Research Article

A Fuzzy Based Sensor Web for Adaptive Prediction Framework to Enhance the Availability of Web Service

Sundharam Ramalingam and Lakshmi Mohandhas

Faculty of Computer Science, Sathyabama University, Chennai 600119, India

Correspondence should be addressed to Sundharam Ramalingam: sun_moonr@yahoo.com

Received 26 September 2015; Revised 4 December 2015; Accepted 31 December 2015

Academic Editor: M. Anwar Hossain

Copyright © 2016 S. Ramalingam and L. Mohandas. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Present day businesses revolve around internet applications (e.g., e-commerce, e-business, and internet banking) which are primarily supported by web services. The increasing consumer reliance on these internet-based applications causes dynamic variation in the number of requests received by web services and a slowdown in response time during peak load periods. To overcome this difficulty, in this paper, we propose a fuzzy logic (FL) based prediction and replication framework to replicate the web services over distributed servers and achieve timely responses. This framework senses the proximity of future interval response time against agreed service level agreement (SLA) time using sensor web called proximity sensor service. Also, it determines the replication requirement using FL and implements the FL decision using resource control algorithm to replicate the web service on another server before the SLA time is violated or to reclaim the excess resource. We conduct appropriate test scenarios in a dedicated test environment and compare the results with existing models. The results show that our framework replicates the web service at more appropriate times than the existing models and achieves 82% prediction accuracy. Thus, our framework significantly improves the availability of web services with minimal intervention from system administrators.

1. Introduction

Web services entail well-defined software components that support web based business applications and increase the efficiency of providing services by exchanging and aggregating data in a distributed environment [1]. Several techniques have been developed for web services [2–4] to enhance their availability. Because the rate of web service requests received from the clients varies, the quality of service (QoS), including the availability and performance, becomes unstable, leading to various uncertainty factors and potential web service failures or temporary interruptions. Hence, a QoS aspect such as the availability of web services has become an important research topic and has attracted a lot of attention in recent years.

In this paper, we propose a fuzzy logic based replication SMA framework, called “A Fuzzy Based Sensor Web for Adaptive Prediction Framework to Enhance the Availability of Web Service” (FAPFEA). The soft sensor or virtual sensor is defined as "A web service which contains an algorithm or simulation model could be a sensor in a sensor web as long as its interfaces and behaviors comply with the specifications" [5]. Sensor web is “web accessible sensor networks and archived sensor data that can be discovered and accessed using standard protocols and application programming interfaces” [6]. Our proposed framework uses virtual sensor (Proximity Sensor Service (PSS)) which is capable of reading from heterogeneous databases (MS/SQL, MySQL, etc.) to sense the proximity of the predicted future interval response time with SLA response time of specific web service (e.g., user status enquiry service \( C_1 \)) and determines the replication requirement. The soft sensors have an important role in sensor validation when performance declines or fault accumulation. The sensor web framework is constructed as per set of standards [6, 7] laid down the Open Geospatial Consortium (OGC) to establish the “sensor web.” The OGC SWE, Microsoft SenseWeb, and Oak Ridge National Lab Sensorpedia are the most common sensor web platforms providing sensor data service [8].
Several techniques have been introduced in this domain. Self-management automation (SMA) system was introduced by IBM [10] to constantly monitor resource utilization and scale the capacity up or down. The SMA system becomes indispensable to monitor resource utilization constantly to scale up/down the resource capacity. So far, several attempts were made to develop SMA [11–14] in order to reduce administrator intervention in distributed resource management. However, this approach requires administrative responsibility to manage the system. The authors in [15] suggested architecture employing the enterprise-level gateway to improve web service availability. This technique continually monitors service availability and unfortunately leads to performance overhead for the gateway, thereby impacting client request and response processing.

A recent trend in web service availability is centered on the replication of web services [16–19]. The authors in [16] discussed different replication components and techniques, such as active, passive, and semiactive replication, as part of their proposed framework. The drawback of their proposed framework is that it uses a multicasting technique that increases the unnecessary load on the servers by processing the same request by multiple replicas. The authors in paper [20] also suggested an active replication for fast response. Also, authors stated that “a slow service is equivalent to unavailable service.” In paper [18], the authors proposed system architecture to enhance the availability of web services by employing an enterprise service bus, replication of services, and multicasting. However, from the load balance perspective, multicasting and parallel invocation is ineffective [17], because it always increases the traffic and operating cost by propagating the same client requests to all of the servers. The work proposed in paper [19] designed a middleware component involving interactions with a java package to maintain the consistency of replica statuses.

The work proposed in paper [21] is a prediction framework to improve the availability of web services by predicting the future response time. The framework issues a replication decision on another server host when the predicted response time exceeds 90% of the SLA time without considering the arrival rate. The authors in [22] have suggested a framework for dynamic placement of service and service replication to improve the availability of services using a team formation algorithm. The framework concentrates on cost management, performance, and availability in the event of web service failover and does not consider the arrival rate and control algorithm (RCA) to dynamically replicate the web service on another host before the response time violates the SLA. This framework is also designed to detect web service unavailability such as the web service being stopped, hung, or unresponsive for various reasons. Additionally, if the response time of any individual server is low and is approaching the threshold limit, the RCS removes the corresponding replica from the load balancer and replicates the web service on another host.
response time while replicating the web service. In paper [23], the authors proposed an adaptive prediction framework to enhance web service availability using replication. In this research work, the authors suggested a linear regression method to predict the future load based on data from the past 15 days and replicated the web service for the current day according to these data. However, this approach may not help in dynamically varying loads because the replication is not predicted based on the current load.

All of the existing research works discussed above used different evaluation techniques and criteria for replication decisions. However, none of the above frameworks proposed to sense both the arrival rate and response time in future interval. Because the arrival rate and response time are interrelated parameters [24, 25] and their combination forms numerous situations, it becomes necessary to consider all of the possible situations in determining the availability of web services and the replication decision. Ignoring either one of these metrics leads to inappropriate replication decisions. Therefore, we focus on predicting both the arrival rate and response time and analyze them using sensor web based FL to sense the proximity of future interval response time with agreed SLA time and to determine requirement of replication. Application server based sensor web architecture has been proposed by authors [7] to access the sensor data over the web. The author in paper [6] proposed architecture for Sensor Web Enablement for accessing sensor web from application interface. The authors in paper [8] proposed geosensor based efficient sensor data management methods and extended query processing system. But none of the works reported on sensor web based service replication. The objective of our proposed work is to achieve the availability of web services using heterogeneous sensor web according to the following aspects:

(1) Availability in terms of response time: the web service should successfully respond to client requests within the specified SLA time.

