A

# Achieving High Availability of Web Services Based on A Particle Filtering Approach

Lina Yao, School of Computer Science, the University of Adelaide, Australia
Quan Z. Sheng, School of Computer Science, the University of Adelaide, Australia
Zakaria Maamar, College of Information Technology, Zayed University, UAE

Guaranteeing the high availability of Web services is a significant challenge due to the varying number of invocation requests the Web services have to handle at a time, as well as the dynamic nature of the Web. The issue becomes even more challenging for composite Web services in the sense that their availability is inevitably dependent on corresponding component Web services. Current Quality of Service (QoS)-based selection approaches assume that the QoS of Web services (such as availability) is readily accessible and services with better availability are selected in the composition. Unfortunately, how to provide real-time availability information of Web services is largely overlooked. In addition, the performance of these approaches will raise questions when the pool of Web services to select from becomes large. In this paper, we tackle these problems by exploiting particle filtering-based techniques. In particular, we developed algorithms to accurately predict the availability of Web services and dynamically maintain a subset of Web services with higher availability ready to join service compositions. Web services can be always selected from this smaller space, thereby ensuring good performance in service compositions. Our implementation and experimental study demonstrate the feasibility and benefits of the proposed approach.

Categories and Subject Descriptors: H.3.5 [**Information Storage and Retrieval**]: Online Information Services—Web-based services

General Terms: Design, Algorithms, Performance

Additional Key Words and Phrases: Web service, service composition, service availability, service community, particle filtering

---

Authors' addresses: L. Yao and Q. Z. Sheng, School of Computer Science, The University of Adelaide, North Terrace, Adelaide, SA 5005, Australia. Email: {lina,qsheng}@cs.adelaide.edu.au; Z. Maamar, College of Information Technology, Zayed University, Dubai, UAE. Email: zakaria.maamar@zu.ac.ae.


DOI 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

## 1. INTRODUCTION

Web services and service-oriented computing (SOC) represent a paradigm for building distributed computing applications over the Internet. Unfortunately, after the development of nearly one decade, Web services are still in their infancy [Papazoglou et al. 2007; Hwang et al. 2007; Yu et al. 2008; Rosario et al. 2008; Sheng et al. 2010]. According to a recent study in Europe [Domingue and Fensel ], the Web currently contains 30 billion Web pages, with 10 million new pages added each day. In contrast, only 12,000 real Web services exist on the Web. Even worse, many Web services have been deployed with dependability problems (e.g., unexpected behaviors and lack of reliability and availability details). This presents a major hurdle for enterprises and government agencies seeking to embrace Web services as a development technology for their mission critical applications.

Guaranteeing the availability of a Web service is a significant challenge due to the varying number of invocation requests the Web service has to handle at a time, as well as the dynamic nature of the Web. Over the last few years, many works have emerged in addressing Web services availability problem. Almost all of these approaches are based on the concept of *service community* where Web services with similar functionalities (but different non-functional properties such as quality of service (QoS)) [Medjahed and Bouguettaya 2011; Benatallah et al. 2003; Zeng et al. 2003] are grouped in a particular "cluster". The basic idea on improving the availability of Web service in a composition is to substitute Web services with poor quality using peers with better quality from the same service community. This typically involves QoS based service selection.

Most QoS service selection approaches assume that the QoS information (e.g., availability of Web service) is pre-existing and readily accessible. This unfortunately is not the case in most real world applications. In reality, the availability status, as well as other QoS properties, of a Web service is highly uncertain, which changes over time. How to accurately estimate and predict the availability status of a Web service becomes an important research problem. In addition, given the wide adoption of Web service technologies in industry, more and more Web services will be available and the size of service communities will be inevitable large. Selecting from such a large space will inevitably lead into performance problems. Ideally, low quality Web services should be automatically filtered and not be considered during service composition.

