology is: $R \times J$. The reason for this outcome is that in the centralised topology all of the agent information is stored in a central component. The other nodes (agents) will not keep the agent information. Therefore, the memory allocated to store the agent information is the size of each information record, which is R, multiplied by the agent number, which is J.

5.2. Decentralised agent network topology (fully connected peer-to-peer)

Unlike other decentralised topologies, such as Minar's [2] decentralised topology, which is a partially connected topology, the decentralised topology discussed in this paper is a fully connected topology. As Fig. 6 shows, the decentralised topology provides direct communication between agents.

This method eliminates the transmission time between agents and the Matrix. However, the coordination and cooperation approaches are more complicated in the decentralised topology than other topologies as agents are communicating directly and individually.

5.2.1. Transmission time for a set of requests in short period

In a decentralised framework, no matter how many requests are occurring concurrently, the total transmission time for N requests is bounded by:

$$2T \leqslant TR \leqslant T + N \times T, \tag{12}$$

where $2T$ is the best case when the communication between agents happens synchronously. In other words, the requesting agents send the requests, which costs T and ideally send back the results to the requesting agent, which also costs T.

$T + N \times T$ is the worst case in which the requesting agents send requests in time T but the results from the corresponding agents are received asynchronously. Only one result transmitted over the network each time. In the decentralised framework, the probabilities for successful requests are decreased dramatically when synchronous requests are increased in number.

5.2.2. Total waiting time for completing a set of requests

Compared with the centralised topology, the communications over the decentralised network are more likely to be disorderly. Central waiting queue for the decentralised topology is not feasible because there is no central control. Therefore, the requests in the decentralised topology will be calculated individually. Concurrent control methodologies are required, and all the requests will send detection signals repeatedly until the corresponding agent is available. In spite of deadlock and other concurrent control failures, the waiting time for all requests is bounded by:

$$0 \leqslant W \leqslant \sum_{i=1}^{N} (2N - 2i), \tag{13}$$

where, if an agent has made X requests, then this agent will be counted as X number of agents. The best-case situation is when all the requests are accepted and processed without delay. In other words, all the requesting agents find their corresponding agents are all available. Therefore, the waiting time in the best case is 0.

Table 1
Total waiting time for processing 4 requests (worst case)

Agent	Agent 1	Agent 2	Agent 3	Agent 4	Agent 5	Agent 6	Agent 7	Agent 8
Request 1	Wait(T)	Wait(T)	Process	Wait(T)	Process	Wait(T)	Wait(T)	Wait(T)
Request 2	Wait(T)	Process	X	Process	X	Wait(T)	Wait(T)	Wait(T)
Request 3	Process	X	X	X	X	Wait(T)	Wait(T)	Process
Request 4	X	X	X	X	X	Process	Process	X
Completion	X	X	X	X	X	X	X	X

The worst-case situation is when every time there are only two agents communicating, while all the other agents are waiting. In other words, there is only one request processed each time. In the calculation, there are $2N$ agents, which means that each request takes two agents for communication. Table 1 shows a demonstration of the worst case when four requests need to be processed. As we can calculate from (13), four requests require a waiting time of $12T$.

5.2.3. Memory consumption for storing agent information

Information sharing technology has been used in many current distributed systems such as Gnutella and BearShare [12,16]. Each node in such a decentralised framework carries a subset of the overall information of the system for searching and other purposes. Information sharing technology reduces redundancy in a decentralised system. Unfortunately, redundancy cannot be eliminated completely in decentralised systems because decentralised systems, particularly p2p networks, apply a high degree of redundancy to secure availability and fault tolerance [23].

In the case of implementing a decentralised topology in AOCD framework, the memory allocated to store agent information is: $\wp(J) \times R \times J$, where, $\wp(J)$ is the average number of agent information records carried by each agent. The reason for the above calculation is that in the decentralised topology, each agent carries a certain number of records of agent information, which is $\wp(J)$. The records of agent information carried by each agent are a subset of the overall records in the system. In other words, the total record number of agent information in the overall system without redundancy is J. These records are distributed to each agent redundantly and each agent is allocated $\wp(J)$ records on average. Therefore, the total memory allocated to store the agent information is the product of the above factors.

5.3. Hybrid agent network topology

Here, we introduce a novel hybrid framework that is different from the hybrid topologies introduced by Minar [2]. The proposed hybrid framework provides a structure combining centralised and decentralised topologies. As shown in Fig. 7, all the agents are connected to a central Matrix and each of them keeps connections with all the others agents. This hybrid topology is a centralised + mesh topology, in which agents are fully connected with each other. This topology reduces the workload of the Matrix and enhances manageability by using the Matrix as a central control component.

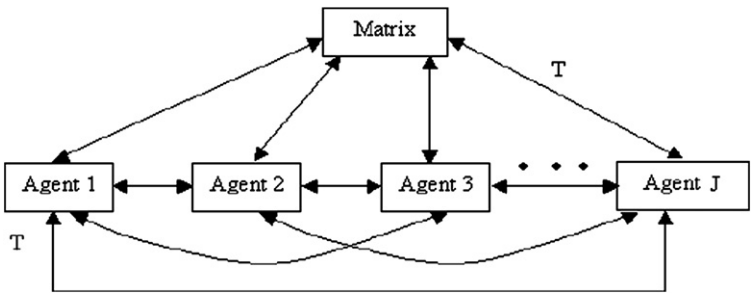

Fig. 7. Hybrid transmission framework.