(2) Availability in terms of web service up time: the web service should be available to process the requests at all times, providing uninterrupted availability.

The remainder of this paper is organized as follows. Section 3 provides an overview of the proposed adaptive replication framework and its components. Section 4 describes the prediction process, techniques, and algorithms. Section 5 details the test environment, the different test scenarios, and the test results. Finally, the conclusion and future work are summarized in Section 6.

3. Overview of Proposed Framework

The proposed framework FAPFEA is built with service gateway; any request from client always passes through the service gateway which has four major components, namely, Monitoring Service (MS), Load Balancing Service (LBS), Intelligent Resource Control Manager (IRCM), and Proximity Sensor Service (PSS). The components of FAPFEA framework are described below and shown in Figure 2.

3.1. Monitoring Service (MS). The MS keep monitors the statuses of web service ($C_1$), servers ($R_1, R_2 \cdots R_n$) and creates metrics for number of requests arrived, response time of each request, and number of requests processed by web service ($C_1$) during a specific interval.

3.2. Load Balancing Service (LBS). LBS receives requests from clients and distributes them to the pool of active replicated web services using round-robin technique. These replicated web services process the requests and send responses back to the clients.

3.3. Intelligent Resource Control Manager (IRCM). The IRCM is a service component which manages and controls the web services replication to maintain the web services response time within the expected SLA time. It uses sensor web service called “Proximity Sensor Service (PSS)” that senses the proximity of the predicted future interval response time with SLA response time and determines the replication requirement using FL for efficient decision-making. IRCM also uses “Resource Control Service (RCS)” to implement the requirement of resources into system. IRCM acts as client to invoke the PSS at defined time interval to identify the availability of web services and requirement of new replica. Once IRCM receives the response from PSS, RCS further analyzes the PSS’s response using RCA to implement the PSS response at appropriate time. RCS is a sub-service component controlled by IRCM, which acts as actuator for PSS response. It analyzes the PSS response using resource control algorithm (RCA) and dynamically replicates the new replica or stops excessive replica at appropriate time.

3.4. Proximity Sensor Service (PSS). PSS adopts OGC SWE standards to provide the interoperable usage of sensor resources. It has been built upon the existing open source SOS implementation from the 52 North Sensor Web frameworks (http://52north.org/swe). The PSS is a web service which is constructed with two subservices, namely, future prediction service (FPS) and fuzzy inference service (FIS). The functional flow of PSS has been illustrated in Figure 3. The FPS reads metrics from MS data base for the arrival rate and response time of last interval time and predicts the future arrival rate and future response time using PD...
and ED, respectively. The FIS senses/observes the closeness or proximity or nearness between SLA response times and predicted future interval response time and determines the requirement of additional replication. It returns the flag of replication requirement “Y” or “N” or “S” to requested service (IRCM).

4. Processes and Techniques

The vision of sensor web is being pursued by the Sensor Web Enablement (SWE) working group of OGC. The PSS has been built with 52-degree North’s SOS 2.0 specification, and the web architecture of PSS implementation is shown in Figure 1. In the SWE architecture, application layer provides services to the clients to deliver processed data from service providers. Application layer can perform the functionality of centralized decision and control where an intelligent algorithm may run to evaluate the current state of the sensor networks and issue the autonomous control commands to the devices in the network. The IRCM resides on application layer which has service composition and execution logic to implement the web service replication intelligently. The IRCM has PSS client interface to configure the service level parameters (e.g., SLA time, execution interval time) for any specific web service (C_i) which has to respond to customer requests within agreed SLA time. Whenever the client (IRCM) requests for future interval availability of web service, the PSS can analyze the proximity and replication requirement of web service and returns the flag of replication requirement. Such a request from IRCM to PSS is processed by presentation layer and transmitted to service layer. The service layer has the details of service provider, service registry, information repository, and SWE’s SOS client information. The information repository contains the sensor description model named SensorML through which sensor processes or sensor systems (PSS) can make themselves known and discoverable. With the help of this service level information, service layer forwards the request to middleware layer which is responsible [7] for the communication and interaction of service layer with physical sensors/sensor services (PSS) and collects the sensed data from the sensors. Thus, PSS received the request from middleware and processes the request to sense the proximity of future interval response time of specific web services with the help of FPS and it determines the replication requirement with the help of FIS. The PSS’s outcome is responded back to middleware in turn to reach IRCM. IRCM further analyzes the PSS’s response with the help of RCS and RCA for the implementation of replication. The processes and techniques used in FPS, FIS, and RCS are explained below in detail and flow diagram is illustrated in Figure 4.

4.1. Future Prediction Service (FPS).

Upon the request from IRCM to PSS, the FPS reads the MS data base and observes last interval’s rate of arrival and response time of web service C_i and predicts the future interval arrival rate (demand) and response time. The arrival rate is the mean number of requests (λ) that arrive per unit of time (T_i). This can be parameterized with multiples of the monitoring pole interval. From the MS database, the FPS reads the number of requests that arrived within multiples of the monitoring pole interval. From the MS database, the FPS reads the number of requests that arrived within multiples of the monitoring pole interval.

The future arrival rate (λ_f) is calculated using

$$\lambda_f = \frac{n_i}{T_i}.$$  

(1)

The variables used in this paper are listed in Nomenclature.

When the system is running with multiple replica servers, the current arrival rate (CA) is calculated using

$$CA = \lambda_c = \sum_{i=1}^{n} \lambda_i$$  

(2)

From λ_c, it predicts the future arrival rate (λ_f) for the next interval using PD [26, page 168]:

$$\lambda_f = P(x) = \lambda_c x e^{-\lambda_c}.$$  

(3)

To find the future arrival rate, we use the mean of two probability confidence σ_u and σ_v. The future arrival rates FA_1 and FA_2 are predicted at probabilities σ_u and σ_v using (4) and (5), respectively [26, page 170]. Consider

$$FA_1 = F(x) = \sum_{i=0}^{x} \frac{e^{-\lambda_u} \lambda_u^i}{i!} \geq \sigma_u,$$  

(4)
where $\sigma_r$ is a known value (e.g., 0.9). Hence, $FA_1$ is calculated using (4) [26, page 170]. Similarly,

$$FA_2 = F(x) = \frac{\sum_{i=0}^{n-1} e^{-\lambda_f} \otimes \lambda_f^i}{i!} \geq \sigma_e,$$  

(5)

where $\sigma_e$ is a known value (e.g., 0.1). Hence, $FA_2$ is calculated using (5) [26, page 170].

