Hysteresis-based Robust Trust Computing Mechanism for Cloud Computing

Mohamed Firdhous*†, Suhaidi Hassan*, Osman Ghazali*
*InterNetWorks Research Laboratory, School of Computing, Universiti Utara Malaysia, Malaysia
Email: mfirdhous@internetworks.my, {suhaidi,osman}@uum.edu.my

†Faculty of Information Technology, University of Moratuwa, Sri Lanka
Email: firdhous@uom.lk

***Abstract*—Cloud computing has been the new paradigm in distributed systems where users can access computing resources and pay only for usage similar to other utilities like electricity, water, gas and telephony. Service Level Agreements signed at the beginning between the clients and service providers stipulate conditions of the services including the QoS requirements. Trust can be used to quantify the QoS levels of providers and rank them according to their performances. Hence trust management systems can play an important role in identifying the right service provider who would maintain the QoS at the levels required by the clients. Researchers have proposed several trust computing mechanisms based on different techniques and trust metrics on the literature. Almost all of these mechanisms increment or decrement the trust scores monotonously based on the inputs. This is a major vulnerability that can be exploited by adversaries to force the trust scores towards extreme values. In this paper, the authors propose a novel trust computing mechanism based on hysteresis function which requires extra efforts to force the output from one end to the other. Hysteresis functions are immune to small changes and hence can be used to protect the system from sporadic attacks. The proposed mechanism has been tested using simulations. The test results show that the trust scores computed using the proposed mechanism are more robust and stable in the face of attacks than other mechanisms.**

Keywords: trust computing, cloud computing, hysteresis, quality of service

I. Introduction

Similar to electricity, water, gas and telephony, cloud computing helps user to access and pay for computing resources such as hardware, operating system, development platform, software and other services as utilities over the Internet. Thus it is now commonly known as the 5th utility in the line of the above [1]. Due to the growing popularity of cloud computing, the cloud service market has already seen a lot of service providers selling their services at varying levels of service qualities and prices [2]. Once a customer has selected a suitable service provider, they enter into a Service Level Agreement (SLA) that specifies the conditions to be met by both parties and penalties imposed, in case of failures. Among the many conditions specified, Quality of Service (QoS) would take an important place in the SLA[3], [4]. The QoS guarantees are monitored and enforced through QoS attributes. Similar to any other distributed system, QoS plays an important role in cloud computing and due to the dynamic nature of it, it is necessary to monitor these attributes continuously [5]. Different QoS parameters measure and characterize different aspects of service. These parameters include performance, throughput, reliability, availability, trust etc., and the selection of the right parameter depends on the type of application and user requirements. According to Garg et al., transactional applications such as web based real time services necessitate better response time and throughput guarantee, while non-interactive services that are run as batch jobs place more importance on job completion times, reliability and accuracy [6]. In order for customers to enter into a contract with a service provider confidently, they require to have a good knowledge of the performance of the service provider. Hence a system that could quantify the QoS would be helpful for customers to identify the service providers who would meet their requirements. A trust management system could be employed to quantify and rank the service providers according to their performance that can be accessed by clients prior to signing the SLA [7], [8]. Several trust computing mechanisms have been proposed in literature for quantifying various aspects of distributed systems. Almost all these mechanisms use a function that is monotonously increasing or decreasing the trust value based on the input. This paper proposes a novel trust computing mechanism based on a hysteresis function. Hysteresis has the special feature of maintaining more than one internal state at a time, which requires to know the history to predict the future states. By exploiting this feature, the proposed mechanism successfully mitigates the effects of malicious attackers on the trust ratings. The proposed mechanism has been tested in a simulated environment and the results show that the proposed mechanism is superior to other systems in mitigating the effects of malicious attacks while maintaining a truly representative trust rating.

This paper consists of five sections as follows: Section I introduces the paper while sections II and III discuss cloud computing and trust and trust management respectively. Section IV introduces the new trust computing mechanism proposed in this paper along with the experiments conducted to verify it. Finally Section V concludes the paper along with recommendations for future work.

