International Journal For Technological Research In Engineering
Volume 1, Issue 5, January – 2014 *ISSN (Online): 2347 - 4718*

A PTC APPROACH BASED CONCILIATION PROCESS FOR CLOUD SERVICE RESERVATION

T. Sanjeevakodi[1], R. Poonkodi[2], Dr. C. Kumar Charlie Paul[3]
Department of Computer Science, Anna University Chennai.
A.S.L Paul's College of Engineering & Technology, Coimbatore

***Abstract*: When we are going to make booking for Cloud services, then client and service providers have to establish service-level agreements through negotiation. Where it is important for both a client and a service provider to reach an agreement on the cost of a services when they are going to cloud services, to date, there is little or no negotiation support for both cost and time-slot negotiations (CTNs) for cloud service booking. This article presents a multi-issue negotiation method to facilitate the following: 1) CTNs between Cloud agents and 2) tradeoff between cost and time-slot utilities. Unlike many existing negotiation method in which a negotiation agent make only one proposal at a time. In our work, agent makes multiple proposals in a negotiation round that generate the same aggregated utility, differing only in terms of individual cost and time-slot utilities. Another novelty of work is formulating a novel time-slot utility function which characterizes preferences for different time slots. These ideas are implemented in an agent-based Cloud test bed. Our results show that CTN agents reach faster agreements and achieve higher utilities than other related approaches.**

***Keywords:* Automated negotiation, Cloud negotiation, Cloud resource allocation, multi-issue negotiation, negotiation agent, resource management.**

I. INTRODUCTION

Cloud is a parallel and distributed system consisting of a collection of interconnected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resource(s) based on service-level agreements (*SLAs*) established through negotiation between service providers and client [1]. Hence, a Cloud service provision is commonly governed by an *SLA* [2], [3]. An *SLA* is a service guarantee method which defines a set of quality of service (*QoS*) constraints such as cost or time constraints and specifies how the service is offered. To establish an agreement between a client and service provider for utilizing a Cloud service, some of the important issues include the following: 1) determining when to use a service (i.e., time slot) and 2) determining the cost of the service.

Even though these issues are more important, mechanisms to automate the negotiation of cost and time slot for Cloud services have not been devised. Whereas previous works have dealt with advance booking considering bandwidth or time constraints [4]–[6] and considered *SLA* negotiation [7], to date, there is no service reservation system that considers both cost and time-slot negotiations (*CTNs*). Since there is an inverse relationship between cost and time-slot utilities [e.g., a client needs to pay a higher cost (obtaining a lower cost utility) to use a service at a more desirable time slot (obtaining a higher time-slot utility)], cost and time slot have to be negotiated simultaneously. This work considers a multi-issue negotiation mechanism for *CTNs* for Cloud service booking.

In this paper, we present a novel time-slot mechanism which is designed to model the clients' and service providers' preferences for different time slots. In general, a client can have multiple sets of acceptable time-slot preferences. For example, using a Cloud service at around 2 P.M. is a client's most preferred choice, using the Cloud service at around 9 P.M. the second best choice, and using the Cloud service at 6 P.M. is the least preferred choice. A time-slot utility function consisting of multiple partial functions is used to model a client's preferences for multiple sets of acceptable time slots.

This work considers bilateral negotiation s between a client and service provider, where both agents are sensitive to time and adopt a time-dependent concession-making strategy for *CTNs*. Since both agents negotiate on both cost and time slot, generating a counterproposal can be making either a concession or a tradeoff between cost and time slot. Hence, an agent's strategy for multi-issue negotiation is implemented using both the following: 1) *Tradeoff Algorithm:* The novelty of this work is adopting a new tradeoff algorithm, called a "burst mode" proposal, which is designed to enhance both the negotiation speed and the aggregated utility. *2) Concession-Making Algorithm:* The concession-making algorithm determines the amount of concession for each negotiation round, which corresponds to the reduction in an agent's expected total utility.

We developed Agent-Based Cloud Testbed approach which provides simple service discovery functionality through message passing. The agent-based Cloud testbed is designed and implemented with the help of Java and the Java agent development (JADE) framework. In a Cloud market, there are many clients and service providers, and thus, the Cloud testbed has client agents and provider agents acting on behalf of clients and providers. A client agent broadcasts a message indicating the name of the Cloud service that the client needs to all provider agents, and a provider agent who has the service replies to the client. In this paper, our contribution of

works as follows:

1. We develop *CTNs* mechanism that includes the design of a novel utility function for time-slot preferences.
2. We design tradeoff and concession algorithms for the concession strategy between clients and providers.
3. We implement an agent-based Cloud testbed for evaluate the *CTC* mechanism.