5.3.1. Transmission time for a set of requests in short period

In the hybrid framework, communication involves the agents and the Matrix. The total transmission time for N requests is:

$$\lceil N/M \rceil \times 2T + T \leqslant TR \leqslant \lceil N/M \rceil \times 2T + N \times T. \tag{14}$$

The reason for the bound $\lceil N/M \rceil \times 2T + T$ is that in the best case, there are $\lceil N/M \rceil$ sets of requests. Each request set requires a transmission time of $2T$. T is incurred by the requesting agents sending requests to Matrix and another T is consumed by the Matrix delivering the requests to the corresponding agents. In the best case, all the requesting agents receive the results from the corresponding agents at the same time, which incur a time of T. Therefore, the total time in the best case includes the time for sending requests that is $\lceil N/M \rceil \times 2T$ and the time for receiving results costing T. The reason for $\lceil N/M \rceil \times 2T + N \times T$ is that the time for sending requests is $\lceil N/M \rceil \times 2T$ because all the requests are sent synchronously as mentioned in the assumptions. In the worst case, there is only one result transmitted over the network each time and there are N results to transmit for completing the whole transmission process. Therefore, the total time for transmission in the worst-ca situation is the sum of the time for sending the requests, which is $\lceil N/M \rceil \times 2T$, and the time for receiving the results, which is $N \times T$.

5.3.2. Total waiting time for completing a set of requests

Similar to the other two topologies, the waiting time in the hybrid topology includes (i) the waiting time for sending requests, and (ii) the waiting time for receiving results. Here, we have the total waiting time for completing a set of requests in the hybrid topology.

$$2T \sum_{i=1}^{\lceil N/M \rceil} i \leqslant W \leqslant 2T \sum_{i=1}^{\lceil N/M \rceil} i + T \sum_{i=1}^{N} (N - i). \tag{15}$$

The reason for $W \geqslant 2T \sum_{i=1}^{\lceil N/M \rceil} i$ is that N requests are delivered to the Matrix, which incur $2T \sum_{i=1}^{\lceil N/M \rceil} i$ time. In the best case, the corresponding agents send back the results to the requesting agent without delay, which means that the waiting time is 0. Therefore, the total time in the best case is: $2T \sum_{i=1}^{\lceil N/M \rceil} i + 0$.

The reason for the second inequality is that the time required for delivering requests to the corresponding agents is still the same as the best case. However, in the worst case, each time there is only one result delivered to the requesting agent and each request incurs a waiting time of T.

5.3.3. Memory consumption for storing agent information

In the AOCD architecture, the memory allocated to store the agent information in the hybrid topology is the same as in the centralised topology, namely, $R \times J$.

The reason for this result is that in the hybrid topology all of the agent information is stored in a central component. The other nodes (agents) will not keep the agent information. Therefore, the memory allocated to store the agent information is the size of each information record, which is R, multiplied by the number of total agents, which is J.

5.4. Overall performance analysis

We take $M = 3$ and the number of requests N is from 1 to 46 (step = 3). Figure 8 shows the total transmission time for the three topologies in the AOCD framework. TRC indicates the area that the centralised topology might incur the transmission time. TRD indicates the area that the decentralised topology might incur the transmission time. TRH indicates the area that the hybrid topology might incur the transmission time.

As shown in Fig. 8, the angle formed between the "best case" line and the "worst case" line in the centralised topology is ∂. According to Eq. (10), $\partial = \text{Arctan}((\lceil 2N/M \rceil \times 2T)/N) - \text{Arctan}((\lceil N/M \rceil \times 2T + N \times 2T)/N)$. The angle formed between the "best case" line and the "worst case" line in the hybrid topology is θ. According to Eq. (14), $\theta = \text{Arctan}((\lceil N/M \rceil \times 2T + N \times T)/N) - \text{Arctan}((\lceil N/M \rceil \times 2T + T)/N)$. The angle formed between the "best case" line and the "worst case" line in the decentralised topology is α. According to Eq. (12), $\alpha = \text{Arctan}((T + N \times T)/N) - \text{Arctan}(2T/N)$. When $N = 10$, the three angles are: (i) $\partial = 15.88°$, (ii) $\theta = 18.96°$, and (iii) $\alpha = 36.4°$. $\partial < \theta < \alpha$ indicates that the centralised topology is the most stable topology. However, the gradient of the "worst case" line of the centralised topology is far higher than the other lines, which reflects that the increase in the number

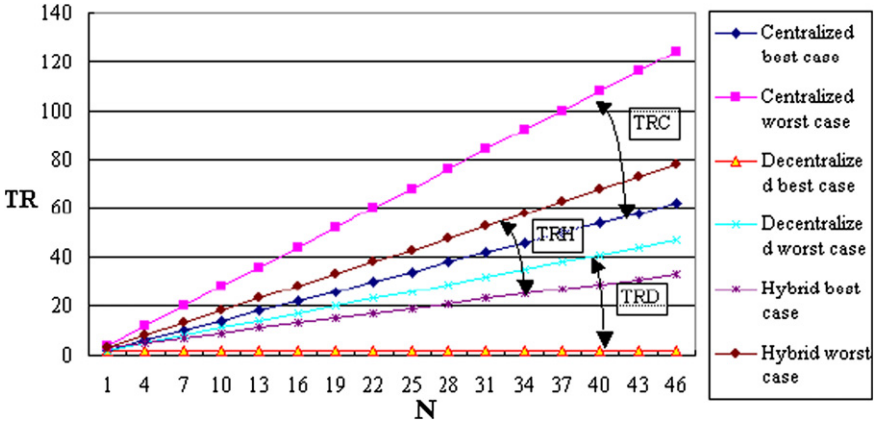

Fig. 8. Total transmission time for the three topologies.

of requests would cause much higher transmission time compared to the other two topologies. The best-case total transmission time for the decentralised topology is the most efficient among the six cases. However, the angle formed by the "worst case" line and the "best case" line is the greatest among the three topologies, which reflects this topology lacks stability.

Among the three topologies, the hybrid topology exhibits best performance. It has good stability and low total transmission time. The hybrid topology balances the weaknesses and advantages of the other two topologies. Figure 9 shows the total waiting time for the three topologies in AOCD framework. The parameters are $M = 3$, N is from 1 to 28.