II. Cloud Computing

In the line of electricity, water, gas and telephony, cloud computing has been considered the 5^{th} utility as it makes the computing resources including hardware, development platform and software available over the Internet on a pay as you go model [1]. Cloud computing helps organizations to fully outsource their computing requirements and pay for them only what is used. The main difference between cloud computing and traditional outsourcing is the total absence of commitment with regard to the resource requirements and only be billed for what is actually used [9]. Under the traditional outsourcing model, organizations are supposed to forecast their computing resource requirements and lease the full amount from large data centres. The clients will be charged for the full commitment from the beginning irrespective of usage. If the clients require more resources than what is leased, the additional requirement will not be available unless they upgrade the contract. But in the cloud computing model, no such initial commitment is required and the resource provisions would closely follow consumption and the billing would be done only for the actual consumption. With cloud computing, organizations that start small and grow big will be able to tailor make their computing requirements to closely follow their growth pattern. Even seasonal variations in demand patterns can be accommodated without any user interventions. Non commitment and strict adaptation of resource provision and payment along with usage help organizations to invest their financial resources on their core business operations rather than on non performing computing resources as with cloud computing, expenditure on computing is an operational cost rather than a capital cost [10]. Cloud computing resources are hosted on virtualized platforms so that they can be made available and removed on the fly based on customer demand [11]. The ability to bring up resources on the fly provides the service providers with the flexibility of selling the same physical resources to multiple clients increasing the utility of the resources. Increasing the utility helps them bring the cost per user down as the number of users supported on a single hardware devices increase. Thus hosting the applications on cloud helps customers financially multiple ways including eliminating the initial cost, reducing the operating cost and maintenance cost etc.

Infrastructure as Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS) are the three categires under which cloud computing services are currently marketed [12]. Virtualizing and selling hardware resources including processing power, hard drive space, memory, database storage etc., are known as IaaS. An IaaS vendor installs a hardware level virtualization software commonly known as hypervisor or Virtual Machine Monitor (VMM) on a host computer in order to create multiple virtual computers on top of a single physical computer. The VMM manages the operation of the virtual computers through multiplexing of the virtual machines on the same hardware [13]. From the customers' perspective, the virtual computers made available to them can be treated as real computers and install any guest operating system of their choice on them [14]. Thus a single physical computer can be configured to host many operating systems and applications on them. The VMM installed on the host computer provides the necessary isolation and security for these operating systems and applications so that they do not interfere with each other [15]. PaaS is the provisioning of complete software development platforms comprising operating system, development and testing tools and Application Programming Interface (API) [2]. PaaS reduces the application development time and cost and help developers to bring the product to the market in a shorter time. This is mainly due to the fact that the developers need not worry about purchasing, installing and maintaining of hardware or software. Also, they will be charged only for the usage of resources, which totally eliminates the payment for idle resources. Web based applications are developed, hosted and sold over the Internet is known as SaaS. SaaS is now considered to be the new paradigm in software marketing opposed to purchasing, installing and maintaining software in house. Now software can be accessed, used and paid only for what is accessed for how long [16]. Moreover, these applications can also be customized to suit the specific requirements of customers similar to locally hosted applications [17]. Fig. 1 shows the cloud computing layered architecture which is made up of physical hardware, virtualized hardware and cloud computing service layers.

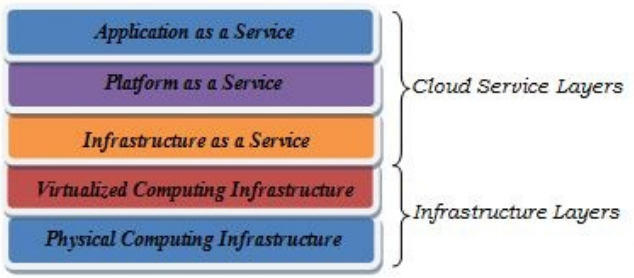

Fig. 1. Layers of Cloud Computing [18]

In addition to the cloud computing services described above, new services have been introduced to the market under different names such as Communication as a Service (CaaS), Data as a Service (DaaS), Network as a Service (NaaS) and Identity and Policy Management as a Service (IPaaS) [19]. Presently all these services have been commonly known as XaaS (Anything as a Service) [20].