The work in this paper focuses on solving the above problems, which is an extension of our work published in [Yao and Sheng 2011]. In particular, we propose a particle filter based approach to accurately predicate and adjust Web service availability in real time. We further propose an algorithm to dynamically filter low quality Web services from service communities so that service compositions have to deal with only partial set of high quality component services. As a result, our approach offers a more efficient and effective solution to service composition which ensures the

high availability of the generated composite Web services. In a nutshell, our contributions are as follows:

— A model for assessing Web services availability using particle filter technique, which can return precise and dynamic prediction of this availability. In our approach, the availability of a Web service combines both historical availability information and predicted availability.
— An algorithm to optimize Web services selection from a cluster of Web services (like service community where all services offer similar functionalities) by dynamically reducing the candidate Web services search space during Web services composition. Top Web services with high QoS can be always maintained for each service community, which are recommended to composite Web services, thereby not only ensuring the high availability of composite Web services, but also significantly improving the efficiency of Web service composition, and
— An implementation of a research prototype system using a number of state-of-the-art technologies. To validate the feasibility and benefits of our approach, we conducted extensive experimental studies.

The rest of the paper is organized as follows. Section 2 briefly introduces service availability model and the particle filter techniques. Section 3 describes the details of our approach and the algorithms. Section 4 reports on the implementation and some preliminary experimental results. Finally, Section 5 overviews related work and Section 6 provides some concluding remarks and future research directions.

## 2. BACKGROUND

In this section, we briefly discuss some basic concepts, namely *Web service community*, *Web service availability*, and *particle filtering*. We then present a motivating example.

### 2.1. Web Service Community

The concept of Web service community [Medjahed and Bouguettaya 2011; Zeng et al. 2003] is proposed to handle the large number and dynamic nature of Web services (e.g., emergence of new services and retraction of old ones) in a flexible way. A service community is a collection of Web services with a common functionality but different nonfunctional properties such as different providers and different QoS. Service communities provide descriptions of a desired functionality without referring to any potential service. When a community receives a request to execute an operation, the request is delegated to one of its current members based on appropriate selection strategies [Benatallah et al. 2003].

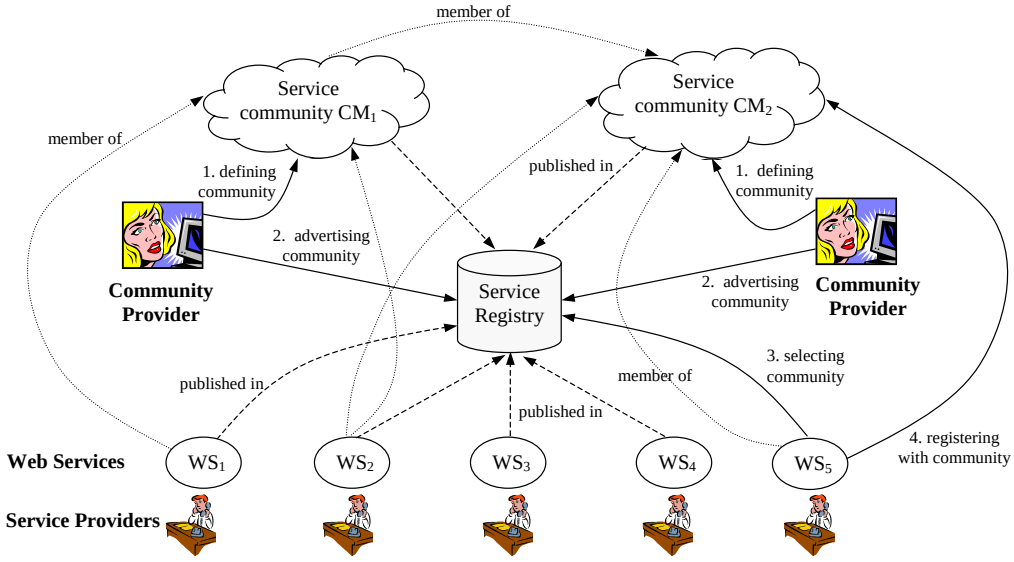



Fig. 1. Illustration of Web service communities

Figure 1 illustrates the basic process of creating a community and registering Web services with it. Communities are specified by community providers (step 1), who then register the communities with service registry (step 2). A community is a service that is created, advertised, discovered, and invoked in the same way that regular Web services are. Communities are published in a service registry (e.g., UDDI) so that they can be discovered by service providers. Service providers search the service registry to find the appropriate communities (step 3) and register their services (atomic or composite) with the communities (step 4). After registering with a community, a Web service becomes a member. It should be noted that a service community can be a member of another community.