The total future arrival rate (FA) is calculated using (6) [26, page 222]:

$$\lambda_f = FA = [([FA_1 \otimes \sigma_a] \oplus [FA_2 \otimes \sigma_r])].$$  

(6)

The response time is the time taken by a web service to completely process one request. It can also be defined as the time difference between the time in which a request is submitted and the time in which the response is received [24]. From the MS database, the FPS reads the number of requests processed ($n_q$) and response time of each request ($T_i^q$) at the end of every interval ($T_i'$) (e.g., 60 seconds). The average response time ($T_c'$) (in milliseconds (ms)) from an individual server is calculated using (7). When system has multiple servers, the total average current response time (CR) is calculated using (8) and response rate $\mu_c$ is calculated using (9). Therefore,

$$T_c' = \frac{\sum_{i=1}^{n_q} T_i^q}{n_q},$$  

(7)

$$CR = T_c = \frac{\sum_{i=1}^{n_q} T_i^q}{n_q},$$  

(8)

$$\text{Current response rate } \mu_c = \frac{T_j}{T_c}.$$  

(9)

From $T_c'$, FPS predicts the future response time (FR) using ED [26, page 172]

$$P(X = x) = \left\{ T_c \otimes e^{-T_c,\text{max}} \right\}. $$  

(10)

We use the mean of two probability confidences $\theta_j$ and $\theta_K$ to find the future response time. The CDF (cumulative distribution function) of the future response time ($FR_1$) is predicted for the probability confidence $\theta_j$ using (11), where $\theta_j$ is a known probability confidence value (e.g., 0.75). Hence, $FR_1$ is calculated using (11) to (11b) [26, page 172]. Consider

$$P(X = x) = \sum_{i=1}^{n} \left\{ T_c \otimes e^{-T_c,\text{max}} \right\} \geq \theta_j,$$  

(11)

$$F(x) = \left\{ \int_0^x T_c \otimes e^{-T_c,\text{max}} dt \right\} = 1 - e^{-T_c,\text{max}} \leq \theta_j,$$  

(11a)

$$FR_1 = x = [-T_c \ln (1 - \theta_j)].$$  

(11b)

Similarly, the future response time ($FR_2$) is predicted at probability confidence $\theta_K$ (e.g., 0.25) using (12) to (12b) [26, page 172]. Consider

$$P(X = x) = \sum_{i=1}^{n} \left\{ T_c \otimes e^{-T_c,\text{max}} \right\} \geq \theta_K,$$  

(12)

$$F(x) = \left\{ \int_0^x T_c \otimes e^{-T_c,\text{max}} dt \right\} = 1 - e^{-T_c,\text{max}} \leq \theta_K,$$  

(12a)

$$FR_2 = x = [-T_c \ln (1 - \theta_K)].$$  

(12b)

The total future response time (FR) (in ms) is calculated using [26, page 222]

$$FR = \left\{ FR_1 \otimes \theta_j \right\} \oplus \left\{ FR_2 \otimes \theta_K \right\}.$$  

(13)

4.2. Fuzzy Inference Service (FIS). The FIS analyzes the predicted arrival rate and response time and evaluates them against various fuzzy rules and a replication decision matrix. It maps these nonlinear input data (FA and FR) to a scalar output (the decision that replication is required or not) which is done using FL system. There are four key processes involved in a FL system [27] to determine the replication requirement decision: (1) fuzzifier, (2) fuzzy rules, (3) fuzzy inference engine, and (4) defuzzifier. In the fuzzy system, the concept of the linguistic variable is used to represent the input or output variable that carries the linguistic label [28]. The linguistic label is generally decayed into a set of linguistic terms. The linguistic variables of $\lambda_f$ and $T_f$ are defined in Tables 1 and 2, respectively.

| Fuzzy set | Crisp input range | Linguistic variable |
|-----------|-------------------|---------------------|
| $\alpha_1$ | $0 \to \omega - (\omega \ast 0.6)$ | Very Very Low (VVL) |
| $\alpha_2$ | $\omega - (\omega \ast 0.8)$ to $\omega - (\omega \ast 0.4)$ | Very Low (VL) |
| $\alpha_3$ | $\omega - (\omega \ast 0.6)$ to $\omega$ | Low (LW) |
| $\alpha_4$ | $\omega - (\omega \ast 0.4)$ to $\omega + (\omega \ast 0.4)$ | Normal (NR) |
| $\alpha_5$ | $\omega + (\omega \ast 0.4)$ to $\omega + (\omega \ast 0.6)$ | High (HG) |
| $\alpha_6$ | $\omega + (\omega \ast 0.6)$ to $0$ | Very Very High (VVH) |

| Fuzzy set | Crisp input range | Linguistic variable |
|-----------|-------------------|---------------------|
| $\beta_1$ | $0 \to \omega - (\omega \ast 0.6)$ | Very Very Low (VVL) |
| $\beta_2$ | $\omega - (\omega \ast 0.8)$ to $\omega - (\omega \ast 0.4)$ | Very Low (VL) |
| $\beta_3$ | $\omega - (\omega \ast 0.6)$ to $\omega$ | Low (LW) |
| $\beta_4$ | $\omega - (\omega \ast 0.4)$ to $\omega + (\omega \ast 0.4)$ | Normal (NR) |
| $\beta_5$ | $\omega + (\omega \ast 0.4)$ to $\omega + (\omega \ast 0.6)$ | High (HG) |
| $\beta_6$ | $\omega + (\omega \ast 0.6)$ to $0$ | Very Very High (VVH) |

The ranges of the fuzzy variables are determined using the triangular method [29]. The linguistic values of arrival rate are converted into triangular fuzzy numbers as shown in Figure 5 and Table 1, where "$\omega$" is 50% of present processing capability ($\rho$). $\rho$ is calculated as $\rho = n_s \ast N_p$, where $n_s$ is number of servers and $N_p$ is number of requests that each server can process within SLA time (e.g., 2500 requests/min). Correspondingly, the linguistic values of response time are converted into triangular fuzzy numbers as shown in Figure 6 and Table 2. Here, "$\psi$" is 50% of SLA response time (e.g., 650) [30]. The fuzzy rules are functions in the form of a series of "if-then-else" statements. Each of the two fuzzy sets (arrival rate, response time) has seven membership functions. The arrival rate and response time are termed as conditional attributes and form 49 ($7^2$) fuzzy decisional matrix elements which
Table 2: Linguistic variables for FR.

