Service Composition Using Efficient Multi-agents in Cloud Computing Environment

Abhijit Bastia, Manoranjan Parhi, B.K. Pattanayak and M.R. Patra

Abstract In today's Internet world, end users need everything as a service (EaaS) being available to them throughout the world. Keeping users in the centre, developers are emplaning cloud computing environment which can satisfy needs of users virtually. Cloud computing is a collection of Web-accessible resources (i.e. Web services) that should be dynamically composed between service providers and brokers and virtualized based on consumer's needs on an on-demand basis. Mapping of the users' requirements should be done in an automated manner. But, distributed and constantly changing cloud computing environments pose new challenges to automated service composition such as: (i) dealing with incomplete information regarding cloud resources (e.g. location and providers), and (ii) dynamics of service providers, which set service fees on a supply-and-demand basis. To address these issues, we have proposed a multi-agent-based approach to compose services in multi-cloud environments for different types of cloud services: *one-time virtualized services, persistent virtualized services*, *vertical services*, and *horizontal services*. Cloud participants and resources are implemented and instantiated by agents. Previously the researchers have proposed self-organizing agents those make use of *Service Capability Table* and the *semi-recursive contract net protocol* (SR-CNP) to evolve and adapt cloud service compositions. To the existing work, we have planned to modify some of agents' behaviours, as a result we can reduce number of message passing by half in order to increase

A. Bastia (✉) · M. Parhi · B.K. Pattanayak
Department of Computer Science and Engineering, ITER, Siksha 'O' Anusandhan University, Bhubaneswar 751030, Odisha, India
e-mail: abhijit.bastia@gmail.com

M. Parhi
e-mail: mrparhi@gmail.com

B.K. Pattanayak
e-mail: bkp_iter@yahoo.co.in

M.R. Patra
Department of Computer Science, Berhampur University, Berhampur 760007, Odisha, India
e-mail: mrpatra12@gmail.com

L.C. Jain et al. (eds.), *Intelligent Computing, Communication and Devices*,
Advances in Intelligent Systems and Computing 308,
DOI 10.1007/978-81-322-2012-1_37

overall performance. Also we are planning a 2-layered (3 levels of multi-agents) self-organizing MAS that will establish a cloud service composition.

Keywords One-time virtualized services · Persistent virtualized services · Vertical services · Horizontal services · Agent behaviour · SR-CNP · SCT

1 Introduction

World Wide Web has become the most popular ammo in today's era of innovations and technologies. Cloud computing is an Internet-based computing, which typically involves the provisioning of dynamically scalable and often virtualized resources as services over the Internet. Some of the distinguishing characteristics of cloud computing are elasticity, scalability, hardware virtualization, fast service configuration, etc. [9, 10]. In cloud computing environments, variety of services can be provided which are broadly classified as infrastructure as a service (IaaS), platform as a service (PaaS), and software as a service (SaaS) as shown in Fig. 1. These cloud services can be composed into a value-added service termed as everything as a service (EaaS) to satisfy the dynamic needs of users. Various uncertainties can affect the correctness, availability, and reliability of service composition in cloud computing environments due to the heterogeneous, autonomous, and dynamic characteristics of cloud services. Therefore, how to ensure service compositions with multi-agents in the unpredictable cloud environments needs to be urgently addressed.

Now-a-days, the number of cloud providers is increasing, and the services offered by cloud providers have also increased. There are also increasing demands for cloud services from consumers. There are two major challenges to address. First, anticipating all of the possible required services is extremely difficult, particularly for software services. The second challenge is in selecting the optimum required single composite services, which are provided by different service providers with different quality of service (QoS) attributes; an optimal combination for forming a complicated service must be composed. Thus, there is a need for