The rest of the paper proceeds as follows: In Section I, we formally introduce the system model and the idea of Cost Price method for cloud services via leasing. In Section II, we discussed about related work for understanding the previous work. In section III, we presented proposed scheme and their framework for implementation of cost-price negotiations. . In section IV, we show implemented result through tables and screenshot. Finally, we conclude overall work in Section V. We provide the acknowledgment in Section VI and reference in section VII.

II. RELATED WORK

In existing systems, Multi-Issue SLA Negotiation approach was proposed by Czajkowski et al. [18] which is generalized resource management model in which resource interactions are mapped onto a well-defined set of platform-independent *SLAs* that formalizes agreements to deliver capability, perform activities, and bind activities to capabilities, respectively. The model is based on a Service Negotiation and Acquisition Protocol, which governs the lifetime management of *SLAs*. For *SLA* specifications, a Meta negotiation was proposed by Brandic *et al.* [19] to allow two parties to reach an agreement on what specific negotiation protocols, security standards, and documents to use before starting the actual negotiation. To manage Paschke *et al.* [20] proposed a declarative rule-based *SLA* (*RBSLA*) language for describing *SLAs* in a generic way. *RBSLA* is machine readable and has executable contract specifications. Whereas [18], [19], and [20] do not focus on specifying negotiation strategies nor designing utility functions for each negotiation term, Yan *et al.* [9] adopted a tradeoff algorithm for multi-issue *SLA* negotiations. Yan *et al.* [9] proposed a framework for a Web service composition that provides *SLA* negotiation for *QoS* constraints. Based on this framework, a utility-function-based decision-making model that supports coordinated negotiations was presented. To ensure coordinated *SLA* negotiations, the framework in [9] consists of a coordinator agent (*CA*) and a set of negotiation agents. The *CA* is responsible for governing the composition of the *SLA* negotiation as a whole, while each negotiation agent is in charge of the *SLA* negotiation for one service in the composition. The negotiation agent presented in [9] is related to the design of *CTN* mechanism. Similar to *CTN* mechanism, the design of the decision-making model of negotiation agent for the multi-issue negotiation in [9] includes the formulation of utility functions and a negotiation strategy.

The difference between [9] and this work is that, while agents in [9] generate a single proposal in each round using heuristics, *CTN* agents concurrently generate multiple proposals in each round. The time-slot utility functions in this paper enable client and service provider agents to express preferences for different time slots. Yan *et al.* [9] designed a concession-making algorithm and a tradeoff algorithm for generating counterproposals. The concession-making algorithm searches within an acceptance range for a point with a fixed deduction of the total utility value of the current offer to generate a new counterproposal. Whereas the concession algorithm in this work adopted the time-dependent strategies in [12], a fixed deduction is preconfigured by the *NA* in [9].

Finally, it should be noted that some of the preliminary results of this work were presented in [21]. This work has significantly and considerably augmented and generalized the preliminary work in [21] as follows. Whereas [21] omitted any details on the timeslot utility function, in this paper, a very detailed formulation of the time-slot function (which has not been previously considered by other related works) is provided- A. Moreover, being a short conference paper, [21] has omitted comparisons with related works. This paper provides a very detailed comparison between this work and related works on advance reservation, concurrent negotiation, *SLA* negotiation, and multi issue negotiation. Exchanging the resources at a service level at least equal to the minimum service level and for a price does not exceed the maximum price specified in the agreement.

III. PROPOSED SCHEME

In this section we implement our scheme and discuss about their feature follow as: in Section A, We introduce with cloud pricing through negotiation and show the advantages of this scheme. In section B, we developed cloud-based Testbed agent for communication client and service provider. In section C, we implement **Cost** and Time-Slot Negotiations with help for trade-off and Concession-making algorithm.

A. Cloud Pricing through Negotiation

One of the challenging issues in Cloud service booking is devising an appropriate pricing model. Amazon elastic Cloud computing (*EC2*) [22] provides clients with both fixed pricing (on-demand instances and reserved instances) and flexible pricing (spot instances) for leasing virtual machine (*VM*) instances. On-demand instances allow clients to pay a fixed cost by the hour without a long-term commitment and to start the instances immediately. With reserved instances, clients need to pay a one-time fee for a one- or three-year term but benefit from paying a discounted hourly usage fee within the term. Spot instances enable clients to bid for unused computing capacity.