III. Trust and Trust Management

Trust has been defined as a psychological state that a person has on another in terms of his capabilities for performing certain tasks, if assigned [21]. Trust has been used in open distributed systems to identify and select the right peer to interact with [22]. Prior experience and knowledge plays an important role in building trust on a person or system. Several trust computing mechanisms have been developed by researchers for the purpose of computing and quantifying

trust, which can help new users to select the suitable remote system. These mechanisms base their computation on well known functions or methods such as entropy, fuzzy reasoning, iteration etc., [23], [24], [25]. In this paper, the authors propose a trust computing mechanism based on a hysteresis function. This is essentially an iterative mechanism as the trust scores are continuously tuned based on the performance of the system and the previous trust score. Researchers have already exploited the non-linear and asymmetric properties of hysteresis for developing systems with immense practical value. Examples of these systems include computational trust model for electronic commerce, pricing threshold queues and handover management in WLANs[26], [27], [28].

IV. Computing Trust

The computation of trust requires to go through three main stages [29]. They are namely;

1) Trust Forming
2) Trust Evolution and
3) Trust Distribution

Out of the three stages, the first two namely, trust formation and trust evolution can be combined as trust computing and the third stage involves the distribution of the trust scores computed in the previous stages among the collaborating trust management systems. In the first stage that is the trust forming, the initial trust score for a fresh system that has not had any interaction with clients so far has been computed. The initial value may simply be assumed to take a neutral value such as the mid point between the best and the worst scores or computed using the initial information available on the system capabilities including hardware, software and network. The second stage, the trust evolution continuously modifies the trust scores based on the performance of the system and the experience of the clients. The main objective of the third stage is to create a larger cooperating system that covers a wider region and can respond to client requests faster. This paper concentrates on developing a trust computing mechanism that is more representative of system capabilities as well as robust in the face of attacks by malicious nodes.

A. Hysteresis-based Trust Computing Mechanism

The proposed trust management system includes both trust forming and trust evolution units along with other subsidiary units required. Fig. 2 shows the proposed system in a block diagrammatic format. In this paper, the initial trust score for any new system is assumed to be neutral. Hence the trust forming unit does not have much significance here. The QoS monitor continuously tracks the performance of the cloud provider in order to obtain the latest QoS attributes. The trust evolution unit receives two inputs, one directly from the client which specifies the expected QoS value and the other one from the QoS monitoring unit which provides the actual QoS value. Fig. 3 shows the trust evolution unit in detail. The trust evolution unit is made up of a temporary storage unit, that stores the actual QoS values in a FIFO queue. When a new QoS value is received, it is added to the end of the queue. If the queue is already full, the oldest QoS value would be removed from the queue to make space for the new value received. The FIFO arrangement of the storage unit makes sure that the system holds the information about the most recent performance of the cloud system. The other units are the summing point and the hysteresis function unit. The input to the summing point are the expected QoS value received from the client and the median value of the most recent QoS values stored in the temporary storage unit. The median value has been selected due to its stability with respect to spurious responses that may occur due to extraneous reasons. The summer would then compute the difference between the median QoS value and the expected QoS value and then the result is supplied to the hysteresis function unit. The hysteresis function unit would then compute the trust value and update the trust table. Fig. 4 shows the algorithm used for computing trust.