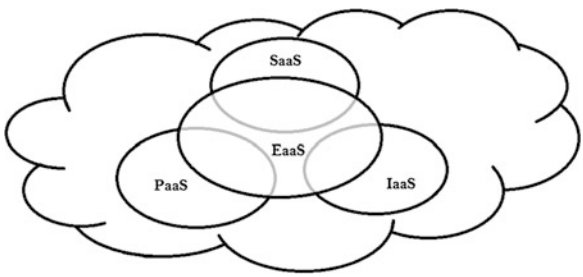

Fig. 1 Basic cloud service composition

dynamic and automated cloud service composition that can support everything as a service model capable of satisfying complex consumer requirements as they emerge and nullify human intervention by the use of agents. Literally, agent is an entity which acts on behalf of another entity. An agent is essentially a special software component that has autonomy that provides an interoperable interface to an arbitrary system and/or behaves like a human agent, working for some clients in pursuit of its own agenda. Agents are independent of problem solvers (e.g. cloud participants) that may collaborate to achieve a global objectives (e.g. service composition) while simultaneously considering both individual goals and constraints. Therefore, a multi-agent system is a collection of autonomous, interacting, and cooperative agents that react to events and may self-organize by means of interaction, negotiation, coordination, and collaboration.

The significance of this paper is that it is an effort in providing an efficient multi-agent-based approach for dealing with one-time, persistent, vertical, and horizontal cloud service compositions. This paper modernized some aspects of [1]. The rest part of this paper is distributed as follows. In Sect. 2, a literature study on cloud service composition model is discussed along with their merits and limitations. In Sect. 3, an efficient multi-agent-based cloud service composition model is proposed, along with the description of agents' job (Sect. 3.1), algorithms for proposed agents' behaviour (Sect. 3.2). In Sect. 3.3, the service capability tables (SCTs) for proposed model is discussed and semi-recursive contract net protocol (SR-CNP) is discussed in Sect. 3.4. In Sect. 4, implementation of the proposed model is discussed along with evaluation of both the models from speculative point of view is described. In Sect. 5, the future direction of this model is discussed along with the concluding remarks.

2 Literature Study

Several researches have been done on cloud service composition by using various approaches. In our review, we have considered the study of those papers which are related to cloud service composition.

A multi-agents-based cloud service composition model was proposed in [1, 3, 8] which is a 3-layered architecture. The positive points in [1] are as follows: (i) it is a multi-agent system that nullified the human intervention in cloud service composition, (ii) the implementation of service ontology which can be semantic or syntactic (which is another research area), and (iii) the distributive architecture of the model. Along with the above advantages, the paper lacks in some aspects like (a) service provider agents are cluster of homogeneous type of resource agents, i.e. an SPA only holds RAs of one-type services from SaaS, PaaS, and IaaS, (b) unnecessary message passing to rejected agents.

A QoS-based service composition using state transition machine is proposed in [2]. Positive point in this paper is that it composes service considering all optimal service resources, but the drawback is that it incurs more communication overhead

to reach at a composite service. A QoS-based service composition considering network traffic is proposed in [4]. Though this paper considers optimal services available locally to incur less network propagation but it may lack in case of a heavy traffic arising in a particular locality resulting a bottleneck condition in a geographical area.

A QoS-based tree-pruning service composition is proposed in [5]. Though it is a QoS-based service composition technique but it is centralized approach. A QoS-based service composition based on service-level agreement is proposed in [7]. Here, agreement is done before delivering the services and the composite services are well tested. But the drawback is that a bottleneck situation may rise at cloud directory and also the testing result may vary for different trusted composition verifiers.

The basic aim of considering [1] is all its positivity as described earlier. Along with the above-discussed advantages, the paper lacks in some aspects like (a) service provider agents are cluster of homogeneous type of resource agents, i.e. an SPA only holds RAs of one-type services from SaaS, PaaS, and IaaS, (b) unnecessary message passing to rejected and/or to useless agents, and (c) inefficient implementation of consumer agent and broker agent levels which may give rise to delay. In our proposed model, we have tried to address the above issues.