WC indicates the area that the centralised topology might incur the waiting time; WD indicates the area that the decentralised topology might incur the waiting time; and WH indicates the area that the hybrid topology might incur the waiting time. As shown in Fig. 9, the decentralised topology exhibits its unstable performance because the WD area, which formed by the "worst case" line and the "best case" line of the decentralised topology, is the biggest among the three topologies. In other words, the waiting time in the decentralised topology can either be the highest or the lowest among all the cases, which reflects its instability. The centralised topology is the most stable one as it only contains a single case of total waiting time. The hybrid topology is more stable than the decentralised topology but less than the centralised topology. Nevertheless, the "best case" line of the hybrid topology is far below the "waiting time" line of the centralised topology and the "worst case" line of the hybrid topology is close to the "waiting time" line. This indicates that usually the consumption of the waiting time for a set of requests in the hybrid topology is lower than in the centralised topology.

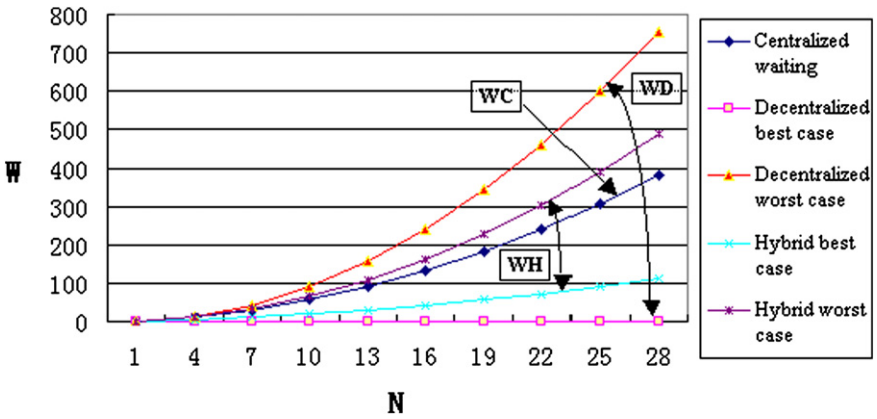

Fig. 9. Total waiting time for the three topologies.

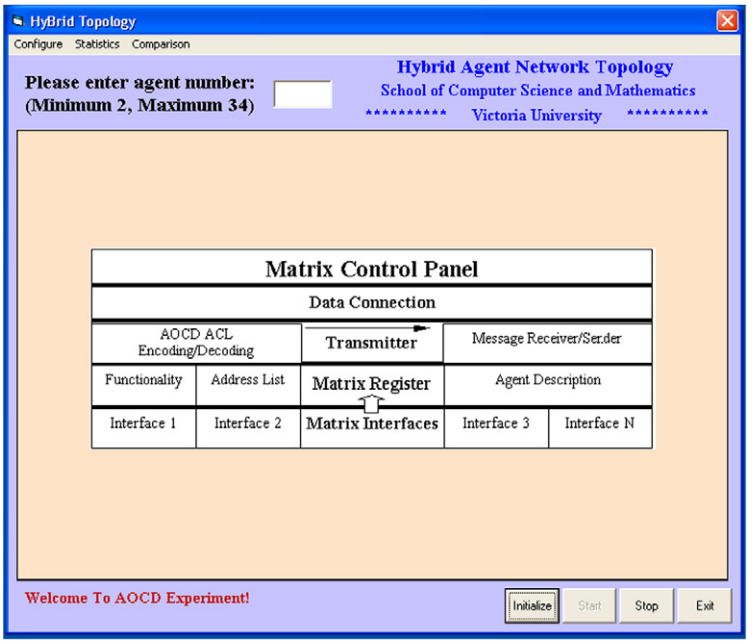

Fig. 10. AOCD topological experiment.

The memory consumption for agent information in the decentralised topology is highest among the three topologies in AOCD design. The hybrid topology and centralised topology have the same memory consumption volume, which is the number of records in the central control multiplied by the average record size. Therefore, the memory consumption volume for the decentralised topology is $\wp(J)$times of the other two topologies, where, $\wp(J)$ is the average size of the subset of the overall agent information in the system.

6. AOCD topological experiments

To validate our theoretical analysis, the authors developed a program, called *AOCD Topological Experimental Program*, based on the AOCD architecture. This is a simulation program that is designed to collect transmission data based on the centralised and hybrid agent network topologies. Figure 10 shows the main interface of the AOCD topological experiment program.

A user can enter an initial agent group number to start an AOCD topological experiment. The minimum number of user input is 2 and the maximum number of user input is 34. The reason we limit the agent group number in this experiment is that the limitation of the screen space, if there are more than 34 agent groups then the program will not be able to provide an overall demonstration on one screen. We enable users to adjust the agent number in an agent group so that the initial number of agents can be theoretically unlimited. Therefore, the agent group number is limited (for total of 34) but it does not limit the total agent number. In most of our experimental sessions we adopted 3 agents in one agent group.

6.1. Operational procedures of the AOCD topological experiment

The AOCD topological experiment is basically followed by five main procedures (for both hybrid and centralised designs). These five main procedures are listed as follows.

• *Initialising agent group*: A user starts this experiment program by entering the initial agent group number. Each agent group contains a number of agents. The agent number in each group is same and can be configured through the "Configuration Control" function.

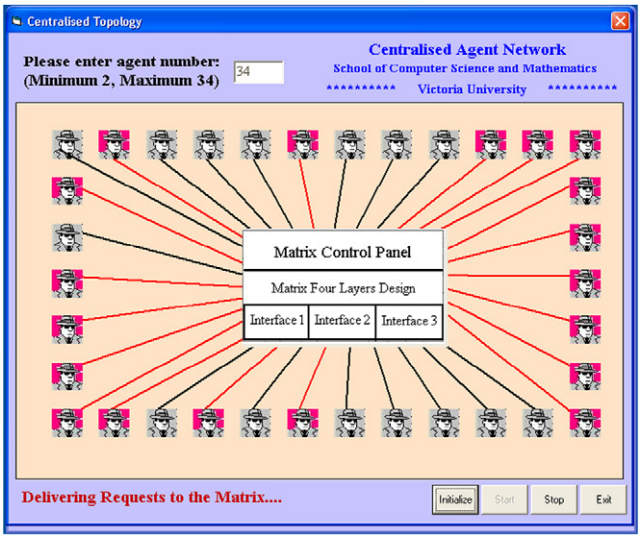

Fig. 11. Delivering agent request to Matrix.