| SL | Crisp input range | Linguistic variable |
|----|-------------------|---------------------|
| $\beta_1$ | $0$ to $\psi - (\psi * 0.6)$ | Very Very Fast (VVF) |
| $\beta_2$ | $\psi - (\psi * 0.8)$ to $\psi - (\psi * 0.4)$ | Fast (FS) |
| $\beta_3$ | $\psi - (\psi * 0.6)$ to $\psi$ | Normal (NR) |
| $\beta_4$ | $\psi - (\psi * 0.4)$ to $\psi$ | Slow (SL) |
| $\beta_5$ | $\psi$ to $\psi + (\psi * 0.6)$ | Very Slow (VS) |
| $\beta_6$ | $\psi + (\psi * 0.4)$ to $\psi$ | Very Very Slow (VVS) |
| $\beta_7$ | $\psi + (\psi * 0.6)$ to $0$ | Very Very Very Slow (VVH) |

Table 3: Replication matrix.

| FA   | VVL | VL | LW | NR | LV | VL | VVH |
|------|-----|----|----|----|----|----|-----|
| VVS  | Y   | Y  | Y  | Y  | N  | N  | N   |
| VS   | Y   | Y  | Y  | N  | N  | N  | N   |
| SL   | Y   | Y  | Y  | N  | N  | N  | N   |
| NR   | Y   | N  | N  | N  | S  | S  | S   |
| FS   | N   | N  | N  | N  | S  | S  | S   |
| VF   | N   | N  | N  | N  | S  | S  | S   |
| VVF  | N   | N  | N  | S  | S  | S  | S   |

Figure 5: Fuzzy sets ($\alpha$) for future arrival rate.

Figure 6: Fuzzy sets ($\beta$) for future response time.

are summarized in Table 3. The replication requirement is termed as the decision attribute. The decision attribute has three categorical values: “Y” (replication required), “N” (replication not required), and “S” (stop existing replica). These decision attributes are arrived at by the FL decisions which will be used further by RCA. In our framework, we use the center of gravity method to defuzzify the strength of rule evaluation to identify the crisp output. This method is illustrated in

$$\delta = \frac{\sum_{i=1}^{n} m_i \cdot \omega_i}{\sum_{i=1}^{n} m_i}.$$  (14)

4.3. Resource Control Service and Its Algorithm. The RCS acts as actuator and performs the following activities: (1) it incorporates the FL decision; (2) it performs health checks of servers and web services; (3) it creates new virtual machines (VM) similar to any replica and powers them on or off; (4) it adds or removes any server from the load balancer; and (5) it starts or stops any server or web service. To analyze the consistency of the system behavior, the RCS uses the four time parameters: (1) “adm.T_STOP,” the administrator defined waiting time (for the RCS) after the FL decision is set to “S” (e.g., 900 seconds); (2) “adm.T_REPL,” the administrator defined waiting time (for the RCS) after the FL decision is set to “Y” (e.g., 300 seconds in our tests); (3) “Y.T_WAIT,” the waiting time of the system after the FL decision is set to “Y”; and (4) “S.T_WAIT,” the waiting time of the system after the FL decision is set to “S.” The “adm.T_STOP” and “adm.T_REPL” parameter values are set by an administrator. The FL decision and these time parameters are applied in the RCA (Pseudocode 1) to implement the FL decision. If the FL decision is “Y,” it will check for $T_f \geq 90\%$ SLA; if true, it implements the replication immediately; otherwise, it continues for the next interval until $Y.T_{\text{WAIT}} \geq \text{adm}_{T_{\text{REPL}}}$; then, it implements the new replica ($R_y$). Once the replication has been done, the RCS will add the new replica server into the load balancer to distribute the request to the new replica service. If the FL decision is “N,” it will continue for the next interval analysis. If the FL decision is “S,” it will continue with next interval until adm.T_STOP; then, it stops server ($R_s$) which provides higher response time. Also, the RCS periodically pings the servers and checks services to determine the health status at every $T_h$ interval (e.g., 10 seconds). If any of the web services is found to be hanging, unresponsive, or stopped, it is restarted. If the restart fails, the RCS sets the flag “Service_Status_Abnormal” to “Yes” and stop sending the request to it and replicates new server. Also, RCS evaluates the performance of each server periodically; if any server’s response time exceeds 90% of the SLA value, RCS replicates a new replica ($R_y$) and stops sending the request to the lower performance replica ($R_s$), and it removes the ($R_s$) from the load balancer once all the existing requests are completed. Then, administrator can investigate the problem servers.

5. Evaluation and Discussion

5.1. Environment. This section explains the test environment to evaluate the performance of FAPFEA and analyze
the results. The test environment was set up with virtual machines on Hyper-V with the Windows 2005 server platform with 4 GB RAM and MS-SQL Server 2005. The software tools “Applications Manager” from Manage Engine, SoapUI from SmartBear, and “ACE (Application Control Engine)” from Cisco were used for monitoring, testing, and load balancing purposes, respectively. Tomcat server and Eclipse were used for application development and implementation. We employed web service, namely, the customer status enquiry service (C1) on each of the replication servers R1 ⋯ Rn which were connected through a LAN (local area network) with a bandwidth of 100 mbps. This web service retrieves the status (e.g., active, closed, dormant, and inactive) of the customer. The SLA time defined for C1 is 1300 ms; concurrent HTTP requests were initiated from seven client machines with the frequency ranging from 1000 to 19,000 requests/minute using SoapUI. The number of concurrent HTTP requests is represented with threads every 60 seconds. A laptop with Core I-5 2.2 GHz 4 GB Ram was used as SOS Station which was connected with an access point to join the same network to the FAPFEA system. To enable Internet access, 3G modem was used via a USB interface. Several tests were carried out by turning on the system and starting operation.

5.2. Experiments. The aim of these test scenarios is to evaluate the ability of our proposed framework to sense the proximity of future interval response time with agreed SLA and determine the replication requirement to automatically enhance web service availability through replication and reclamation of the resources based on demand with minimal intervention from the system administrator. Table 4 provides the overview of the test scenarios we conduct in this paper.