3 Proposed Cloud Service Composition Model

In our proposed composition model, we have designed a 2-layered multi-agent-based cloud service composition model instead of a 3-layered model with an aim to reduce the processing time. The proposed model is given in Fig. 2. The service ontology considered here is a syntactic though semantic ontology can also be implementable as described in [6] which we will try to address in our future work. We have taken 3 agents namely (i) consumer agent (CA), (ii) service broker agent (SBA), and (iii) resource provider agent (RPA). Also we have taken an SBA of heterogeneous type, i.e. the SBA is a cluster of different types of services (i.e. RPAs). SBA will perform two tasks. Firstly, it will act as a service provider. Secondly, it will act as a broker agent if it fails to do the first task. The agents' behaviours are summarized in Table 1.

3.1 Agents' Job Description

The agents' job is defined from their functionality point of view, and the tasks that can be performed by agents are defined as their behaviour. Agents (CAs, SBAs, and RPAs) interact among each other to compose and manage cloud services by adopting diverse agent behaviours.

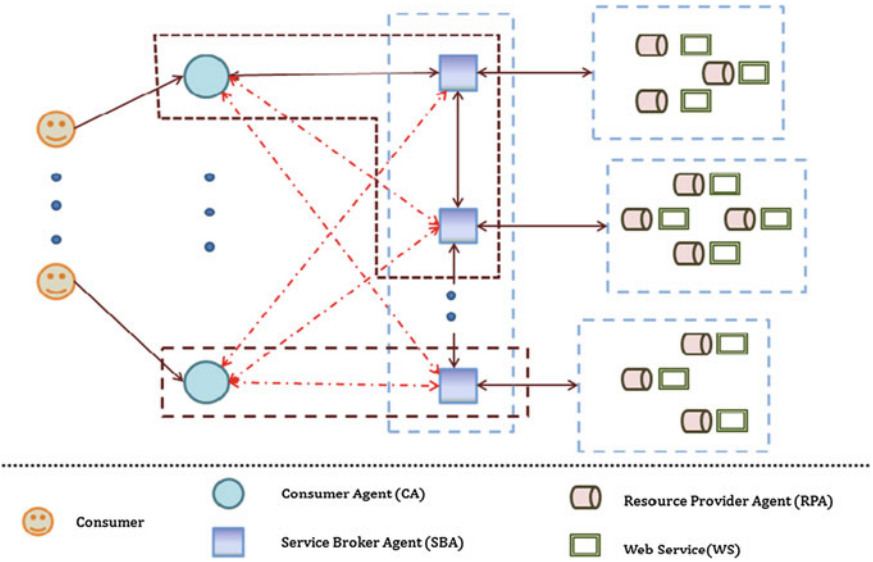

Fig. 2 Proposed cloud service composition model

Table 1 Agents' behaviour

Agent	Behaviour identifier	Main function
CA	*SR-CNPinitiatorCA*	To submit service composition call-for-proposals to SBAs
	ServiceAugmenterCA	To submit requests for incremental updates
	ServiceRevokerCA	To submit requests for subtractive updates
	ResultHandlerCA	To composite virtualized service
	ContractChangeMonitor	To receive expired contracts' notifications
SBA	*SR-CNPparticipantSBA*	To handle consumers' service composition requests from CAs or other SBAs
	SR-CNPinitiatorSBA	To submit call-for-proposals to resolve requirements to RPAs and service composition requests to other SBAs
	ReqAssignerSBA	To assign requirements to RPAs
	ResultHandlerSBA	To receive results from both RPAs and contracted SBAs
	IntermediarySBA	To handle requests to resolve requirements from RPAs
	ServiceRevokerSBA	To submit requests for subtractive updates
	ContractChangeMonitor	To receive expired contracts' notifications
RPA	*MainStructureRPA*	To resolve requirements by orchestrating a Web service
	ResourceHandlerRPA	To receive new web/cloud services and/or terminate web/cloud services
	RequesterRPA	To request external requirements to SBAs
	ReleaserRPA	To release cloud resources

a. *Consumer Agent Job*: Consumer agents are interfaces to remote users with the composition system. It also provides a single virtualized service to cloud consumers.