Instances are charged at the spot cost set by Amazon. The spot cost changes periodically, depending on the supply and demand for spot instances. Clients' requests can be fulfilled

if their maximum bid costs are above the spot cost and they can run their applications on the spot instances for as long as their maximum bid costs exceed the current spot cost. All clients will pay the same spot cost for that period even if their maximum bid costs are above the spot cost. For example, if clients A's and B'smaximum bid costs are $0.7 and $0.72, respectively, and the spot cost is $0.67, then both A and B will pay $0.67.

The advantage of spot instances over the on-demand and reserved instances is that, by allowing clients to flexibly set their maximum bid costs, clients can save cost in some situations. For example, if a client specifies a low maximum bid cost and there is a period where the spot cost is less than the client's maximum bid cost, the client can save cost for running instances during that period. However, even though spot instances enable clients to save cost in some situations through flexible pricing, clients generally cannot plan when to start and terminate their applications.

B. Cloud Based TESTBED Agent

We demonstrate the key ideas presented in this paper, an agent-based Cloud testbed (Fig. 1) was designed and implemented using Java and the Java agent development (*JADE*) framework. In a Cloud market, there are many clients and service providers, and thus, the Cloud testbed has client agents and provider agents acting on behalf of clients and providers. The roles of each component in the testbed are summarized in Table I.

Cloud service providers and clients participate in the Cloud market of the testbed through the Cloud market registry. All agents participating in the Cloud market is registered in the Cloud market registry implemented using the *JADE* directory facilitator. All client agents connected to the Cloud market registry can then recognize and communicate with each provider agent. Provider agents and client agents generate service descriptions and specify their preferences with regard to service name, cost, time slot, and negotiation strategy based on a *GUI*. For accessing cloud resources, the consumer and the provider making deals

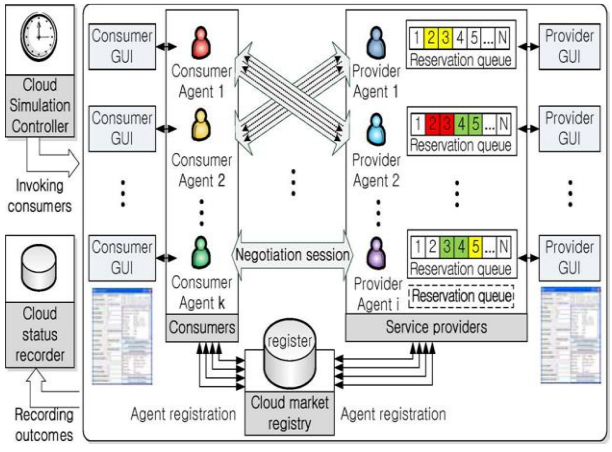

Figure 1: Agent-based Cloud testbed.

within the cloud computing environment. Identifying an agreement among the consumer and the provider to address the loss, the agreement specifying at least a minimum service requirement through negotiation by the provider.

TABLE I
COMPONENTS OF THE AGENT-BASED CLOUD TESTBED

Component	Roles
Provider agent	Service provider, service advertisement, price and time–slot negotiation
Consumer agent	Service consumer, service discovery, price and time–slot negotiation
Cloud market registry	Agent information repository
Cloud simulation controller	Simulation controller for periodic simulation
Cloud status recorder	Status recorder of information of the Cloud market and negotiation outcomes from all negotiation sessions in the market

C. Cost and Time-Slot Negotiations

This work considers bilateral negotiations between a client and a provider, where both agents are sensitive to time and adopt a time-dependent concession-making strategy for *CTNs*. Since both agents negotiate on both cost and time slot, generating a counterproposal can be making either a concession or a tradeoff between cost and time slot. Hence, an agent's strategy for multi-issue negotiation is implemented using both the following:

1) Tradeoff Algorithm: The novelty of this work is adopting a new tradeoff algorithm, called a "burst mode" proposal, which is designed to enhance both the negotiation speed and the aggregated utility. In the existing literature, a multi-issue proposal from agent

2) Concession-Making Algorithm: The concession-making algorithm determines the amount of concession

IV. EXPERIMENTAL RESULT

In this section, we show the result which we are implemented in Section –IV we will provide result topic-by topic with help of table and screen shot.