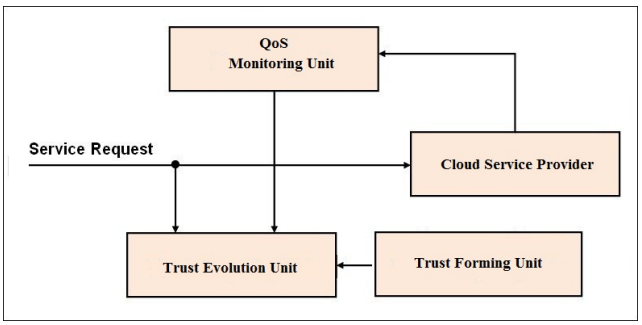

Fig. 2. Trust Management System

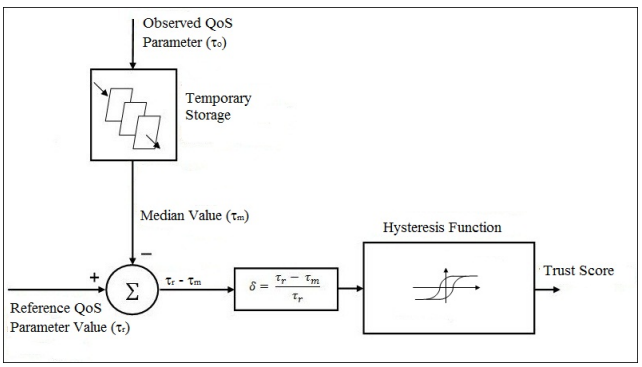

Fig. 3. Trust Computing Unit

The trust computing algorithm presented in Fig. 4, calculates the trust score using the inputs on a hysteresis function. Based on the previous trust score and the normalized QoS value δ, the new trust score would be computed. The hysteresis has several special features that distinguish it from other non-linear functions. These special features have been exploited in our work in order to make the trust computing mechanism more rugged and representative of the actual system performances. The special properties that can be useful for trust computing are:

Store observed QoS value τ_o in temporary storage

Compute median QoS value τ_m

Retrieve reference QoS value τ_r

Compute differential $d = \tau_r - \tau_m$

Compute normalized QoS parameter $\delta = \frac{\tau_r - \tau_m}{\tau_r}$

Computer Trust Score $T = Hysteresis(\delta)$

Fig. 4. Trust Computing Algorithm

- Shape of the curve (SS-shape)
- Limited linear operation in the middle region
- Large input range (from -∞ to +∞)
- Inherent memory of the system

B. Hysteresis Function

Fig. 5 shows the shape of a hysteresis function. From this figure, it can be seen that the output of the function not only depends on the input but also on the current state or the history. Hence to drive the output from one extreme to other more effort on the part of the input is required than what was required to bring it to that state originally. Hysteresis has been associated with several natural phenomena. Magnetic/electrical hysteresis in ferromagnetic and ferroelectric materials, deformation of rubber or alloys under applied force and natural rate of unemployment in a given economy are few examples for hysteresis in natural systems. In order to exploit the beneficial effects of hysteresis, scientists have developed artificial systems with hysteresis behaviour in order to avoid the ill effects of rapid oscillations due to external environment. Hysteresis based systems have been used in several engineering domains including electronics, control systems, telecommunications, mechanical engineering, software engineering etc [30]. In this paper, the authors exploit the delayed response of hysteresis to create a stable trust computing mechanism for cloud computing.

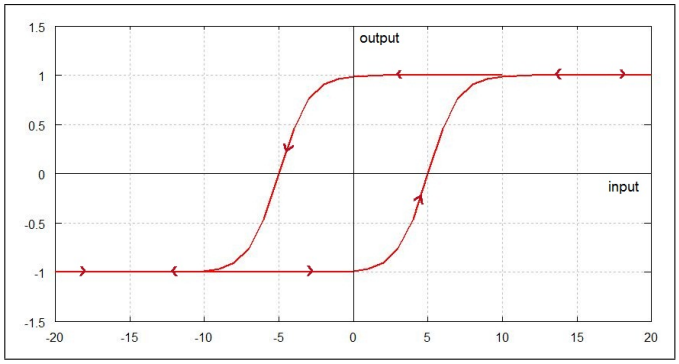

Fig. 5. Example Hysteresis Curve

The proposed system further reduces the transient behaviour of the systems by computing the median value of the last recent outputs instead of using the last value for computation. Hence the combination of both statistical behaviour of the system through the median score of the QoS parameter along with a hysteresis based approach would result in a better stable mechanism than the existing ones. Also the median score of performance reduces the number of responses needed to be stored compared to other statistics like mean or mode for consistent performance. The computed trust values are stored in a trust buffer as the new value computed would either increment or decrement the previously stored value based on the recent performance of the system. Only the last trust value computed needed to be stored in the buffer reducing the storage requirements of the system. By maintaining multiple output buffers, the trust system can be modified to support differentiated services with varying performance requirements.

