Performance Analysis of Network Topologies in Agent-based Open Connectivity Architecture for DSS

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

School of Computer Science and Mathematics
Victoria University
PO Box 14428, Melbourne City MC 8001 Australia
haolan@sci.vu.edu.au, Clement.Leung@vu.edu.au, Gitesh.Raikundalia@vu.edu.au

Abstract

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 the specific multi-agent systems. In this paper, we propose a novel hybrid topology for distributed multi-agent systems, and compare the performance of this topology with two other common agent network topologies within the new multi-agent framework, *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.

1 Introduction

Application of Distributed Artificial Intelligence (DAI) theories and concepts to multi-agent systems has become common and efficient. The main reason for multi-agent systems inheriting DAI technologies is that multi-agent systems originally evolved from early DAI systems, which are based on distributed networks. Other forces have also been at work in driving multi-agent systems to become a major sub-discipline of DAI. For instance, many agent researchers come from a DAI background and they bring DAI technologies to multi-agent systems [1].

Some research works have been carried out to analyse the performance of network topologies in the DAI and network areas [2] [3] [4]. 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. The analysis enables system designers to compare the merit of different agent topologies, and select the most appropriate topology for their specific multi-agent-based systems.

The paper is organized as follows. The next section introduces related work that has been done in agent network topology and decentralised DSS areas. Section 3 describes the concept of AOCD, and in Section 4 we conduct a performance analysis of three common agent network topologies based on the AOCD architecture. Section 5 presents a summary of the overall performance results. In the final section we conclude our research work and consider future issues that need to be addressed.

2 Related Work

Gachet and Haettenschwiler [5] proposed a decentralised approach to Decision Support Systems (DSS). This approach overcomes the disadvantages of traditional DSSs and offers flexible, extendible, and mobile features to DSS. However, the major disadvantage in the Gachet and Haettenschwiler's framework is the lack of manageability, which is caused by removing central control [6]. Concurrent control and synchronicity problems are the main difficulties that hold back decentralised systems' manageability. In recent years, the hybrid topology began to be used in distributed systems like those described in Groove [7], KaZaa [8], and Morpheus [9]. The successful performance of these systems inspires the use of the 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 [10]. The AOCD architecture makes use of a unique component, the *Matrix*, to enhance the central control capability. AOCD design eliminates concurrent control and synchronicity problems that plague many decentralised systems.

Minar [2] introduces three basic network topologies: centralised, decentralised and hybrid topologies. In Minar's hybrid topology, there are three major forms: (*i*) centralised + centralised, (*ii*) centralised + ring, and (*iii*) centralised + decentralised (partial connection). The hybrid topology in AOCD design is different from Minar's methods and adopts centralised + mesh topology (fully connected). Centralised + mesh topology is an efficient but costly solution. Nevertheless, existing technologies are able to reduce costs by using message passing or SuperNode [11] mechanisms.

Proceedings of the 20th International Conference on Advanced Information Networking and Applications (AINA'06)
1550-445X/06 $20.00 © 2006 IEEE

The analysis of network topologies has been carried out in many disciplines. For instance, *(i)* cost calculation methodologies for wide area network design [3] has been used for network design estimation; *(ii)* performance analysis of distributed systems has been carried out by Spinellis and Androutsellis-Theotokis [12] and Helsinger *et al.* [13]; *(iii)* analysis of network topologies impact on dependability offers an evaluation method for estimating the effect of topology on dependability [14]; *(iv)* The description of the seven evaluation properties for analysing distributed systems topologies in Minar [2]. However, the current topology analysis based on multi-agent systems is inadequate.

3 Agent-based Open Connectivity for Decision support systems (AOCD)

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

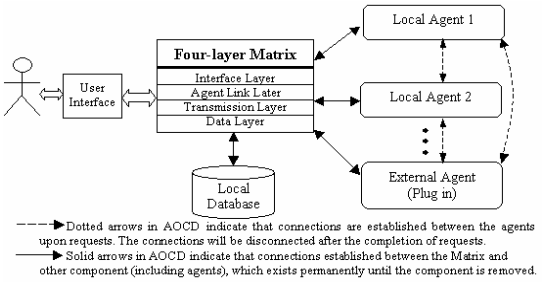


Figure 1. A conceptual view of AOCD architecture

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 standardized and independent of the environment. In other words, an AOCD system may vary according to the different business processes used by different organizations; however the Matrices used in these different systems are standardized.