   i. Receiving and mapping consumer requirements.
   ii. Submitting service composition requests to SBAs.
   iii. Selecting an optimal SBA.
   iv. Handling (receiving) services and making single virtualized service.
   v. Submitting update requests.

b. *Service Broker Agent Job*: service broker agents manage cloud resources by controlling and organizing RPAs and interacting with other SBAs to accomplish composition if needed.

   i. Directing and delegating consumers' requirements to RPAs.
   ii. Keeping track of available resources.
   iii. Leasing cloud resources to CAs.
   iv. Managing parallel agent conversation.
   v. Execution of concurrent and parallel RPAs.

c. *Resource Provider Agent Job*: resource provider agents arrange Web services and control the access to them. RPAs receive requests to resolve requirements from service brokers. Then, RPAs handle the requests via their associated Web service, returning the output to the SBAs.

   i. Arrange Web services and control access to them.
   ii. Process received requests from SBAs.

3.2 Algorithm for Agent Behaviour

We have defined about sixteen algorithms for these agent behaviours. Two of these algorithms are given in Tables 2 and 3 for readers' interest and rest are not given for the implementation point of view.

3.3 Service Capability Tables

SCTs are used by agents to register and consult information about other cloud agents. The records of SCTs are composed of (i) agent address, (ii) requirements fulfilled by agent, and (iii) last known status of the agent/service. The SCTs are (a) *dynamic*: because agents can be removed or added to the table, (b) *exact*: because agents' address and functionalities are well known, and (c) *incomplete*: because agent may unaware of the full list of existing agents. The SCT parameters for different agents are tabled in Tables 4 and 5.

Table 2 Algorithm for SR-CNPinitiatorCA

SR-CNPinitiatorCA Behaviour:
Input: (i) Consumer Requirements, (ii) SBAs' address
Output: Single Virtualized service
1. creating atomic requirements
2. CA sends *call-for-proposals* to resolve requirements to m feasible SBAs' from SCT
3. if(ProposalReceived(Proposals,timeout1)) then
4. evaluate proposals 5. CA sends accept-proposal to *1 Best* SBA
6. create contract between CA and SBA
7. if(Receive(virtualized service,timeout2)) then
8. consume virtualized service
9. else
10. remove SBA and update status of SBA to *failed* in SCT
11. throw exception
12. else
13. update status of SBAs to unreachable in SCT
14. throw exception
Exception: Repeat steps again

3.4 Semi-recursive Contract Net Protocol

A problem whose solution can be obtained from the solution to smaller instances of the same problem is a recursive problem. A semi-recursive (i.e. to some extent recursive) agent interaction protocol is a protocol that attains its design objective (e.g. composing a set of cloud consumer requirements) by re-instantiating its communication pattern within itself to solve smaller instances of its design objective (e.g. composing a subset of the cloud consumer requirements). The SR-CNP follows a divide-and-conquer strategy. Agents in the CNP have two roles, i.e. initiator and participant. Here, CAs and SBAs can adopt the SR-CNP initiator role, but only SBAs and RAs can take the role of participant. Figure 3 shows the interaction of these agent behaviours among each other.

4 Implementation and Speculative Evaluation

To implement our framework, we have used Java Agent DEvelopment (JADE) platform. JADE is a very powerful middleware framework built with Java to design a multi-agent systems-based architecture. Consumer agents, service broker agents, and resource provider agents are created with JADE as shown in Figs. 4, 5, and 6. Figure 4 shows a JADE environment which consist of 3 proposed agents