• *Generating agents and requests*: After entering the agent group number, user can click the "Initialise" button. The system will automatically and randomly generate agents and requests. The generated agent information and requests will be stored into a database.

• *Delivering request to Matrix*: Once the generation of agents and requests is completed, the "Start" button will be activated. When user clicks the "Start" button, matching process will start. As shown in Fig. 11, a number of agent groups connect to Matrix. For an agent group that is sending requests to the Matrix, a red connection line will be used to mark the link between this agent and the Matrix. Meanwhile, the requesting agent's colour will also change to red background.

• *Agent matching process*: the agent matching processes for centralised and hybrid agent network topologies are different. As we have mentioned in Section 5, the centralised process involves in four transaction processes, which are (i) requesting agents to Matrix, (ii) Matrix to corresponding agents, (iii) corresponding agents to Matrix, and (iv) Matrix to requesting agents. The hybrid process involves in 3 transaction processes, which are (i) requesting agents to Matrix, (ii) Matrix to corresponding agents, and (iii) corresponding agents to requesting agents. The colour of the lines between the agent groups and the Matrix will turn to black when the requests of an agent group have been processed.

• *Completion of matching process*: The program will send a message once a matching process is completed. Figure 12 shows the screen of a completion of an agent matching process. All the connecting lines between agents and Matrix will turn into black when a matching process is completed. Whereas the background colours of the requesting agents still remain same in order to distinguish the requesting agents from other agents.

6.2. *Functionalities and configurations of the AOCD experimental program*

This experimental program is designed based on the AOCD architecture. The program provides some functionalities that allow user to configure the experimental settings.

• *Control commander*: Control Commander allows user to configure the experimental parameters including (i) Matrix Capacity, (ii) Agent number in a group, (iii) *Deadlock recovery* parameter, and (iv) Transmission interval. Figure 13 shows a screen of "Control Commander."

As mentioned in Section 5, a Matrix can handle a number of requests at any one time. *Matrix Capacity* indicates the number of requests that Matrix can handle at one time. This program also allows users to adjust agent number in an agent group and it enables more agents can be deployed in the experiment. The transmission interval parameter setting allows users to adjust transmission time of one transaction. In this experiment, we assume that all transactions have same transmission time. The default transmission interval is 1 second.

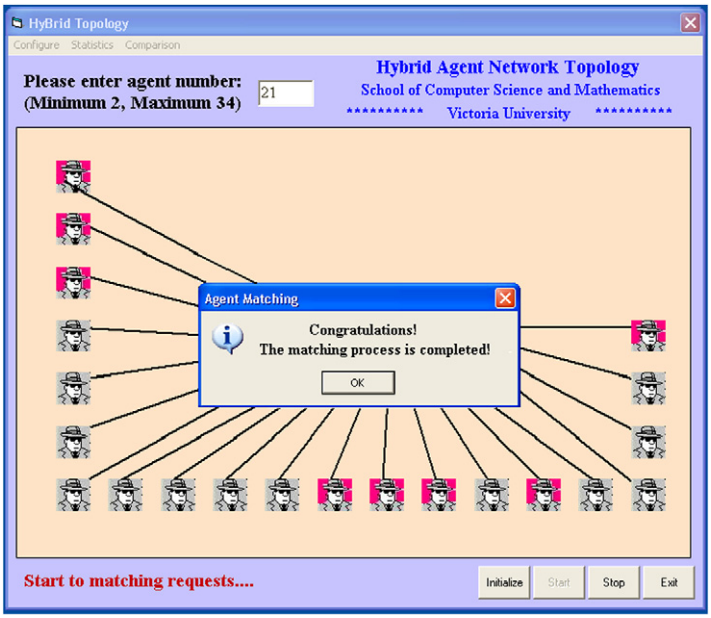

Fig. 12. Completion of agent matching process.

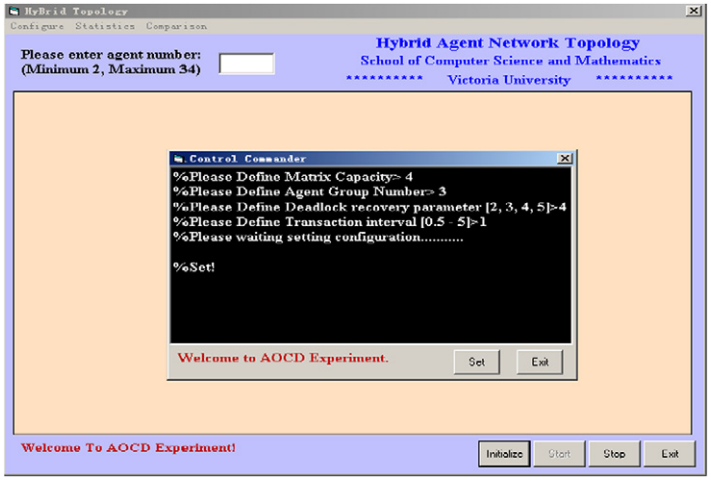

Fig. 13. Control commander.

The *Deadlock recovery* parameter is a unique design in this experiment. We observed that deadlocks occur when two agents request a same corresponding agent at same time or requests queue forms a loop when requesting agents waiting for other requesting agents. In this experiment, we use a tag to monitor the requesting queue. If an agent matching process runs through one round and the total requests in the queue do not decrease then the tag will be increased by 1. Once the tag value reaches to the value of the *Deadlock recovery* parameter then we force to release the first request in the queue and set the tag value to default. The deadlock recovery mechanism affects the success rate of AOCD topological experiment. The success rate is calculated through the following equation,

$$S = PR/TR, \tag{16}$$

where S denotes the success rate. PR denotes the processed requests. TR denotes the total requests that include the requests are not processed because of deadlock.