### 2.2. Modeling Web Services Availability

There are different classifications of availability and many ways to calculate it [Elsayed 1996]. Almost all existing approaches (e.g., [Zeng et al. 2004; Liu et al. 2004; Guo et al. 2008]) use *operational* availability that measures the average availability over a period of time (i.e., the ratio of the service uptime to total time). Although this is simple to calculate, it is hard to measure the availability of a Web service at a specific time.

In this work, we model Web service availability as *instantaneous* (or point) availability. The instantaneous availability of a Web service $s$ is the probability that $s$ will be operational (i.e., up and running) at a specific time $t$. The following discusses how this is calculated.

At a given time $t$, a Web service $s$ will be available if it satisfies one of the following conditions:

— The Web service $s$ is working in the time frame of [0,$t$] (i.e., it never fails by time $t$). We represent the probability of this case as $\mathcal{R}(s,t) = \frac{\mathcal{T}_a(t)}{\mathcal{T}_s(t)}$, where $\mathcal{T}_a(t)$ is the total available time for each component Web service and $\mathcal{T}_s(t)$ is the total measurement time [Ran 2003].

— The Web service $s$ works properly since the latest repair that occurred at time $u$ ($0 < u < t$). The probability of this condition is $\int_0^t \mathcal{R}(s, t-u)m(s,u)du$, where $m(s,u)$ is the renewal density function of $s$. In our work, we model $m(s,u)$ as *Poisson* distribution $m(s,u) \sim \frac{e^{-\lambda}\lambda^k}{k!}$.

Based on these two conditions, the availability of $s$ at time $t$, $\mathcal{A}(s,t)$, is calculated using the following formula:

$$\mathcal{A}(s,t) = \mathcal{R}(s,t) + \int_0^t R(s,t-u)m(s,u)du \tag{1}$$

### 2.3. Particle Filtering

We consider the availability of Web services as a dynamic system (i.e., it changes over time), which can be modeled as two equations: *state transition* equation and *measurement* equation. The states can not be observed directly and need to be estimated, while the measurements can be observed directly. For a very simple example, if we track a robot, we can model its state as a vector including the robot's position and velocity $\{p, v\}$, and the observation of the position (i.e., measurements) can be obtained from the GPS. For Web services, [Guo et al. 2008] exploits Extended Kalman Filter to predict the Web service dependability state. In our work, we model the component Web service availability state as:

$$x_t = f_t(x_{t-1}, v_{t-1}) \tag{2}$$

where $f_t$ is a non-linear state transition function of the availability of a component Web service, $x_t, x_{t-1}$ are estimated and previous states of the component Web services respectively, and $v_{t-1}$ is the state noise in a non-Gaussian distribution (e.g., disturbance caused by network throughput to the Web service availability). Similarly, measurement equation for the component Web service availability is represented as