Table 3 Algorithm for SR-CNP participant SBA

SR-CNPparticipantSBA Behaviour:
Input: (i) *call-for-proposals* from CAs to resolve Req
Output: (i) instantiation of *SR-CNPinitiatorSBA* Behaviour or (ii) *refuse* message
1. if(ProposalReceive(*call-for-proposals*(Req.))) then
2. reqRPA ← get requirements fulfilled by RPAs from Req
3. if(reqRPA == NULL) then
4. reqSBA ← get requirements fulfilled by other SBAs
5. end if
6. if(reqRPA ∨ reqSBA == Req) then
7. SBA sends *Proposal*
8. if(ProposalReceive(reply, timeout)) then
8. if(reply == accepted) then
9. if(reqRPA ! = NULL) then
10. contract RPA by SR-CNPinitiatorSBA
11. else if(reqSBA ! = NULL) then
12. contract SBA by SR-CNPinitiatorSBA
13. end if
14. end if
15. end if
16. else
17. SBA sends *refuse* message

Table 4 SCT for CA

About CA	About SBA
i. CAs' address	i. SBAs' address
ii. CAs' status (available, unreachable, and failed)	ii. SBAs' status (available, unreachable, and failed)
	iii. Requirements fulfilled by SBAs

Table 5 SCT of SBA

About SBA	About RPA
i. SBAs' address	i. RPAs' address
ii. SBAs' status (available, unreachable, and failed)	ii. RPAs' status (available, unreachable, and failed)
iii. Requirements fulfilled by SBAs	iii. Requirements fulfilled by RPAs

namely consumer agent, service broker agent, and resource provider agent with their agent IDs. Figure 5 shows a consumer interface which will give a composite service, and Fig. 6 shows a resource interface to register a resource.