C. Experimental Setup

The proposed mechanism was tested under a simulated environment and the results were compared against that of entropy based trust computing mechanism proposed by Yang el al., in [23]. The individual units were created using GNU Octave software and combined to form the final system. The hysteresis function shown in Eq. 1 was used by combining two horizontally shifted sigmoid functions. Eq. 2 was used to compute the final trust score. Eq. 2 spreads the trust scores between (-1, +1) without modifying the shape of the distribution of scores. In order to eliminate the bias inherent in small samples, 30 most recent response times were collected and stored in the Temporary Storage. The response time of the system was considered as the QoS parameter of interest during this study. But, the selection of the QoS parameter is independent of the proposed mechanism and hence any QoS parameter can be used to compute the trust score based on that parameter.

$$T_t = \begin{cases} \frac{1}{1+e^{-(\delta_s - d)}} & \text{if } \delta > 0 \\ \frac{1}{1+e^{-(\delta_s + d)}} & \text{if } \delta < 0 \end{cases} \quad (1)$$

δ_s - cumulative response time
d - horizontal shift

$$T = 2 * T_t - 1 \quad (2)$$

For creating a consistent environment throughout the experiments, the median response time was maintained fixed while a random number generator was used to generate actual response times. The horizontal shift was also fixed at $d = 5$ throughout the experiments. The randomly generated response times were trimmed to contain within a specified range as extreme values would not be practical in a real situation. Fig. 6 shows trust scores generated using the proposed mechanism and entropy based trust computing mechanism. In this experiment, the response times were controlled to make sure that the normalized response time δ would cover the

entire range from negative to positive and back in order to understand the complete behaviour of the mechanism. Using the same response times, the entropy based mechanism was also executed simultaneously. From this figure, it could be seen that the trust scores computed using the proposed mechanism resists drastic changes while the entropy based mechanism reacts even to small changes in response times. Hence the proposed mechanism is more stable in the face of fluctuations compared to the entropy based mechanism. It is also possible to see that the proposed mechanism has a steeper linear region in the middle, where the new systems would be operating compared to the entropy based mechanism. The steeper region would help new systems to converge towards their appropriate trust scores faster than the entropy based mechanism.