4 Performance Analysis of 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 [5]: 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 the role of centralised coordinator 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.

4.1 Centralised Topology

In the centralised topology, as Figure 2 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.

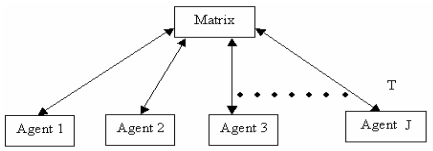


Figure 2. Centralised transmission framework

4.1.1 The transmission time for a set of requests in a short period of time

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

$$\lceil 2N/M \rceil \times 2T \le TR \le \lceil N/M \rceil \times 2T + N \times 2T \qquad (1)$$

The reason for $TR \ge \lceil 2N/M \rceil \times 2T$ 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 2*N* 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, 2*N* 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 2*T* to process one set of requests, which include the requests from the requesting agent costing 1*T* and the results from corresponding agents costing 1*T*. 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 $TR \le \lceil N/M \rceil \times 2T + N \times 2T$ 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 $N \times 2T$.

4.1.2 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.

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.

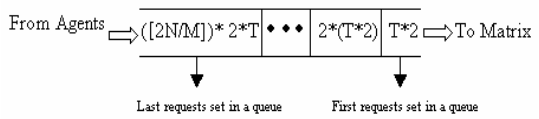


Figure 3. Waiting time in a queue

The total waiting time for all request sets in the queue is: $2T \sum_{i=1}^{\lceil 2N/M \rceil} i$, (2)

where $N > M, M > 0$. 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 (2), we find 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.

4.1.3 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.

4.2 Decentralised Topology

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 Figure 4 shows, the decentralised topology provides direct communication between agents.

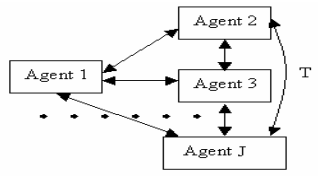


Figure 4. Decentralised transmission framework

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.

4.2.1 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:

$$2T \le TR \le T + N \times T \quad , \qquad (3)$$

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.

4.2.2 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:

$$0 \le W \le \sum_{i=1}^{N} (2N - 2i) \quad , \qquad (4)$$

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.

The worst-case situation is: every time there are only two agents communicating, 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 4 requests need to be processed.

Table 1. Total waiting time for 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

As we can calculate from (4), four requests require a waiting time of 12 T.

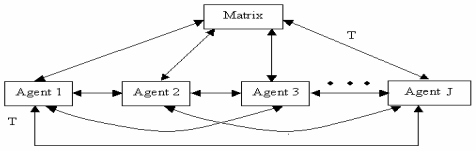
4.2.3 Memory consumption for storing agent information

Information sharing technology has been used in many current distributed systems such as Gnutella and BearShare [8] [14]. 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 [15]. 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.

4.3 Hybrid 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 Figure 5, 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.

Figure 5. Hybrid transmission framework



4.3.1 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:

$$\lceil N/M \rceil \times 2T + T \le TR \le \lceil N/M \rceil \times 2T + N \times T \quad (5)$$

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 (Session 4). 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 $N \times T$.

4.3.2 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

1550-445X/06 $20.00 © 2006 IEEE

results. Here, we have the total waiting time for completing a set of requests in the hybrid topology.

$$2T\sum_{i=1}^{\lceil N/M \rceil} i \le W \le 2T\sum_{i=1}^{\lceil N/M \rceil} i + T\sum_{i=1}^{N}(N-i) \quad (6)$$

The reason for $W \ge 2T\sum_{i=1}^{\lceil N/M \rceil} i$ 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: $2T\sum_{i=1}^{\lceil N/M \rceil} + 0$.