5.2.1. Conventional System Test. This test has been conducted to understand the behavior and performance of the conventional system where there are no replication techniques facilitated. Here, two servers were engaged with web service C1 and the HTTP requests sent to web services, ranging from 1000 requests/minute to 8,000 requests/minute. The CA, CR data has been collected and shown in Table 5 and graphical representation of the test results is shown in Figure 7. It can be inferred from the data and graph that when the arrival rate is greater than 6000 requests/minute, the response time exceeds SLA time and the request failure rate increases too. Since there is no replication technique incorporated in the conventional system, it does not have the ability to replicate new web service automatically. Hence, it is found to be incapable of processing all the requests within expected SLA time. So the web service is considered as unavailable when web service response time exceeds SLA time.
Table 6: Experimental results of reliability test.

| CA (in Nos.) | FA (in Nos.) | CR (in ms) | FR (in ms) | Time (in min) | Servers (in Nos.) | SLA (in ms) |
|-------------|-------------|------------|------------|---------------|------------------|------------|
| 7000        | 7880        | 880        | 978        | 5             | 4                | 1300       |
| 7000        | 7880        | 880        | 978        | 10            | 4                | 1300       |
| 7000        | 7880        | 880        | 978        | 15            | 4                | 1300       |
| 7500        | 8380        | 1004       | 1113       | 20            | 4                | 1300       |
| 7500        | 8380        | 1004       | 1113       | 25            | 5                | 1300       |
| 7500        | 8380        | 916        | 1018       | 30            | 5                | 1300       |
| 7500        | 8380        | 811        | 902        | 35            | 5                | 1300       |
| 8000        | 8970        | 870        | 967        | 40            | 5                | 1300       |
| 8000        | 8970        | 870        | 967        | 45            | 5                | 1300       |
| 8000        | 8970        | 870        | 967        | 50            | 5                | 1300       |
| 9000        | 9960        | 1071       | 1184       | 55            | 5                | 1300       |
| 9000        | 9960        | 893        | 1004       | 60            | 6                | 1300       |
| 9000        | 9960        | 840        | 934        | 65            | 6                | 1300       |
| 10000       | 11040       | 978        | 1087       | 70            | 6                | 1300       |
| 10000       | 11040       | 978        | 1087       | 75            | 7                | 1300       |
| 10000       | 11040       | 896        | 996        | 80            | 7                | 1300       |

5.2.2. Reliability Test for Dynamic Replication. The aim of this test is to ascertain the reliability of the proposed framework for sensing the proximity of future interval response time with defined SLA time and replicate the service dynamically. To test this scenario, the HTTP requests were sent from client machines to $C_1$ at frequencies ranging from 7000 to 10,000 requests/minute. The $\lambda_c$ and $T_c$ data are collected and calculated from the MS metrics and $\lambda_f$ and $T_f$ are calculated using (1) to (13). $\lambda_f$ and $T_f$ are then evaluated against fuzzy rules and defuzzified using the center of gravity method in (14). These test results are summarized in Table 6 and shown in Figure 11. By virtue of the fuzzy decision output, the “FL_flag” is set to “Y,” “N,” or “S” depending on the loads against the response time. As the FL decision varies with time, the RCS applies the FL decision in the RCA and implements the new replication or removes the replication as shown in Figure 11. For instance, here, the prediction processes, calculations, fuzzy decision, and algorithms used in our framework are explained for $\lambda_c$ at 7500 requests/minute and $T_c$ at 1004.

The FPS predicts the $FA_1$ using (4) with $\sigma_u = 90\%$:

$$FA_1 = F(x) = \sum_{i=0}^{n} \frac{e^{-\lambda_c} \times \lambda_c^i}{i!} \geq 0.90,$$

$$F(x) = [P(X = 1) + P(X = 2) + P(X = 3) + \ldots ] \geq 0.90.$$  

(15)

From the above equations, $FA_1$ is derived by summing up $P(X)$ until being greater than 0.90, where $x = 1$ to $n$; hence, $FA_1$ is calculated as 8600 requests/minute.

Similarly, $FA_2$ is calculated using (5) with $\sigma_v = 10\%$:

$$FA_2 = F(x) = \sum_{i=0}^{n} \frac{e^{-\lambda_c} \times \lambda_c^i}{i!} \geq 0.10,$$

(16)

where $FA_2 = 6400$ requests/minute.

The total future arrival rate is calculated using (6):

$$FA = 8600 \times 0.9 + 6400 \times 0.10 = 7740 + 640 = 8380 \text{ requests/minute.}$$

(17)

The $FR_1$ is calculated using (11) to (11b) with $\theta_f = 75\%$:

$$FR_1 = [-T_c \times \ln (1 - \sigma_j)],$$

$$FR_1 = [-1004 \times \ln (1 - 0.75)] = 1391.8395 \text{ ms.}$$

(18)

Similarly, the $FR_2$ is calculated using (12) to (12b) with $\theta_K = 25\%$:

$$FR_2 = 288.8328 \text{ ms.}$$

(19)
Table 7: Linguistic variable of FA.

| SN | Fuzzysets | Input | Output set | Output |
|----|-----------|-------|------------|--------|
| α1 | VVL       | 0     | 1000       | 0      |
| α2 | VL        | 0     | 2000       | 0      |
| α3 | LW        | 0     | 3000       | 0      |
| α4 | NR        | 0     | 5000       | 0      |
| α5 | HG        | 0     | 7000       | 0      |
| α6 | VH        | 0.62  | 8000       | 4960   |
| α7 | VVH       | 0.38  | 9000       | 3420   |

The total future response time (FR) is calculated using (13):

\[
FR = T_f = 1391.8395 \times 0.75 + 288.8328 \times 0.25 = 1043.8796 + 72.2082 = 1116.0878 \text{ ms.} \tag{20}
\]

The predicted FA is converted into fuzzy sets using triangular method as shown in Figure 8, and respective linguistic values are represented in Table 7. Similarly, FR is converted into fuzzy sets and represented in Figure 9 and respective linguistic values are shown in Table 8.

Evaluation of the rules is as follows:

\[
\gamma_1 = \text{Max}(0.39, 0.61, 0, 0) = 0.61,
\gamma_2 = \text{Max}(0, 0, 0, 0, 0, 0, 0) = 0,
\gamma_3 = \text{Max}(0, 0, 0, 0) = 0. \tag{21}
\]

Using (14), the fuzzified values are defuzzified and illustrated in Figure 10:

\[
\delta = \frac{450 \times 0.61}{5 \times 0.61} = \frac{274.5}{3.05} = 90. \tag{22}
\]

\(\delta = 90\) signifies that replication is required and the FL flag is set to “Y.” The RCS analyzes the FL decision by applying RCA to determine the urgency of the replication. Since \(Y < Y_{\text{WAIT}} \leq \text{adm}_{\text{REPL}}\) it waits till \(Y_{\text{WAIT}} \geq \text{adm}_{\text{REPL}}\) (i.e., 300 seconds); then, the RCA replicates the new server (\(R_f\)) with \(C_1\). Thus, the load balancer starts distributing the load to the new replica. Similarly, when \(\lambda_f\) of \(C_1\) reaches 9,000 requests/minute and the future response time exceeds 90% of SLA, without waiting for adm_{REPL}, the RCA immediately replicates the new server \((R_q)\) with web service \(C_1\). Thus, the response time of web service is maintained by our proposed framework even during heavy load periods by replicating new servers dynamically when required as shown in Figure 11.