- *Stop button and agent information*:

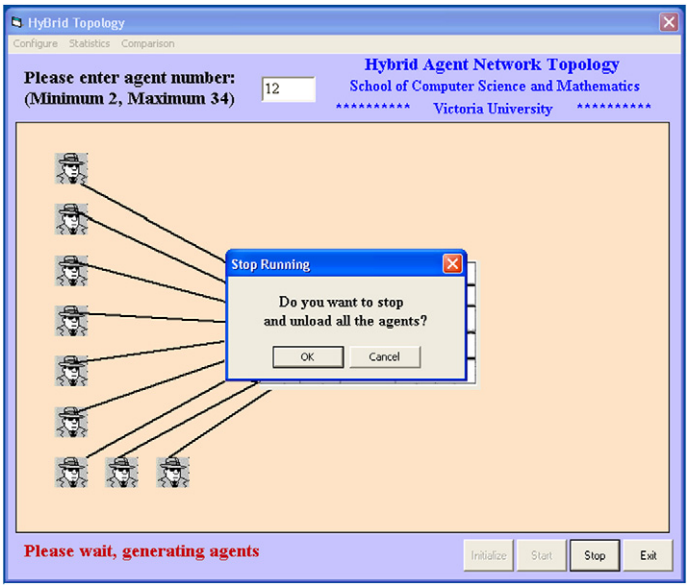

Fig. 14. Terminating matching process.

The "Stop" button allows users to terminate the program and unload all agents from Matrix. Figure 14 shows the termination of a matching process. User also can obtain agent information by double click on agents. The information includes agent number and requesting content (if the agent is requesting agent).

• *Progress indicator*:

We use a process indicator, as shown in Fig. 15, to inform the user about the current status of the experiment.

6.3. Design methodologies of the AOCD topological experiment

There are 9 major modules both in the hybrid and centralised agent matching process, which include:

- User Input Module.
- Request Generation Module.
- Randomisation Module.
- Connection Module.
- Matrix Control Module.
- Agent Matching Module.
- Requests Delivery Module (Both to/from Matrix).
- Waiting Queue Operation Module.
- Results Delivery Module.

As shown in Fig. 16, the structure of AOCD topological experiment program consists of 9 modules. Both centralised and hybrid designs have the same structure. In centralised agent topology, the results delivery module is called 4 times whereas this module is called 3 times in hybrid agent topology.

Another major difference between the centralised and hybrid design is in the results sending process. In the hybrid design, the corresponding agents will be released after they send their results. In the centralised design, the corresponding agents will not be released until the requesting agents receive the results from the Matrix. The reason for this design is that: in the hybrid agent topology, the results are directly delivered between the requesting agents and the corresponding agents. In other words, once the corresponding agents send their results means the requesting agents is going to receive the results and the process is over. On the contrary, the results in the centralised agent topology are delivered by the Matrix. In other words, the corresponding agents must not be released until the Matrix delivers the results to the requesting agents, which is to ensure the system consistency.

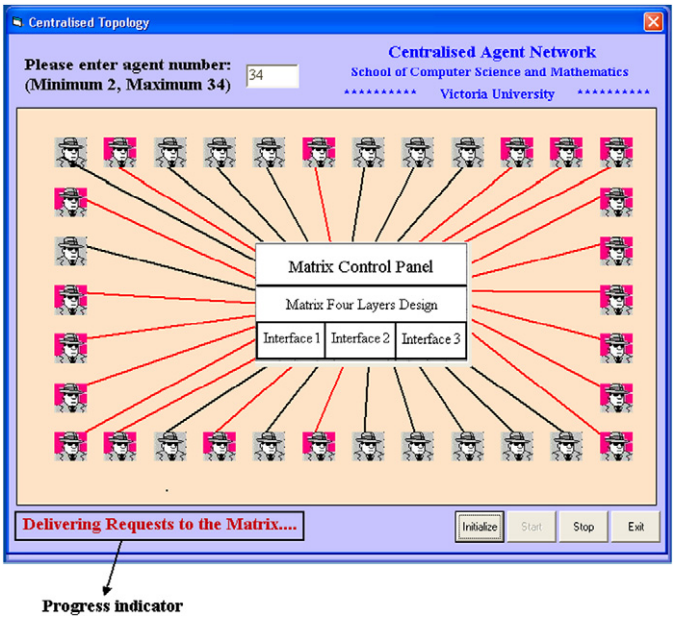

Fig. 15. Progress indicator.

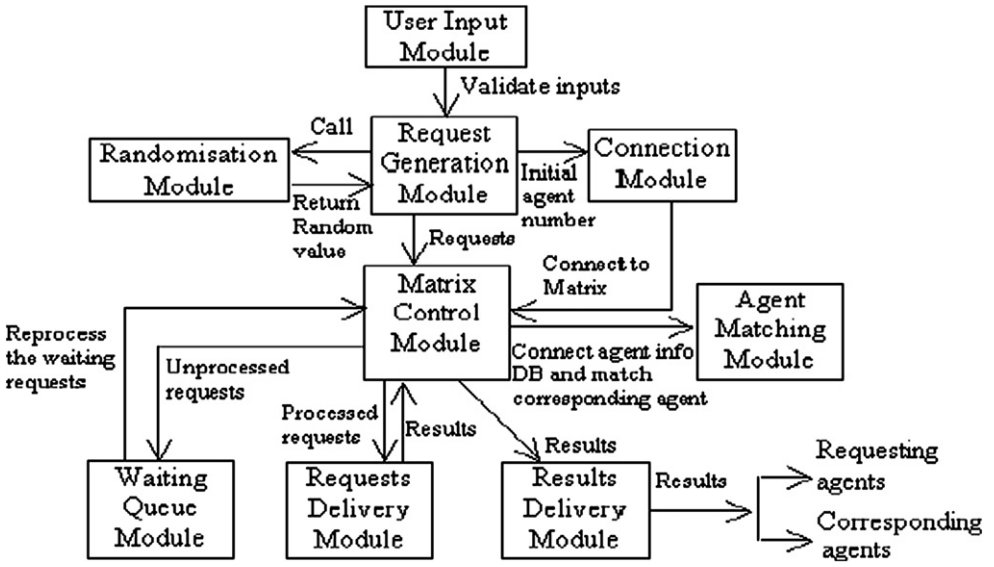

Fig. 16. Program structure of the AOCD topological experiment.