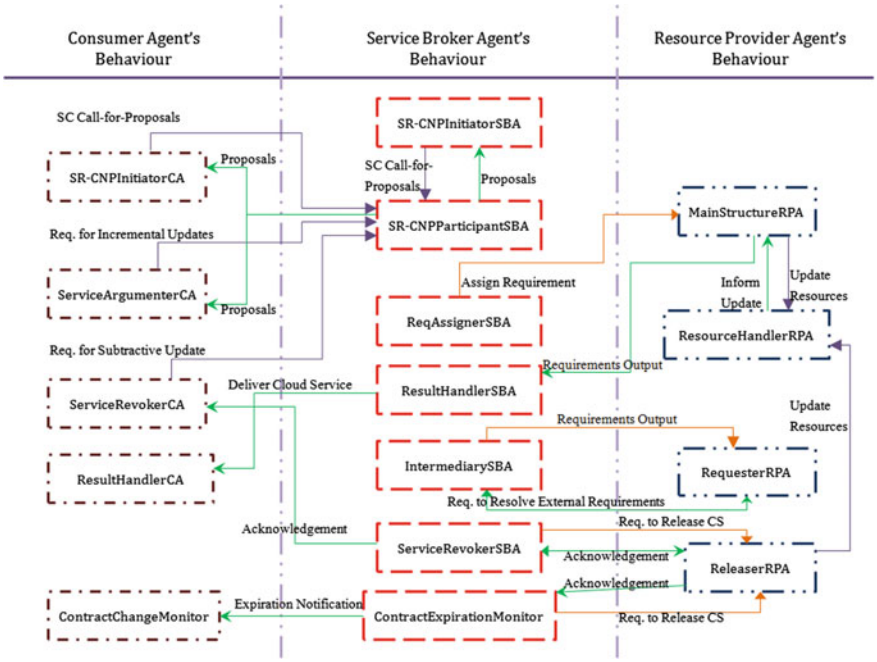

Fig. 3 Interaction of agent behaviours

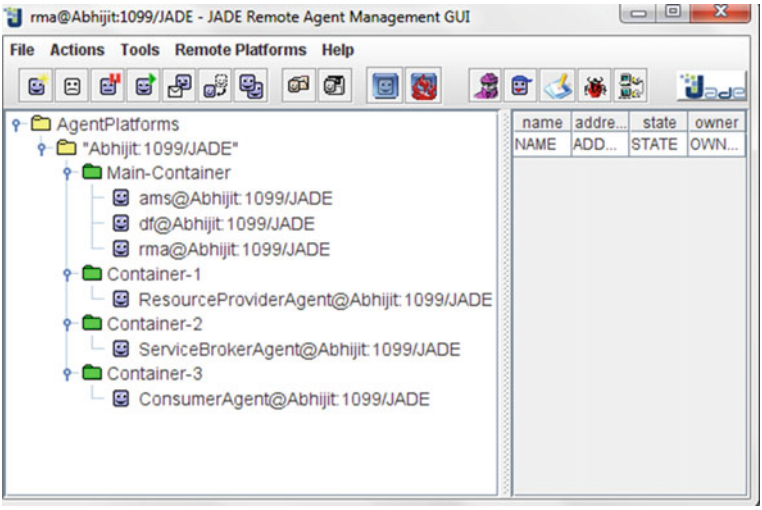

Fig. 4 JADE framework for proposed cloud service composition

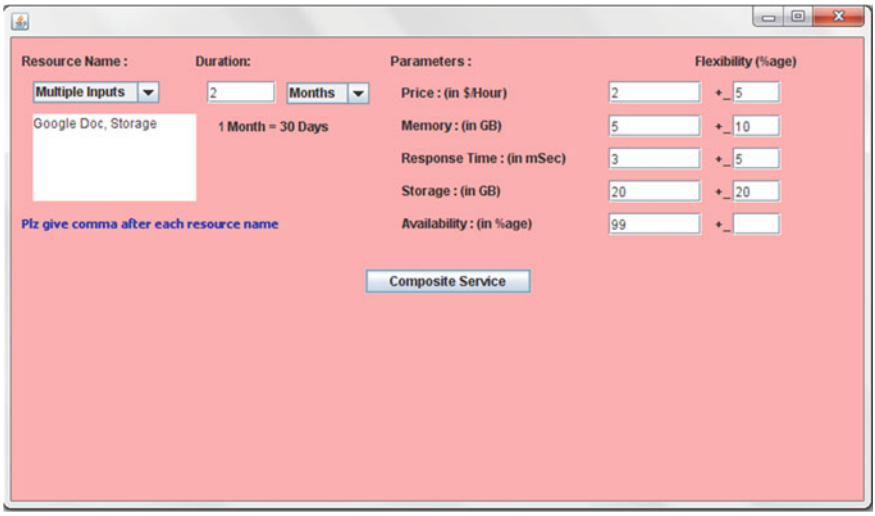

Fig. 5 Consumer agent interface

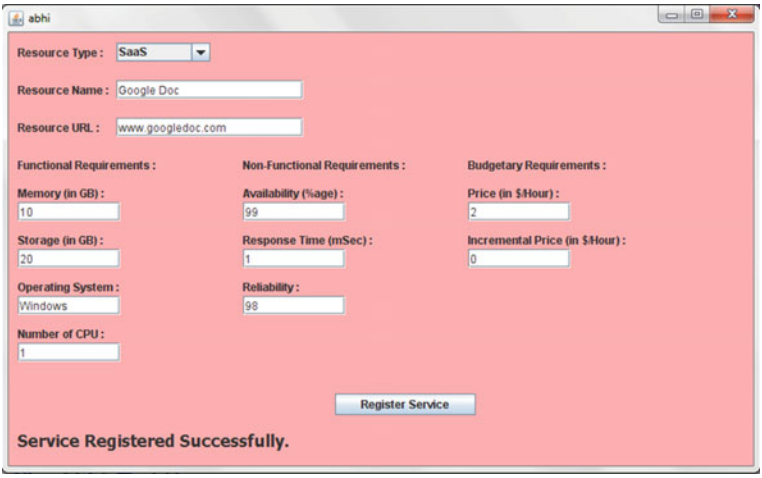

Fig. 6 Resource provider agent interface

To measure the complexity of agent behaviours, we have considered two performance measures which are number of messages exchanged by agents and processing time of request at each agent level. Section 4.1 gives a speculative evaluation on number of message exchanged, and in Sect. 4.2, the speculative evaluation on the processing of request is discussed.