5.2.3. Scalability Test for Continuous Overload. This test is to ascertain the scalability of the proposed framework when the load on the web service increases continuously. In order to test this scenario, we activated four servers with web service \(C_1\) and HTTP requests were sent to \(C_1\) at frequencies ranging from 7000 to 120000 requests/minute. The \(\lambda_f\) and \(T_f\) data are collected and calculated from the MS metrics and \(\lambda_f\)
and $T_f$ are calculated using (1) to (13) and evaluated against fuzzy rules and defuzzified using (14) as we described in Section 5.2.2. These test results are summarized in Table 9 and shown in Figure 12. By virtue of the fuzzy decision output, the “FL_flag” is set to “Y,” “N,” or “S” depending on the loads against the response time. As the FL decision varies with time, the RCS applies the FL decision in the RCA and implements the new replication as shown in Figure 12. Thus, the future interval response time is sensed accurately and replicated a new replica when required to provide uninterrupted web service availability.

5.2.4. Sustainability Test for Randomly Varying Load Scenario. The aim of this test is to ascertain the ability of the proposed framework to maintain the response time of web service within SLA time when the system is under circumstances where it receives the randomly varying load. To test this scenario, the HTTP requests were sent from client machines to $C_1$ at frequencies ranging from 2000 to 19,000 requests/minute by randomly stepping the load up or down every minute and $adm . T_{REP}$ are set to 0 seconds to replicate the new service immediately. In total, approximately 163,000 requests were generated and processed by $C_1$ for this test. The $\lambda_c$ and $T_{c}$ data are collected and calculated from the MS metrics and $\lambda_f$ and $T_f$ are calculated using (1) to (13) and evaluated against fuzzy rules and defuzzified using (14) as described in Section 5.2.2. These test results are summarized in Table 10 and shown in Figure 13. By virtue of the fuzzy decision output, the “FL_flag” is set to “Y,” “N,” or “S” depending on the loads against the response time. As the FL decision varies with time, the RCS applies the FL decision in the RCA and implements the new replication as shown in Figure 13. For example, the system received a lower load and normal response time after the 14th min (19,000 requests/minute) and the RCS waits for the $adm . T_{STOP}$ period to stop the replica. Hence, after $adm . T_{STOP}$ (300 seconds) of continuous low load, the RCS stops one of the replicas. Thus, FAPFEA implements the new replication or reclaims the excessive servers when required to provide uninterrupted web service availability.

5.2.5. Comparison Test of Replication Timing. To validate the efficiency of FAPFEA, we compare it with the linear regression model (LRM) suggested by [23] and Holt’s linear exponential smoothing model (HLES) suggested by [21]. The comparison test analyzes the number of replicated web services provided by each model to process the same load. The LRM predicts the future load based on data from the past 15 days and replicates the web service accordingly to serve for the 16th day. The HLES approach computes the response time using the queuing network model, predicts the response time using HLES, and replicates web services when the predicted response time exceeds 90% of SLA. The replication techniques of these two existing models are computed mathematically and compared with FAPFEA for various loads varying from 2000 requests/minute to 3500 requests/minute. The results are compared and shown in Figure 14. This indicates that the FAPFEA predicts the replication requirement accurately and at the most appropriate time compared with these two existing models. This entails lower maintenance requirements, less electric power consumption, and less operational cost with the use of our approach.

5.3. Prediction Error. To evaluate the prediction accuracy of our framework, we used the common approaches of mean absolute error (MAE) and root mean squared error (RMSE) [31]. MAE is used to calculate the average magnitude of errors. The “RMSE is a quadratic scoring rule which is measuring average magnitude of the error” [32]. Both approaches can have values ranging from zero to $\infty$ (where lower values

| SN | Fuzzysets | Input | Output set | Output |
|----|-----------|-------|------------|--------|
| $\beta_1$ | VVF | 0 | 130 | |
| $\beta_2$ | VF | 0 | 260 | 0 |
| $\beta_3$ | FS | 0 | 390 | 0 |
| $\beta_4$ | NR | 0 | 650 | 0 |
| $\beta_5$ | SL | 0 | 910 | 0 |
| $\beta_6$ | VS | 0.39 | 1040 | 408.72 |
| $\beta_7$ | VVS | 0.61 | 1170 | 707.85 |

Figure 12: Scalability test for continuous over load.

Figure 13: Sustainability test for randomly varying load.

Table 8: Linguistic variable of FR.
Table 9: Experimental results of scalability test.

| CA (in Nos.) | FA (in Nos.) | CR (in ms) | FR (in ms) | Time (in min.) | Servers (in Nos.) | SLA (in ms) |
|-------------|-------------|------------|------------|----------------|------------------|-------------|
| 7000        | 7880        | 880        | 978        | 5              | 4                | 1300        |
| 7500        | 8380        | 1004       | 1113       | 10             | 5                | 1300        |
| 8000        | 8970        | 870        | 967        | 15             | 5                | 1300        |
| 8500        | 9500        | 946        | 1046       | 20             | 5                | 1300        |
| 9000        | 10000       | 1021       | 1121       | 25             | 5                | 1300        |
| 9500        | 10500       | 1101       | 1184       | 30             | 6                | 1300        |
| 10000       | 11000       | 893        | 993        | 35             | 6                | 1300        |
| 10500       | 11500       | 967        | 1067       | 40             | 6                | 1300        |
| 11000       | 12000       | 1052       | 1152       | 45             | 6                | 1300        |
| 11500       | 12500       | 1171       | 1271       | 50             | 7                | 1300        |
| 12000       | 13000       | 896        | 996        | 55             | 7                | 1300        |

Table 10: Experimental results of sustainability test.

| CA (in Nos.) | FA (in Nos.) | CR (in ms) | FR (in ms) | Time (in min.) | Servers (in Nos.) | SLA (in ms) |
|-------------|-------------|------------|------------|----------------|------------------|-------------|
| 2000        | 2480        | 652        | 730        | 1              | 2                | 1300        |
| 4000        | 4640        | 978        | 1087       | 2              | 3                | 1300        |
| 6000        | 6800        | 935        | 1039       | 3              | 3                | 1300        |
| 8000        | 3560        | 640        | 711        | 4              | 3                | 1300        |
| 2000        | 2480        | 465        | 516        | 5              | 3                | 1300        |
| 7000        | 7880        | 913        | 1014       | 6              | 4                | 1300        |
| 14000       | 16160       | 998        | 1109       | 7              | 5                | 1300        |
| 10000       | 11040       | 836        | 929        | 8              | 5                | 1300        |
| 18000       | 19920       | 993        | 1103       | 9              | 6                | 1300        |
| 8000        | 8970        | 690        | 767        | 10             | 6                | 1300        |
| 7000        | 7880        | 576        | 640        | 11             | 6                | 1300        |
| 10000       | 11040       | 827        | 943        | 12             | 6                | 1300        |
| 16000       | 18010       | 921        | 1011       | 13             | 6                | 1300        |
| 19000       | 20990       | 1034       | 1197       | 14             | 7                | 1300        |
| 13000       | 15080       | 729        | 823        | 15             | 7                | 1300        |
| 12000       | 13130       | 581        | 645        | 16             | 7                | 1300        |
| 8000        | 314         | 411        | 19         | 6              | 6                | 1300        |
| 4000        | 4640        | 643        | 719        | 20             | 6                | 1300        |