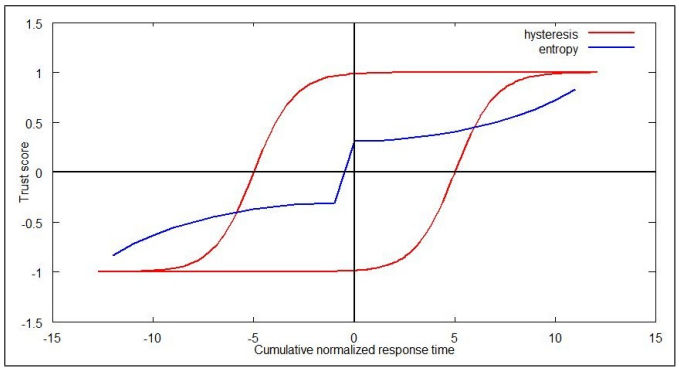

Fig. 6. Trust Scores for Random Request Times

Fig. 7 shows the number of attempts required to change the trust score from a positive value to negative. This experiment has been carried out by sending the same normalized response time repeatedly with the intention of modifying the trust scores. The main objective of this experiment was to test the robustness the proposed mechanism in the face of malicious attacks and also the compare its performance against the entropy based mechanism under the same conditions. Malicious peers may in collusion with others send false results repeatedly to the trust management system in order to manipulate the trust scores. Hence a trust management system, especially the trust computing mechanism must be robust enough to withstand such attacks. From Fig. 7, it can be seen that the proposed hysteresis based mechanism is very robust in the face of attacks compared to the entropy based mechanism. It is necessary to send very large number of attacks continuously to modify the trust scores in the proposed mechanism whereas the entropy based mechanism gets affected immediately starting from the very first attack. The main reason for the robustness of the proposed mechanism is its non linearity along with the hysteresis behaviour where it maintains different thresholds based on in which direction trust scores have been modified. The different thresholds on each direction, makes the proposed mechanism resilient for momentary fluctuations and robust in the face of attacks. Similarly, it would be possible to show that it would take similar efforts to change the trust score from -1 to $+1$ in the proposed mechanism. The entropy based mechanism would also simply follow the same pattern it followed from $+1$ to -1 as the trust scores move along the same line for both positive and negative response times.

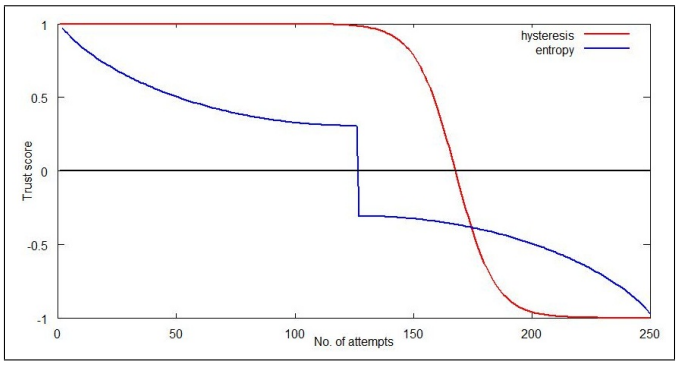

Fig. 7. Malicious Attempts to Change Trust Score

Fig. 8 shows the effect of the magnitude of the response time on the computation of trust scores on both proposed mechanism as well as the entropy based mechanism. It can be seen that when the normalized response time is large, trust scores converge faster in both systems. This effect can be considered as both positive and negative. Larger the deviation of response time from the requested response time, faster the trust score must converge towards that end. But at the same time, if the malicious nodes use larger values rather than the number of attempts to attack the system, both mechanisms would be affected in a similar fashion. But, from Fig. 8, it can be seen that entropy based mechanism gets affects faster than the proposed mechanism. Hence, it can be concluded that the proposed mechanism is more robust and resilient compared to the entropy based mechanism in the face of attacks. Since continuous streams of similar request times or larger request times may indicate an attack on the system. Hence when such an attack pattern is detected, it is possible to employ other security measures to protect the system against such attacks. But, the development or discussion of such a mechanism is the beyond the scope of this paper and hence not discussed further.

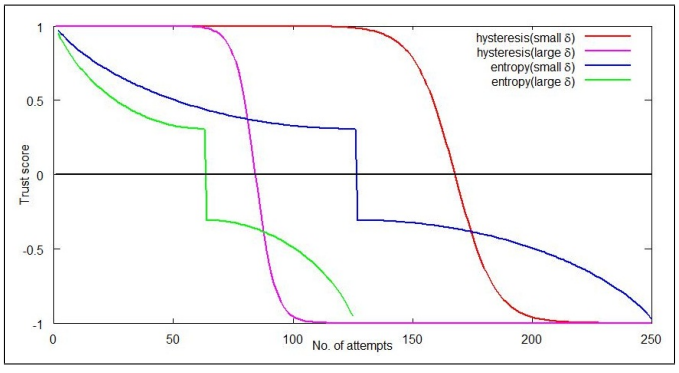

Fig. 8. Effect of Response Time on Trust Scores

From the results above, it can be seen that the proposed mechanism helps systems to reach their appropriate trust

scores faster than the entropy based mechanism. Also, the proposed mechanism is better and stable in the face of repeated requests as it requires much larger and stronger attack than that is required to break the entropy based mechanism.