A. Cloud Based Testbed Agent

In this module, we show that" how message passing between client service provider agents in fig-2. The Cloud status recorder records all negotiation and reservation results of all provider and client agents in the Cloud market. It automatically generates random values for a resource reservation and transmits the values For Cloud service booking; the testbed provides simple service discovery

functionality through message passing. A client agent broadcasts a message indicating the name of the Cloud service that the client needs to all provider agents, and a provider agent who has the service replies to the client. In addition, the testbed provides a *PTN* mechanism to search for a mutually acceptable agreement for leasing the service. In the testbed, provider agents represent reserved time slots by marking the memory array and transmit reservation results to the contracted client to prevent duplicated booking.

The broker agent shows in figure 6.1 is mainly used for negotiation process. The information about the communication between the consumer and producer send through this broker agent. This broker agent isalso known as dummy agent of this negotiation process.

The QoS monitors the consumer data and combine these data into QoS metrics for the path. Based on this monitoring process the service provider who is exactly matching with the negotiation processes of the consumer will be selected. A service provider, publishing the descriptions and conditions for task negotiation Service consumers perform the negotiation on tasks, time and price.

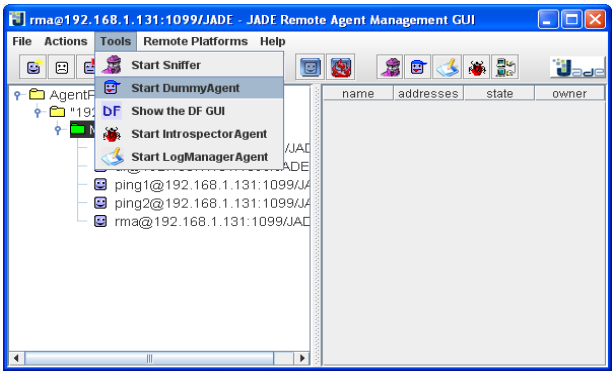

Figure 2: Establish the communication between client and agents

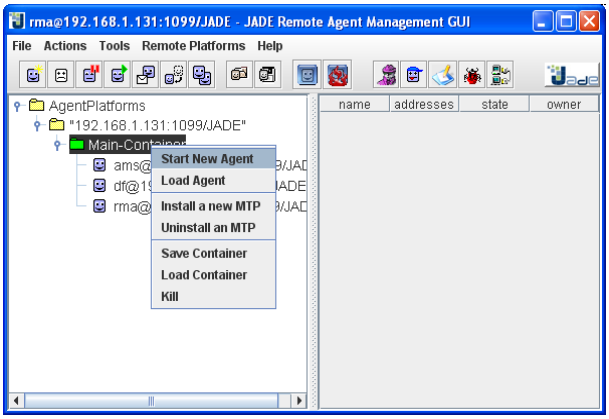

Figure 3: establish the communication other agent at a same time

B. Cloud Pricing through Negotiation

In this module, we show the cloud registry to client where client can find service provider details and infrastructure details such hardware, software, feedback. This registry is also helpful for service provider to get knowledge about client history, requirement details, and payment delivery feedback from other service provider. Here table II show the service provider details and table –III shows cloud details who already participated in cloud market.

The method used here to obtain a predictive model, and to answer the questions above, is tradeoff. Boosting was tried also, and some derived attributes were added to the training set. After two important derived attributes were added, boosting did not give any significant increase in accuracy on a validation set taken from the training set, and neither did adding more derived attributes. Therefore, the results here do not use boosting, and only use two derived attributes.

Table-II Cloud Registry Data

Input Data	Possible Values	Settings
Cloud Loading (CL)	$CL = N_{res}/N_{tot}$	$0 \le CL \le 1$
No. of provider agents	Integer	100 service provider agents
No. of consumer agents	Integer	200 consumer agents
Cloud services a provider leases	String	200 services/provider (randomly selected)
No. of negotiation sessions per each simulation	Integer	300 negotiation sessions

Table-III Cloud Participants' Inputs for Service Booking

Input Data	Possible Values	Settings	
		Consumer	Provider
Initial price (IP)	Integer (Cloud $)	10–60	200–250
Reserve price (RP)	Integer (Cloud $)	200–250	10–60
First time slot (FT)	Integer, FT<LT	10–60	10–60
Last time slot (LT)	Integer, FT<LT	300–350	300–350
Job size	Integer (Unit: Cloud time)	2–8	2–8
Negotiation Strategy(λ)	Conciliatory ($\lambda < 1$) Linear ($\lambda = 1$) Conservative ($\lambda > 1$)	1/3, 1.0, 3.0	1/3, 1.0, 3.0
Negotiation deadline (τ)	Integer (Unit: Round)	50 rounds, 200 rounds	50 rounds, 200 rounds

Cloud market is parameters for the Cloud simulation controller. In the experiments, 100 service provider agents and 200 client agents registered in the Cloud market. They are automatically generated by the controller, and the controller randomly invokes a client agent 300 times for each simulation to start a service reservation. The selection of the value of each variable is based on experimental tuning. Experimental tuning shows that 300 negotiation sessions for each simulation are sufficient to obtain stable results.