To explain our design methodology, we illustrate a section of pseudocode in agent matching module, which is one of the core modules of the AOCD topological experiment.

Pseudocode of the *agent-matching* module:

```
1 READ Processing_Requests_File
2 IF Requests_Number > 0 THEN //Requests are not empty
3    CALL Stop_Sending_RequestsTo_Matrix
4    DO LOOP UNTIL Request_Number = 0
5       FOR X = 0 to Matrix_Capacity − 1
```

```
6        IF Corresponding_ID = RequestID THEN
7        IF Corresponding_availability = 'available' THEN
8          //**Update Requests Records
9          SET Request_ProcessValue = 'True'
10         SET Request_SuccessValue = 'True'
11         SET Request_Status = 'Processed'
12       ELSE //** Corresponding agent unavailable.
13         ADD RequestID to Waiting_Queue
14         SET Request_ProcessValue = 'False'
15         SET Request_SuccessValue = 'False'
16         SET Request_Status = 'Unprocessed'
17       END IF
18       ELSE //** Not found Corresponding agent
19         //** Update Requests Records
20         SET Request_ProcessValue = 'True'
21         SET Request_SuccessValue = 'False'
22         SET Request_Status = 'Processed'
23       END IF
24       Request_Number = Request_Number − 1
25       IF Request_Number = 0 THEN EXIT FOR
26     END FOR
27   END LOOP
28 END IF
29 Call Matrix_Control_Module // Matching process completed
```

7. Results of the AOCD topological experiments

We conducted a set of AOCD topological experiments to observe the transmission performance of hybrid and centralised agent networks. Our experiments based on the centralised and hybrid agent systems. We conducted 100 sets of experiments for each system. When all the randomly generated requests (according to the agent group number) are matched then one set of the experiment is completed.

Our experiments were conducted in three different computing systems. (1) MS Windows XP (SP1), CPU PIII 700 MHZ; (2) MS Windows Server 2003, CPU AMD 900 MHZ; and (3) MS Windows XP (SP1), CPU Celeron 1.5 GHZ.

The pre-set values of the experimental parameters are listed as follows:

- Matrix Capacity: 3.
- Agent number in a group: 3.
- Transmission interval: 1 second.
- Deadlock recovery parameter: 4 or 3.

7.1. Results of the hybrid agent network

There are total 2406 requests were produced by our hybrid agent program in 100 sets of hybrid experiments and the total time consumption for processing these 2406 requests is 8200 seconds. The average time consumption for one request in the hybrid system is 3.408 seconds. The total participated agent number is 5160 and the average success rate for these 100 sets is 0.9545.

In the hybrid agent network, processing time is affected by the success rate as shown in Fig. 17. We calculate the average processing time for each matching process and measured by its success rate. We found, when the success rate of a matching process decreases then the average processing time of the matching process increases. In other words, if there is no deadlock in a matching process then the processing time will be much less than where there are some deadlocks. As shown in Fig. 18, the total request number also affects the processing time. The processing time

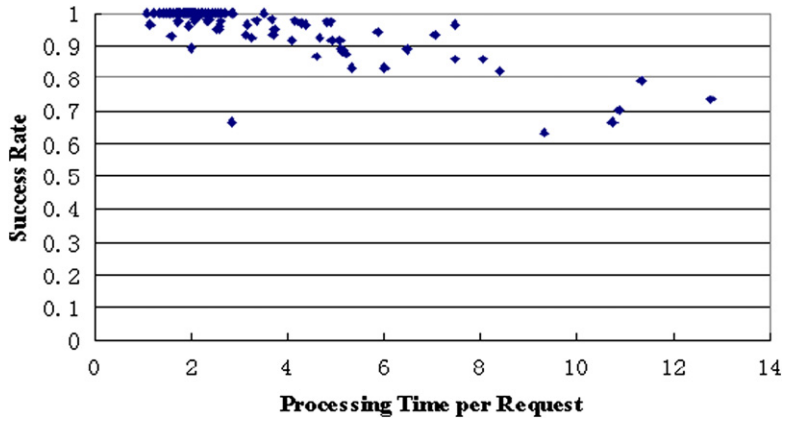

Fig. 17. Average processing time in one set affected by success rate in hybrid agent network.

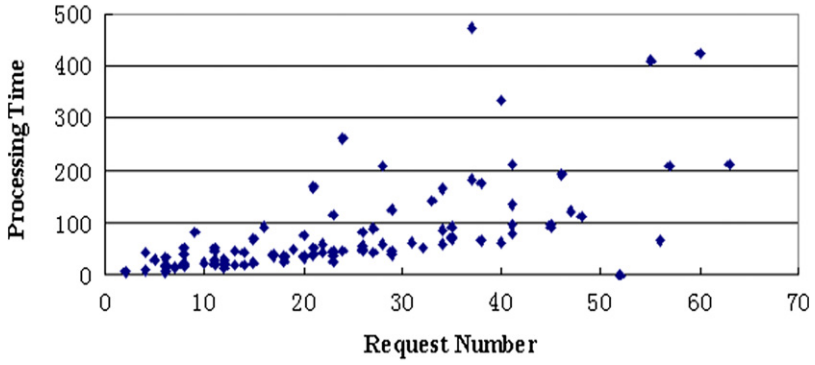

Fig. 18. Processing time affected by request number in hybrid agent network.

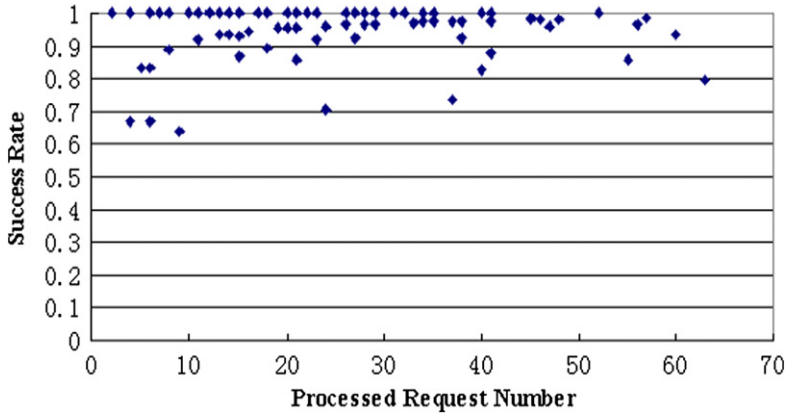

Fig. 19. Processed request number and success rate in hybrid agent network.

increases when the total request number increases. However, the total processed request number does not affect the success rate, as shown in Fig. 19, and the distributions of the success rates are relatively even among the different request numbers.