The overall prediction error of FAPFEA is evaluated using

\[
\text{MAE} = \frac{\sum_{i=1}^{n} \left| P_i - \hat{P}_i \right|}{N},
\]

\[
\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{n} (P_i - \hat{P}_i)^2}{N}}.
\]

The MAE and RMSE values are compared with those of HLESM and LRM. 500 request samples were collected and averaged to 10 samples which are presented in Figure 15. The graph shows that the RMSE value for FAPFEA reached a maximum of 18% for sample 10 and the MAE value reached a maximum of 17% at sample 6. By comparison, the
maximum RMSE and MAE values for HLESM were 23% at sample 6 and 20% at sample 8, respectively. The maximum RMSE and MAE values for LRM were 25% at sample 10 and 22% maximum at sample 7, respectively. Thus, the FAPFEA prediction accuracy is demonstrated to be more reliable than the existing methods.

The results of the above tests signify that our proposed framework is efficient in dynamically replicating the web service on-demand, improving the availability of web services to respond to client requests within the defined SLA time for all constant, varied, and robust load conditions.

6. Conclusion and Future Work

In this paper, we present a sensor web based adaptive prediction framework for improving the availability of web services. Our approach uses the sensor web service PSS to sense the proximity of the predicted future interval response time with SLA response time of web service using PD and ED, respectively. FL is used to determine the replication requirement. IRCM passes the FL decision to RCS for implementing the FL decision that replicates the web service on another server host whenever the predicted response time violates the SLA or claims the existing excess resource. In addition, our framework detects the failure of any web service or server and attempts to restart it. In case of web service failure, it replicates the web service on another server. Also, if the response time of any individual server is approaching the threshold limit, it replicates the web service on another server and stops the replica with lower performance.

We conducted several testing scenarios, demonstrating that our framework replicates the web service and claims the unnecessary resources dynamically based on demand with 82% accuracy. The comparison results of two existing models (LRM and HLESM) show that FAPFEA predicts the replication requirement more accurately and at the most appropriate time. Thus, FAPFEA facilitates a high availability with minimal administrator intervention and less maintenance and operational cost. The advantages of our proposed framework include that it is proactive, dynamic, on-demand, and accurate. We thereby achieve web service availability by maintaining the response time within SLA and by maintaining the continuous (uninterrupted) availability of web services to provide uninterrupted responses, respectively. Because our framework meets basic attributes such as sustainability, scalability, and elasticity, it provides a better solution to the current internet business world.

Our future work will be directed towards implementing our framework on cloud computing web services and analyzing its performance for large scale applications. One limitation of FAPFEA is that it is capable of one replication for one interval time. If the system requires two or more replications, the other replication will occur after the $\text{adm}_{\text{REPL}}$ time. We will be addressing this issue in future papers.

**Nomenclature**

- $R_n$: “n”th servers
- $C_i$: Web services
- $T_i$: Statistics read interval
- $T_{pr}$: Previous status read time
- Y: Replication required is “Yes”
- N: Replication required is “No”
- S: “Stop” existing replica
- $P_i$: Measured arrival rate
- $\hat{P}_i$: Predicted arrival rate
- $m^i$ : Membership degree of rule “i”
- $w^i$: Weightage of membership “i”
- $x$: Random variable
- $e$: Euler’s number
- $T_i$: The average current response time
- $T_{f}$: Future response time (FR)
- $n^i$: Number of requests
- $n_s$: Number of servers
- $T^c$: Current response time of server “R”
- $T^q$: Response time of each request
- $n^q$: Number of requests that arrived to server “R”
- $T_{h}$: Health status check time interval.

**Greek Symbols**

- $\mu$: Average response rate
- $\mu_c$: Average current response rate
- $\gamma^1$: Fuzzy variable of replication rule 1
- $\gamma^2$: Fuzzy variable of replication rule 2
- $\gamma^3$: Fuzzy variable of replication rule 3
- $\alpha_i$ to $\alpha_f$: Fuzzy set variables of FA
- $\beta_i$ to $\beta_f$: Fuzzy set variables of FR
- $\lambda$: Mean number of requests that arrived/$T_i$
- $\lambda^c$: Current arrival rate (CA) per $T_i$
- $\lambda_f$: Future arrival rate
- $\sigma^1$: Probability confidence for FA
- $\sigma^2$: Probability confidence for FA
- $\sigma^3$: Probability confidence for FA
- $\sigma^4$: Probability confidence for FA
- $\sigma^5$: Probability confidence for FA
- $\lambda^c$: Current arrival rate of server “R”
\( \omega \): Value of 50% of arrival rate
\( \psi \): Value of 50% of SLA response time
\( \delta \): Defuzzified crisp output.

**Conflict of Interests**

The authors declare that there is no conflict of interests regarding the publication of this paper.

**References**

[1] B. Benatallah, Q. Z. Sheng, and M. Dumas, “The self-serv environment for web services composition,” IEEE Internet Computing, vol. 7, no. 1, pp. 40–48, 2003.

[2] M. Abd-El-Barr, Design and Analysis of Reliable a Fault-tolerant Computer Systems, Imperial College Press, London, UK, 2007.

[3] K. Birman, R. Van Renesse, and W. Vogels, “Adding high availability and autonomic behavior to web services,” in Proceedings of the 26th IEEE International Conference on Software Engineering (ICSE ’04), pp. 17–26, Edinburgh, UK, May 2004.

[4] Q. Z. Sheng, Z. Maamor, and J. Yu, “Robust web services provisioning through on-demand replication,” in Information Systems: Modeling, Development, and Integration, vol. 20 of Lecture Notes in Business Information Processing, pp. 4–16, Springer, Berlin, Germany, 2009.

[5] L. Di, “Geospatial sensor web and Self-adaptive Earth Predictive Systems (SEPS),” in Proceedings of the Earth Science Technology Office (ESTO)/Advanced Information System Technology (AIST) Sensor Web Principal Investigator (PI) Meeting, pp. 1–4, San Diego, Calif, USA, 2007.