The reason for $W \le 2T\sum_{i=1}^{\lceil N/M \rceil} i + T\sum_{i=1}^{N}(N-i)$ 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.

4.3.3 Memory consumption for storing agent information

In the AOCD architecture, the memory allocated to store the agent information in the hybrid topology is 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 Conclusion and Future Work

The performance analysis of agent network topologies presented in this paper is based on the AOCD architecture. 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 emphasizes the importance of agent network topology analysis in multi-agent systems, which will help the practical development of multi-agent systems.

We also find that the hybrid 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. Centralised topology is also shown to be superior in this analysis. However, some researches [2] [11] show that the disadvantages of centralised topology such as lack of fault-tolerant and inefficient extensibility, limit its efficiency in distributed systems.

In future work, following issues can be considered:

- Finding an efficient mechanism to improve the cost-efficiency of centralised + mesh 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.

6 References

1. R. A. Flores-Mendez, "Towards a Standardization of Multi-Agent System Frameworks", *Crossroads*, 1999, Vol. 5, Issue 4, 18-24. AMC Press, New York, USA.
2. N. Minar, "Distributed system topologies: Part 1 and 2", 2002. Retrieved on July 19, 2005 from http://www.openp2p.com/lpt/a/1461
3. T. C. Piliouras, "*Network Design: Management and Technical Perspective"* (2nd ed.), 141-196. Published by CRC Press LLC, Printed in U.S.A. 2005.
4. S. J. Habib, "Simulated Analysis of Server Placement on Network Topology Designs", *Proceedings of The 3rd ACS/IEEE International Conference on Computer Systems and Applications*, Cairo, Egypt, 2005, p. 80-87.
5. A. Gachet, & P. Haettenschwiler, "A Decentralised Approach to Distributed Decision Support Systems", *Journal of Decision Systems,* 12 (2), 2003, p.141-158.
6. D. Kroenke, & R. Hatch, "*Management Information Systems",* Published by McGraw-Hill, Watsonville, CA, USA. 1994.
7. J. Edwards, "*Peer-to-Peer Programming on Groove",* Published by Addison Wesley Professional Press, 2002.
8. OECD. "Information technology outlook 2004: Peer to peer networks in OECD countries", *OECD Information Technology Outlook2004*, Chapter 5. 2004.
9. K. Truelove & A. Chasin, "Morpheus Out of the Underworld", 2001, Retrieved August 10, 2005, from http://www.openp2p.com/pub/a/p2p/2001/07/02/morpheus.html
10. H. L. Zhang, C. H. C. Leung, & G. K. Raikundalia, "AOCD: A Multi-agent Based Open Architecture for Decision Support Systems", *Proceedings of International Conference on Computational Intelligence for Modelling, Control & Automation – CIMCA 2005*, Vienna, Austria, November 2005.
11. Fiorano Software Inc. "Super-Peer Architectures for Distributed Computing", *Whitepapers,* 2003. Retrieved 08/2005, http://www.fiorano.com/whitepapers/superpeer.pdf
12. S. Androutsellis-Theotokis, & D. Spinellis, "A Survey of Peer-to-Peer Content Distribution Technologies", *ACM Computing Surveys (CSUR)*, Vol. 36, 335-371. ACM Press New York, U.S.A, 2004.
13. A. Helsinger, R. Lazarus, W. Wright, & J. Zinky, "Tools and Techniques for Performance Measurement of Large Distributed Multiagent Systems", *Proceedings of AAMAS'03 Conference*, Australia. pp. 843-850.
14. P. Karwaczynski, & J. Kwiatkowski, "Analysis of Overlay Network Impact on Dependability", Proceedings of the 38th Hawaii International Conference on System Sciences. Hawaii, 2005.
15. M. Parameswaran, A. Susarla, & A. B. Whinston, "P2P Networking: An Information-Sharing Alternative", *IEEE Computer*, 2001, Vol. 34, Issue 7. 31-38.