C. Cost-Time Slot Negatiations

In this module, we demonstrate the application of the *CTN* mechanism for cost Cloud resources. Here, we show that "how will work our proposed cost-time negation method in table". We also present feature of this how the client are booking the according their time and getting service with his/her own time with affordable price.

We implemented a Cloud computing testbed for simulating an Infrastructure as a Service (*IaaS*) Cloud provider using the Xen hypervisor that virtualizes computer hardware to multiple guest operating systems. The testbed (Fig. 11) consists of the following: 1) an *IaaS* Cloud provider represented by a provider agent and 2) several clients represented by client agents. Client and provider agents can adopt one of the four pricing models: on demand, reserved, spot, or *PTN* instances. A test set is used to determine the accuracy of the model. Usually, the given data set is divided into training and test sets, with training set used to build the model and test set used to validate it. Find a model for class attribute as a function of the values of other attributes.

The hardware resource of the Cloud is virtualized by a Xen hypervisor (version 4.0.1) that launches two guest domain types (Domain 0 and Domain U).Whereas Domain 0 is the privileged guest running on the hypervisor with direct hardware access and guest management responsibilities,

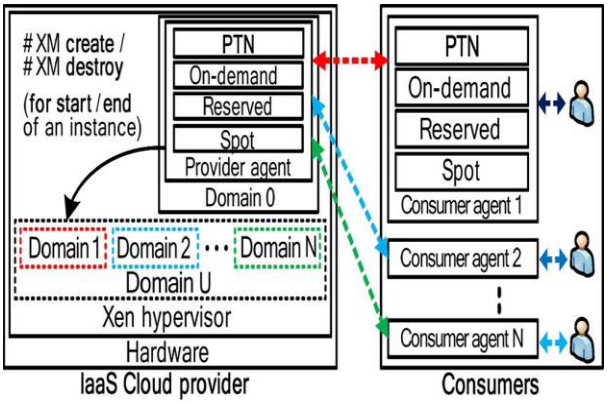

Figure 4: IaaS Cloud provider agent and consumer agents.

Table-V Hourly Usage Price of Four Pricing Models

	On-demand	Reserved	Spot	PTN		
					IP	RP
[Small instance] vCPU: 4 Memory: 600MB Storage: 40GB	$0.085	$0.030	$0.010 – $0.100	Provider	$0.100	$0.010
				Consumer	$0.010	$0.100
[Large instance] vCPU: 8 Memory: 1GB Storage: 80GB	$0.340	$0.120	$0.040 – $0.400	Provider	$0.400	$0.040
				Consumer	$0.040	$0.400

Domain U (Domain 1-N) launched and controlled by Domain 0 is the unprivileged guest with no direct access to the hardware. Domain U is provided to clients as *VM* instances. In Domain 0, the provider agent is implemented using JADE. The Cloud provides two types of instances, namely, small instances and large instances, and their hourly costs are listed in Table VI. The pricing for both ondemand and reserved instances follow that in Amazon EC2.

V. CONCLUSION AND FUTUREWORK

In this paper, we designs and implements a *CTN* mechanism for Cloud service booking. *CTN* mechanism is designed for both cost and time-slot negotiations. Although there are single-issue negotiation mechanisms and multi-issue negotiation mechanisms for Grid resource negotiation, none of these works considers time-slot negotiation. We present novel tradeoff algorithm, known as the "burst mode" proposal, has been designed to enhance both the negotiation speed and the aggregated utility of cost and time slot in a multi-issue negotiation. *PTN* agents can concurrently make multiple proposals that generate the same aggregated utility but differ only in terms of the individual cost and time-slot utilities. We have developed an agent-based Cloud testbed for design the CTN mechanism.

For future work, this paper can be utilized for following works1) considering and specifying other negotiation issues (e.g., QoS); 2) enhancing the proposed tradeoff algorithm by adaptively controlling the number of concurrent proposals in a burst mode proposal to reduce the computational complexity; and 3) providing a user interface for translating high-level user preferences into low-level technical specifications of the time-slot function. We expect that it is a more efficient tradeoff algorithm by adaptively controlling the number of concurrent proposals in a burst mode proposal.