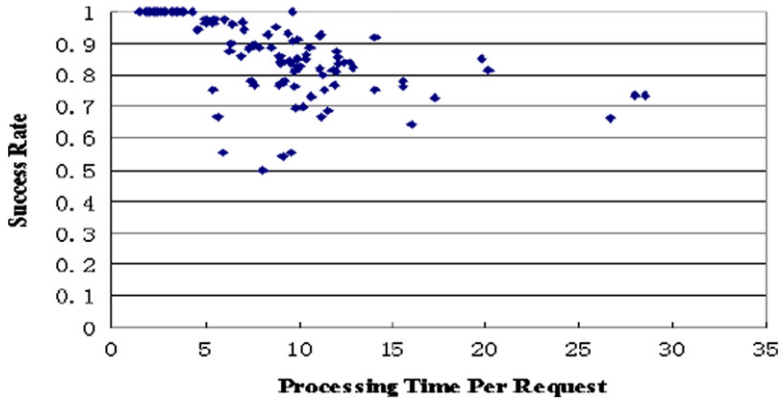

Fig. 20. Average processing time in one set affected by success rate in centralised agent network.

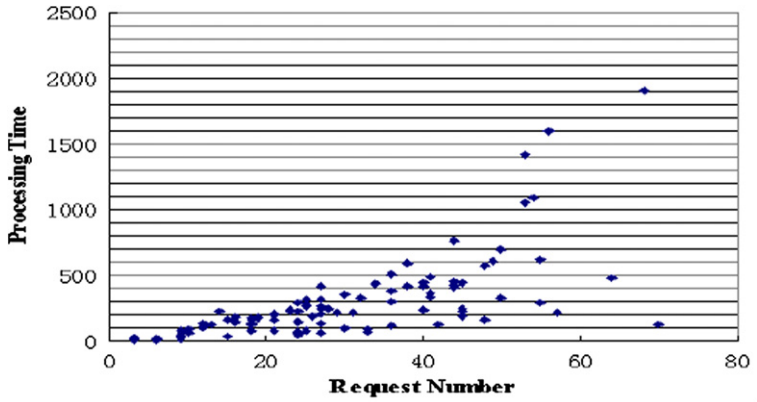

Fig. 21. Processing time affected by request number in centralised agent network.

7.2. Results of the centralised agent network

There are total 2917 requests were produced by our centralised agent network in 100 sets of centralised experiments and the total time consumption for processing these 2917 requests is 28477 seconds. The average time consumption for one request in the centralised system is 9.762 seconds. The total participated agent number is 5520 and the average success rate for the 100 sets of centralised experiments is 0.8674.

In the centralised agent network, we also discovered that average processing time is affected by success rate. When the success rate of a matching process decreases then the average processing time of the matching process increases. However, the influence of the success rate on the processing time in the centralised agent network is not as strong as it is in the hybrid agent network as shown in Fig. 20. In some cases, the success rates are low but their average processing time is not much. For instance, the average processing time is 8 seconds when its success rate is 0.5.

As shown in Fig. 21, the processing time of the centralised agent network is also affected by the total request number. In general, more requests consume more processing time. However, there are some exceptions that high volume of requests consume little processing time because of its success rate is high. For example, the processing time of 70 requests in Fig. 21 is 131 seconds (its success rate is 1) compared to 68 requests consume 1908 seconds (its success rate is 0.735).

In centralised agent network, the processed request number also does not affect the success rate because the distributions of processing time based on different success rates are relatively even.

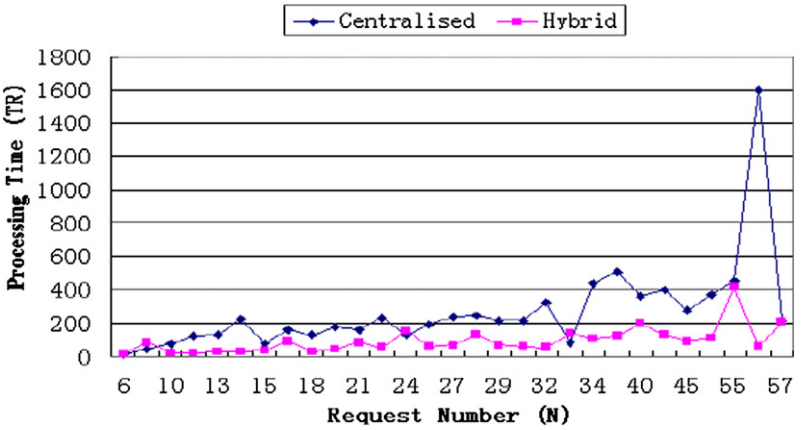

Fig. 22. Processing time for centralised and hybrid agent networks.

7.3. Centralised versus hybrid

The hybrid agent network presents a superior performance in AOCD frameworks compared to the centralised agent network. The average processing time for one request in the hybrid agent network is 3.408 seconds, whereas it is 9.762 seconds in the centralised agent network.

As mentioned in Section 5, "the gradient of the 'worst case' line of the centralised topology is far higher than the other lines, which reflects that the increase in the number of requests would cause much higher transmission time compared to the other two topologies." Although this conclusion based on the theoretical analysis, our experiments show this conclusion is accurate. Figure 22 shows, the processing time of the centralised agent network is above the processing time of the hybrid agent network on average. In Section 5, we also indicate that the transmission (processing) time in both centralised and hybrid networks are relatively stable and the centralised topology is slightly more stable than the hybrid topology. However, it seems hybrid topology is the most stable topology in Fig. 22. The reason of increasing hybrid topology's stability is that we use deadlock protection mechanism, which eliminates the negative side caused by deadlock (worst case). Nevertheless, both topologies have very close stability. Our experiments also prove that the theoretical analysis in Section 5 is generally accurate. Especially, when we compare Fig. 22 to Fig. 8.