[6] M. Botts, G. Percivall, C. Reed, and J. Davidson, “OGC© sensor web enablement: overview and high level architecture,” in GeoSensor Networks, vol. 45/40 of Lecture Notes in Computer Science, pp. 175–190, Springer, Berlin, Germany, 2008.

[7] M. S. Khan and D. Kim, "Enhanced service-oriented open sensor web architecture with application server based mashup,” International Journal of Distributed Sensor Networks, vol. 2014, Article ID 313981, 10 pages, 2014.

[8] M. S. Kim, C. H. Lee, I. S. Jang, and K.-J. Li, “GeoSensorBase: integrating and managing huge number of heterogeneous sensors using sensor adaptors and extended SQL querying,” in Web and Wireless Geographical Information Systems: 13th International Symposium, WGIS 2014, Seoul, South Korea, May 29–30, 2014. Proceedings, vol. 8470 of Lecture Notes in Computer Science, pp. 100–114, Springer, Berlin, Germany, 2014.

[9] L. A. Zadeh, “Fuzzy logic and approximate reasoning,” Synthesis, vol. 30, no. 3, pp. 407–428, 1975.

[10] P. Horn, “Autonomic computing: IBM’s perspective on the state of information technology,” Tech. Rep., IBM Corporation, 2001.

[11] M. Agarwal, V. Bhat, H. Liu et al., “AutoMate: enabling autonomic applications on the grid,” in Proceedings of the Autonomic Computing Workshop, pp. 48–57, IEEE, Seattle, Wash, USA, June 2003.

[12] G. Folino and G. Spezzano, “An autonomic tool for building self-organizing Grid-enabled applications,” Future Generation Computer Systems, vol. 23, no. 5, pp. 671–679, 2007.

[13] A. G. Ganek and T. A. Corbi, “The dawning of the autonomic computing era,” IBM Systems Journal, vol. 42, no. 1, pp. 5–18, 2003.

[14] S. Kounев, R. Nou, and J. Torres, “Autonomic QoS-aware resource management in grid computing using online performance models,” in Proceedings of the 2nd International Conference on Performance Evaluation Methodologies and Tools & Workshops (ValueTools ’07), ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), Nantes, France, October 2007.

[15] S. Abraham, M. Thomas, and J. Thomas, “Enhancing web services availability,” in Proceedings of the 26th IEEE International Conference on e-Business Engineering (ICEBE ’05), pp. 352–355, Beijing, China, October 2005.

[16] J. Salas, F. Perez-Sorrosal, M. Patino-Martinez, and R. Jimenez-Peris, “WS-replication: a framework for highly available web services,” in Proceedings of the 15th International Conference on World Wide Web (WWW ’06), pp. 357–366, ACM, Edinburgh, UK, May 2006.

[17] N. C. Mendonça, J. A. F. Silva, and R. O. Anido, “Client-side selection of replicated web services: an empirical assessment,” Journal of Systems and Software, vol. 81, no. 8, pp. 1346–1363, 2008.

[18] R. Sundharam, M. Lakshmi, and D. Abarajithan, “Enhancing the availability of web services for mission critical applications,” in Proceedings of the Trends in Information Sciences & Computing (TISC ’10), pp. 149–151, IEEE, Chennai, India, December 2010.

[19] X. Ye and Y. Shen, “A middleware for replicated web services,” in Proceedings of the IEEE International Conference on Web Services (ICWS ’05), pp. 631–638, IEEE, July 2005.

[20] L. Liu, Z. Wu, Z. Ma, and Y. Cai, “A dynamic fault tolerant algorithm based on active replication,” in Proceedings of the 7th International Conference on Grid and Cooperative Computing (GCC ’08), pp. 557–562, Shenzhen, China, October 2008.

[21] M. K. Hussein and M. H. Mousa, “A framework for adaptive Qos of web services using replication,” International Journal of Computer Science and Communication Networks, vol. 2, no. 2, pp. 288–294, 2012.

[22] B.-Y. Ooi, H.-Y. Chan, and Y.-N. Cheah, “Dynamic service placement and replication framework to enhance service availability using team formation algorithm,” The Journal of Systems and Software, vol. 85, no. 9, pp. 2048–2062, 2012.

[23] M. F. Mohamed, H. F. ElYamany, and H. M. Nassar, “A study of an adaptive replication framework for orchestrated composite web services,” SpringerPlus, vol. 2, no. 1, p. 511, 2013.

[24] S.-Y. Hwang, H. Wang, J. Tang, and J. Srivastava, “A probabilistic approach to modeling and estimating the QoS of web-services-based workflows,” Information Sciences, vol. 177, no. 23, pp. 5484–5503, 2007.

[25] M. Conti, M. Kumar, S. K. Das, and B. A. Shirazi, “Quality of service issues in internet web services,” IEEE Transactions on Computers, vol. 51, no. 6, pp. 593–594, 2002.

[26] J. Banks, J. Carson, B. L. Nelson, and D. Nicol, “Chapter 5. Statistical models in simulations and chapter 6. Queuing models,” in Discrete-Event Systems Simulation, pp. 141–222, Prentice Hall, Upper Saddle River, NJ, USA, 4th edition, 2004.

[27] L.-X. Wang and J. M. Mendel, “Generating fuzzy rules by learning from examples,” IEEE Transactions on Systems, Man, and Cybernetics, vol. 22, no. 6, pp. 1414–1427, 1992.

[28] L. A. Zadeh, “Is there a need for fuzzy logic?” Information Sciences, vol. 178, no. 13, pp. 2751–2779, 2008.

[29] S. Murugan and V. Ramachandran, “Fuzzy decision making model for byzantine agreement,” Journal of Engineering Science and Technology, vol. 9, no. 2, pp. 206–219, 2014.
[30] P. Singhala, D. N. Shah, and B. Patel, “Temperature control using fuzzy logic,” *International Journal of Instrumentation and Control Systems*, vol. 4, no. 1, pp. 1–10, 2014.

[31] X. Luo, H. Luo, and X. Chang, “Online optimization of collaborative web service qos prediction based on approximate dynamic programming,” *International Journal of Distributed Sensor Networks*, vol. 2015, Article ID 452492, 9 pages, 2015.

[32] M. Silic, G. Delac, I. Krka, and S. Srbljic, "Scalable and accurate prediction of availability of atomic web services," *IEEE Transactions on Services Computing*, vol. 7, no. 2, pp. 252–264, 2014.
Submit your manuscripts at http://www.hindawi.com