Table 6 Comparative Speculative Evaluation

After getting the inputs, the CA sends q responses (1 accept-proposal message and $(q-1)$ reject-proposal messages). If the contracted SPAs fail, the failure is propagated to the BA. Then, the CA sends $(q-1)$ call-for-proposals messages to the remaining feasible BAs, and $(q-1)$ responses, and so on. Thus, in the worst case scenario the CA sends: $p(2q + 2(q-1) + 2(q-2) + \cdots + 2(2) + 2(1))$ $= 2p(q + (q-1) + (q-2) + \cdots + (2) + (1))$ $= 2p(q(q+1)/2)$ $= p * q(q+1)$ messages	After getting the inputs, the CA sends 1 response (1 accept-proposal message). (In this model, a timeout is set so that all other agents will consider themselves rejected if they do not get any accept-proposal within that timeout period.) If the contracted RPAs fail, the failure is propagated to the SBA. Then, the CA sends $(q-1)$ call-for-proposals messages to the remaining feasible SBAs, and so on. Thus, in the worst case scenario, the CA sends: $p((q+1) + ((q-1)+1) + ((q-2)+1) + \cdots)$ $= p((q+1) + q + (q-1) + (q-2) + \cdots)$ $= p((q+2)(q+1)/2)$ $= p * (q+2)(q+1)/2$ messages

4.1 Evaluation of Number of Message Exchanged

The number of message sent by an agent is dependent upon its connectivity. For both the models, we have considered similar test bed. The agents involved in evaluation were as follows: (i) for previous model, 25 CAs, 25 BAs, 30 SPAs, and 4,500 RAs (ii) for our proposed model, 25 CAs, 30 SBAs, and 4,500 RPAs. Both RAs and RPAs are randomly distributed to SPAs and SBAs, respectively. Weak connection has up to 33 % connectivity to next layer. Similarly, moderate has up to 66 % connectivity and strong has up to 100 % connectivity to next layer. The comparison shows a reduction of message passing by half through our proposed model which holds for each layer. A description on the speculative evaluation between the two approaches is discussed in Table 6. Figure 7 shows the evaluated number of message exchanged by previous model, and Fig. 8 shows the same for our proposed model.