Overall, our experiments based on the AOCD architecture show that the hybrid agent network topology has a superior performance than the centralised agent network topology.

8. Conclusion and future work

The topological aspect in agent network is an important but somehow underdeveloped research area. The significance of this paper is that it provides a systematic treatment in clarifying and organising the current topological structure in the intelligent agent field.

We provide a detailed performance analysis based on a novel agent-based architecture called *AOCD*. We believe the use of this architecture will enhance the efficiency of DSS. We have developed a program to validate our theoretical analysis of topological performance based on the AOCD framework. The significance of this research is: (i) it establishes a model for analysing agent network topologies and (ii) it provides a concrete methodology for meaningful performance analysis. In addition, this paper emphasises the importance of agent network topology analysis in multi-agent systems, which will help the practical development of such systems.

In order to support the topological analysis of agent network, we conducted a number of experiments based on the AOCD architecture. Through the experiments, we discovered that the hybrid agent network topology presents a superior performance in AOCD frameworks compared to the other two topologies. In general, the hybrid topology provides (i) stability in requests transmission, (ii) low memory consumption, and (iii) relatively low waiting time. The centralised topology is also shown to be superior in this analysis.

In future work, following issues can be considered:

- Developing an experimental system for decentralised agent network topology.
- Finding an efficient solution that could prevent the possible bottleneck problems, which might occur in the AOCD Matrix component.
- Analysing the feasibility of applying a Super-Node methodology in AOCD architecture, which could enhance the hybrid topology's fault-tolerance capability.

Acknowledgments

The authors wish to thank the facility staff in the School of Computer Science and Mathematics at Victoria University for providing experimental facilities and the referees for their comments.

References

[1] F.J. Martin, E. Plaza, J.A. Rodriguez-Aguilar, An infrastructure for agent-based systems: An interagent approach, Int. J. Intell. Systems 15 (2000) 217–240.
[2] N. Minar, Distributed system topologies: Part 1 and 2, http://www.openp2p.com/lpt/a/1461, 2002.
[3] T.C. Piliouras, Network Design: Management and Technical Perspective, second ed., CRC Press LLC, 2005, pp. 141–196.
[4] Fiorano Software Inc., Super-peer architectures for distributed computing, whitepapers, http://www.fiorano.com/whitepapers/superpeer.pdf, 2003.
[5] D.J. Watts, S.H. Strogatz, Collective dynamics of 'small world' networks, Nature 393 (1998) 440–442.
[6] D.J. Watts, Small Worlds: The Dynamics of Networks between Order and Randomness, Princeton University Press, 1999 (Part I).
[7] S.H. Strogatz, Exploring complex networks, Nature 410 (2001) 268–276.
[8] C. Aguirre, J. Martinez-Munoz, F. Corbacho, R. Huerta, Small-world topology for multi-agent collaboration, in: Proceedings of the 11th International Workshop on Database and Expert Systems Applications, 2000, pp. 231–235.
[9] X. Jin, J. Liu, Agent network topology and complexity, in: Proceedings of AAMAS '03, Australia, July 2003.
[10] S. Androutsellis-Theotokis, D. Spinellis, A survey of peer-to-peer content distribution technologies, in: ACM Comput. Surveys (CSUR), vol. 36, ACM Press, New York, 2004, pp. 335–371.
[11] A. Helsinger, R. Lazarus, W. Wright, J. Zinky, Tools and techniques for performance measurement of large distributed multiagent systems, in: Proceedings of AAMAS '03 Conference, Australia, 2003, pp. 843–850.
[12] P. Karwaczynski, J. Kwiatkowski, Analysis of overlay network impact on dependability, in: Proceedings of the 38th Hawaii International Conference on System Sciences, Hawaii, 2005.
[13] A. Gachet, P. Haettenschwiler, A decentralized approach to distributed decision support systems, J. Decision Systems 12 (2) (2003) 141–158.
[14] D. Kroenke, R. Hatch, Management Information Systems, McGraw–Hill, Watsonville, CA, 1994.
[15] J. Edwards, Peer-to-Peer Programming on Groove, Addison–Wesley Professional Press, 2002.
[16] OECD, Information technology outlook 2004: Peer-to-peer networks in OECD countries, in: OECD Information Technology Outlook, 2004 (Chapter 5).
[17] K. Truelove, A. Chasin, Morpheus out of the underworld, http://www.openp2p.com/pub/a/p2p/2001/07/02/morpheus.html, 2001.
[18] H.L. Zhang, C.H.C. Leung, G.K. Raikundalia, AOCD: A multi-agent based open architecture for decision support systems, in: Proceedings of International Conference on Computational Intelligence for Modelling, Control & Automation, Vienna, Austria, November 2005, pp. 295–300.
[19] P. Miller, M. Cummins, LAN Technologies Explained, Butterworth-Heinemann, USA, 2000, pp. 6–9.
[20] P. Erdós, A. Rényi, On the evolution of random graphs, Publ. Math. Inst. Hung. Acad. Sci. 5 (1959) 17–60.
[21] X.F. Wang, G. Chen, Complex networks: Small-world, scale-free and beyond, IEEE Circuits and Systems Magazine, 2003.
[22] A.-L. Barabási, R. Albert, Emergence of scaling in random networks, Science 286 (October 1999) 509–512.
[23] M. Parameswaran, A. Susarla, A.B. Whinston, P2P networking: An information-sharing alternative, IEEE Comput. 7 (34) (2001) 31–38.
[24] H.L. Zhang, C.H.C. Leung, G.K. Raikundalia, Classification of intelligent agent network topologies and a new topological description language for agent networks, in: Intelligent Information Processing III, in: IFIP Int. Fed. Inf. Process., vol. 228, Springer-Verlag, Boston, 2007, pp. 21–31.