REFERENCES

[1] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Gener. Comput. Syst.*, vol. 25, no. 6, pp. 599–616, Jun. 2009.

[2] A. Andrieux, K. Czajkowski, A. Dan, K. Keahey, H. Ludwig, T. Nakata, J. Pruyne, J. Rofrano, S. Tuecke, and M. Xu, "Web Services Agreement Specification (WS-Agreement)," WS-Agreement Specification from the Open Grid Forum (OGF), Mar. 2007. [Online]. Available: http://ogf.org/documents/GFD.107.pdf

[3] J. Wilkes, "Utility functions, costs, and negotiation," in *Market Oriented Grid and Utility Computing*, R. Buyya and K. Bobendorfer, Eds. New York: Wiley, 2009.

[4] I. Foster, C. Kesselman, C. Lee, B. Lindell, K. Nahrstedt, and A. Roy, ,"A distributed resource management architecture that supports advance, booking and co-

allocation," in *Proc. 7th IWQoS*, Los Alamitos, CA, Mar. 1999, pp. 27–36.
[5] I. Foster, A. Roy, and V. Sander, "A quality of service architecture that combines resource reservation and application adaptation," in *Proc. 8th Int. Workshop Quality Service*, 2000, pp. 181–188.
[6] M. A. Netto, K. Bubendorfer, and R. Buyya, "SLA-based advance booking with flexible and adaptive time QoS parameters," in *Proc. 5th Int. Conf. Service-Oriented Comput.*, Vienna, Austria, Sep. 2007, pp. 119–131.
[7] S. Venugopal, X. Chu, and R. Buyya, "A negotiation mechanism for advance resource reservation using the alternate offers protocol," in *Proc. 16th IWQoS*, Twente, The Netherlands, Jun. 2008, pp. 40–49.
[8] K. M. Sim, "Grid resource negotiation: Survey and new directions," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 40, no. 3, pp. 245–257, May 2010.
[9] J. Yan, R. Kowalczyk, J. Lin, M. B. Chhetri, S. K. Goh, and J. Zhang, "Autonomous service level agreement negotiation for service composition provision," *Future Gener. Comput. Syst.*, vol. 23, no. 6, pp. 748–759, Jul. 2007.
[10] F. Lang, "Developing dynamic strategies for multi-issue automated contracting in the agent based commercial grid," in *Proc. IEEE Int. Symp. CCGrid, Workshop Agent-Based Grid Econ.*, Cardiff, U.K., May 2005, pp. 342–349.
[11] K. M. Sim, "Towards complex negotiation for Cloud economy (position paper)," in *Proc. 5th Int. Conf. Grid Pervasive Comput., Lecture Notes in Computer Science*, 2010, pp. 395–406.
[12] K. M. Sim, "Equilibria, prudent compromises, and the "waiting" game," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 35, no. 4, pp. 712–724, Aug. 2005.
[13] A. Rubinstein, "Perfect equilibrium in a bargaining model," *Econometrica*, vol. 50, no. 1, pp. 97–109, Jan. 1982.
[14] H. Gimpel, H. Ludwig, A. Dan, and R. Kearney, "PANDA: Specifying policies for automated negotiations of service contracts," in *Proc. ICSOC*, vol. 2910, *LNCS*, New York, 2003, pp. 287–302.
[15] R. Lawley, M. Luck, K. Decker, T. R. Payne, and L. Moreau, "Automated negotiation between publishers and clients of grid notifications," *Parallel Process. Lett.*, vol. 13, no. 4, pp. 537–548, Dec. 2003.
[16] K. M. Sim, "G-commerce, market-driven G-negotiation agents and Grid resource management," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 36, no. 6, pp. 1381–1394, Dec. 2006.

T.Sanjeevakodi received the B.E degree(first-class) in computer science and Engineering from the Anna University Chennai 2012.currently doing the ME degree in Computer Science at Anna University Chennai/A.S.L Pauls College of Engineering & Technology ,Coimbatore. Her research interest include reliability/security of cloud computing.

R.Poonkodi received the ME degree in computer Science at Anna University Chennai 2012.currently working assistant professor at the A.S.L Pauls College of Engineering & Technology ,Coimbatore .His research interest include reliability/security of cloud computing and storage, distributed systems and Networks.