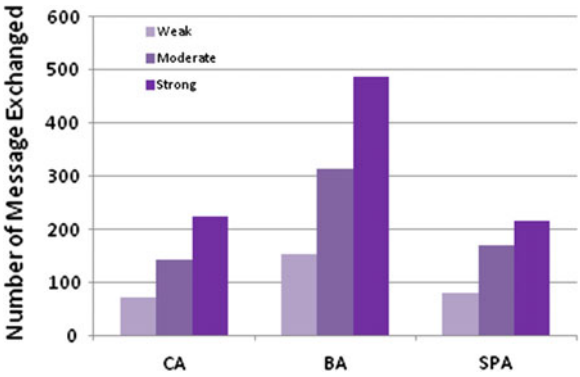

Fig. 7 Number of message exchanged by previous model

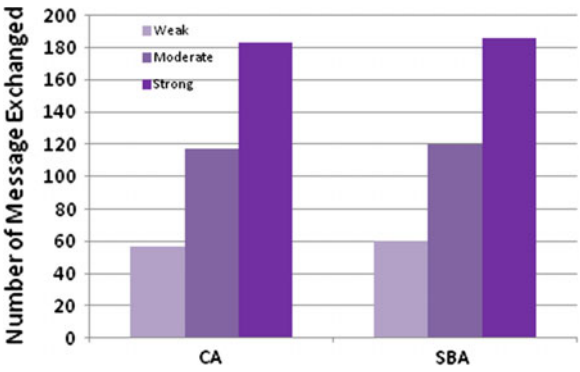

Fig. 8 Number of message exchanged by proposed model

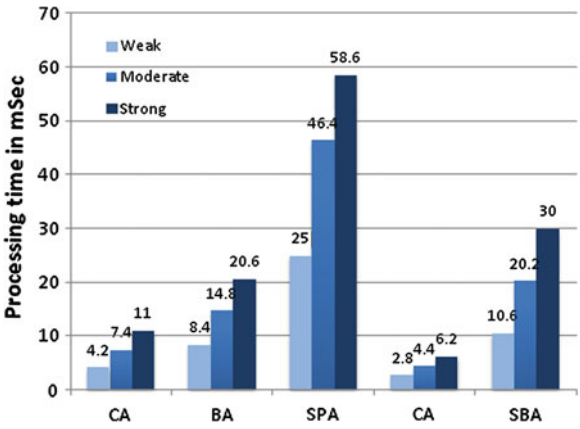

Fig. 9 Overall processing time of agents by both models

4.2 Evaluation of Processing Time of Request

To evaluate the processing time for each request, we have considered similar conditions, i.e. for sending a message 0.2 ms time is considered, and 1 ms time is considered for processing a request for both the models. Figure 9 shows the overall processing time needed to process single instance of a request by both the models, where CA, BA, and SPA belong to the previous model, and CA and SBA belong to our proposed model.

5 Conclusion and Future Work

In this paper, we have focused on demonstrating the effectiveness of adopting agent-based techniques for cloud service composition by implementing the desirable property that our agents can autonomously and successfully deal with changing service requirements through self-organization and collaboration. As an extension to the current work, we intend to incorporate the semantic ontology to our proposed model. We hope that our multi-agent-based framework can become more practical in real-world applications with full-phase implementation.

References

1. Gutierrez-Garcia, J.O., Sim, K.M.: Agent-based cloud service composition. Appl. Intell. doi:10.1007/s10489-012-0380-x, 5 Sept 2012
2. Liu, S., Xiong, G., Zhao, H., Dong, X., Yao, J.: Service composition execution optimization based on state transition matrix for cloud computing. In: Proceedings of the 10th World Congress on Intelligent Control and Automation, IEEE, Beijing, China, 6–8 July 2012

3. Gutierrez-Garcia, J.O, Sim, K.M.: Agent-based service composition in cloud computing. In: Proceedings of the 10th World Congress on Intelligent Control and Automation, IEEE, Beijing, China, 6–8 July 2012
4. Klein, A., Ishikawa, F., Honiden, S.: Towards network-aware service composition in the cloud. In: International World Wide Web Conference Committee, IW3C2, ACM, Lyon, France, 16–20 Apr 2012
5. Bao, H., Dou, W.: A QoS-aware service selection method for cloud service composition. In: 26th International Parallel and Distributed Processing Symposium Workshops and PhD Forum, IEEE, ISBN:978-0-7695-4676-6, 2012
6. Xiangbing, Z., Fang, M.: A semantics web service composition approach based on cloud computing. In: 4th International Conference on Computational and Information Sciences, IEEE; doi:10.1109/ICCIS.2012.43, 2012
7. Al Falasi, A., Serhani, M.A.: A framework for SLA-based cloud services verification and composition. In: International Conference on Innovations in Information Technology, IEEE, ISBN:978-1-4577-0314-0, 2011
8. Gutierrez-Garcia, J.O., Sim, K.M.: Self-organising agents for service composition in cloud computing. In: 2nd IEEE International Conference on Cloud Computing Technology and Science, doi:10.1109/CloudCom.2010.10, 2010
9. Zhang, Q., Cheng, L.: Cloud computing: state-of-the-art and research challenges. J. Internet Serv. Appl. **1**(1), 7–18 (2010)
10. Dillon, T., Chen, Wu., Chang E.: Cloud computing: issues and challenges. In: Proceedings of 24th IEEE International Conference on Advanced Information Networking and Application (AINA) pp. 27–33, Apr 2010