V. Conclusions

In this paper, the authors have presented a hysteresis based robust trust computing mechanism that computes the trust scores for a cloud computing system based on any QoS parameter. The proposed mechanism computes the trust scores using a non linear hysteresis function that can be in more than one state at any given time. The hysteresis function makes the trust computing mechanism more stable due to its special properties, such as the shape, the large input range and inherent memory. The proposed system has been tested using simulations and the results were compared against that of entropy based mechanism proposed by Yang et al., in [23]. The results obtained from the proposed mechanism were found to be converging faster towards their ultimate scores and more stable in the face of attacks by malicious peers.

As future work, it is proposed to investigate further into impact of the shape of the hysteresis curve on the stability of the mechanism. This can be done by using different functions to create the hysteresis loop. It is also proposed to investigate further into the security of the system with special reference to identifying malicious requests and isolating them.

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," *Journal of Future Generation Computer Systems*, vol. 25, no. 6, pp. 599–616, June 2009.

[2] B. P. Rimal, E. Choi, and I. Lumb, "A taxonomy and survey of cloud computing systems," in *Fifth International Joint Conference on INC, IMS and IDC, Seoul, Korea*, 2009, pp. 44–51.

[3] P. Wieder, J. M. Butler, W. Theilmann, and R. Yahyapour, Eds., *Service Level Agreements for Cloud Computing*. Springer, 2011.

[4] L. Wu and R. Buyya, *Performance and Dependability in Service Computing: Concepts, Techniques and Research Directions*. IGI Global, 2012, ch. Service Level Agreement (SLA) in Utility Computing Systems, pp. 1–25.

[5] P. Patel, A. Ranabahu, and A. Sheth, "Service level agreement in cloud computing," in *ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications, Orlando, FL, USA*, 2009, pp. 1–10.

[6] S. K. Garg, S. K. Gopalaiyengar, and R. Buyya, "SLA-based resource provisioning for heterogeneous workloads in a virtualized cloud datacenter," in *11th International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP 2011), Melbourne, Australia*, ser. LNCS 7016, vol. I. Springer, 2011, pp. 371–384.

[7] L. H. Vu, M. Hauswirth, and K. Aberer, "Qos-based service selection and ranking with trust and reputation management," in *Cooperative Information System Conference (CoopIS05), Agia Napa, Cyprus*, 2005, pp. 466–483.

[8] M. Firdhous, O. Ghazali, S. Hassan, N. Z. Harun, and A. Abas, "Honey bee based trust management system for cloud computing," in *3rd International Conference on Computing and Informatics (ICOCI 2011), Bandung, Indonesia*, 2011, pp. 327–332.

[9] B. An, V. Lesser, D. Irwin, and M. Zink, "Automated negotiation with decommitment for dynamic resource allocation in cloud computing," in *9th International Conference on Autonomous Agents and Multiagent System (AAMAS '10), Toronto, Canada*, vol. 1, no. 1, 2010, pp. 981–988.

[10] V. Fusenig and A. Sharma, "Security architecture for cloud networking," in *International Conference on Computing, Networking and Communications (ICNC), Maui, HI, USA*, 2012, pp. 45–49.

[11] S. Zaman and D. Grosu, "Combinatorial auction-based allocation of virtual machine instances in clouds," in *IEEE Second International Conference on Cloud Computing Technology and Science, Indianapolis, IN, USA*, 2010, pp. 127–134.

[12] C. Vecchiola, S. Pandey, and R. Buyya, "High-performance cloud computing: A view of scientific applications," in *10th International Symposium on Pervasive Systems, Algorithms, and Networks (ISPAN), Kaohsiung, Taiwan*, 2009, pp. 4–16.

[13] K. Hwang, J. Dongarra, and G. Fox, *Distributed and Cloud Computing: From Parallel Processing to the Internet of Things*, 1st ed. Morgan Kaufmann, 2011, ch. Virtual Machines and Virtualization of Clusters and Data Centers, pp. 130–187.

[14] M. Firdhous, O. Ghazali, and S. Hassan, "A trust computing mechanism for cloud computing with multilevel thresholding," in *6th International Conference on Industrial and Information Systems, Kandy, Sri Lanka*, 2011, pp. 457–461.

[15] F. Zhao, Y. Jiang, G. Xiang, H. Jin, and W. Jiang, "VRFPS: a novel virtual machine-based real-time file protection system," in *7th ACIS International Conference on Software Engineering Research, Management and Applications, Haikou, China*, 2009, pp. 217–224.

[16] R. Prodan and S. Ostermann, "A survey and taxonomy of Infrastructure as a Service and web hosting cloud providers," in *10th IEEE/ACM International Conference on Grid Computing, Banff, Alberta, Canada*, 2009, pp. 17–25.

[17] J. Kong, "Protecting the confidentiality of virtual machines against untrusted host," in *International Symposium on Intelligence Information Processing and Trusted Computing (IPTC), Huanggang, China*, 2010, pp. 364–368.

[18] M. Firdhous, O. Ghazali, and S. Hassan, "A trust computing mechanism for cloud computing," in *Fourth ITU Kaleidoscope Academic Conference*, 2011, pp. 199–205.

[19] M. Zhou, R. Zhang, D. Zeng, and W. Qian, "Services in the cloud computing era: A survey," in *Fourth International Universal Communication Symposium, Beijing, China*, 2010, pp. 40–46.

[20] R. Mikkilineni and V. Sarathy, "Cloud computing and the lessons from the past," in *18th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises, Groningen, Netherlands*, 2009, pp. 57–62.

[21] T. Bosse, C. M. Jonker, J. Treur, and D. Tykhonov, "Formal analysis of trust dynamics in human and software agent experiments," in *11th International Workshop Cooperative Information Agents, Delft, The Netherlands*, 2007, pp. 343–359.

[22] H. Li and M. Singhal, "Trust management in distributed systems," *IEEE Computer*, vol. 40, no. 2, pp. 45–53, 2007.

[23] L. Yang, Z. G. Qin, C. Wang, Y. Liu, and C. S. Feng, "A P2P reputation model based on ant colony algorithm," in *International Conference on Communications, Circuits and Systems, Chengdu, China*, 2010, pp. 236–240.

[24] H. Chen and Z. Yeo, "Research of P2P trust based on fuzzy decision-making," in *12th International Conference on Computer Supported Cooperative Work in Design, Xi'an, China*, 2008, pp. 793–796.

[25] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina, "The eigentrust algorithm for reputation management in P2P networks," in *12th International Conference on the World Wide Web, Budapest, Hungary*, 2003, pp. 640–651.

[26] J. Urbano, A. P. Rocha, and E. Oliveira, "Trust evaluation for reliable electronic transactions between business partners," in *Agent-Based Technologies and Applications for Enterprise Interoperability*, ser. Lecture Notes in Business Information Processing, K. Fischer, J. Mller, and R. Levy, Eds. Springer Berlin Heidelberg, 2012, vol. 98, pp. 219–237.

[27] L. M. L. Ny and B. Tuffin, "Pricing a threshold-queue with hysteresis," in *18th IASTED International Conference on Modelling and Simulation (MS 2007), Montreal, Canada*, 2007.

[28] M. Zaki, "Handover in a wireless local area network (WLAN)," US Patent 7 164 915, January 16, 2007.

[29] M. Carbone, M. Nielsen, and V. Sassone, "A formal model for trust in dynamic networks," in *1st International Conference on Software Engineering and Formal Methods, Brisbane, Australia*, 2003, pp. 54–61.

[30] F. Ikhouane and J. Rodellar, *Systems with Hysteresis: Analysis, Identification and Control Using the Bouc-Wen Model*. John Wiley & Sons, 2007.