$$z_t = h_t(x_t, n_t) \tag{3}$$

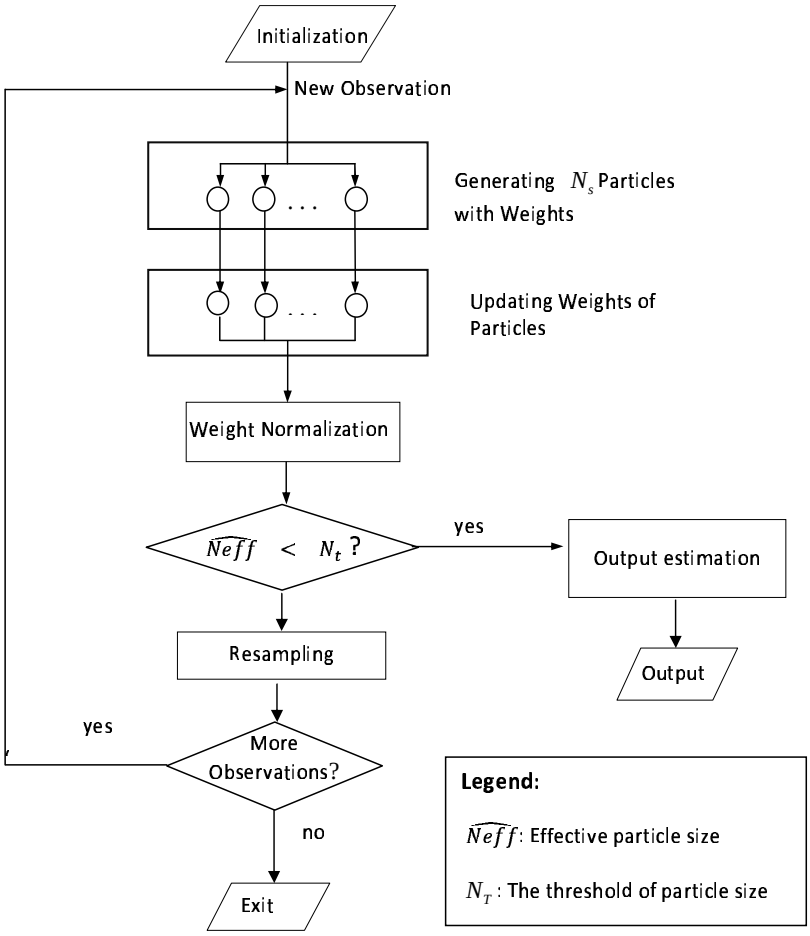



Fig. 2. The main processes of the particle filter technique

where $h_t$ is a non-linear measurement function, $z_t$ is a measurement, $x_t$ is the estimated Web service availability state, and $n_t$ is the measurement noise which is not confined as Gaussian distribution, (e.g., observation error).

The availability of Web services changes over time, which is full of uncertainty due to problems of network issues, hosting servers' loads, and even service requester environments. However, the state transition of availability from time $t-1$ to time $t$ can not be guaranteed as a linear transition, and in the measurement equation, the noise can also not be guaranteed as Gaussian distribution. We therefore propose to exploit the generic *particle filter* [Kitagawa 1996] to solve the dynamic availability of Web services. Particle filtering can deal with the non-linear and non-gaussian distribution situation presented in Web services, which will be detailed later.

**Algorithm 1:** Generic Particle Filter Algorithm

Let $\widehat{Neff}$ be the effective particle sample size and $\mathcal{N}_t$ be the threshold of the particle size.

**for** $i = 1 : \mathcal{N}_s$ **do**

Draw $x_t^i \sim q(x_t|x_{t-1}^i, z_t)$

Assign the particle a weight, $w_t^i$ according to $w_t^i \propto w_{t-1}^i \frac{p(z_t|x_t^i)p(x_t^i|x_{t-1}^i)}{q(x_t^i|x_{t-1}^i, z_t)}$, $q(\cdot)$ is a proposal function,and can be defined.

**end**

Calculate total weight: $t = \sum_{i=1}^{\mathcal{N}_s} w_t^i$

**for** $i = 1 : \mathcal{N}_s$ **do**

Normalize: $w_t^i = w_t^i$/t ;

**end**

Calculate $\widehat{Neff}$ using $\widehat{Neff} = \frac{1}{\sum_{i=1}^{\mathcal{N}_s}(w_t^i)^2}$

**if** $\widehat{Neff} < \mathcal{N}_t$ **then**

Resample (Algorithm 2).

**end**

The reasons backing particle filter adoption are as follows:

— Particle filters can represent arbitrary probability densities by a collection of particles with weight;
— Unlike Kalman filters, particle filters can converge to the true posterior even in non-Gaussian, non-linear dynamic systems; and
— Compared to grid-based approaches, particle filters are very efficient because they automatically focus their resources (particles) on regions in state space with high probability.

Briefly, the particle filter is a technique for implementing Bayesian filter recursively by Mont Carlo sampling, and it is a sequential Monte Carlo methods based on particles representations of probability densities other than the Gaussian distribution which can be used in more general areas and for any state space model [Arulampalam et al. 2002; Ng et al. 2002]. The particle filter aims at tracking the state of a system as it evolves over time and typically with a non-Gaussian and potentially multi-model probability density function ($pdf$). It represents the $pdf$ as particles which are associated with weight, and estimates the states by recursively updating approximations of posterior. Figure 2 shows the basic implementation process of the generic particle filter, consisting the following three main processes:

(1) *Particle generation:* draw $\mathcal{N}$ particles with weights for state from a proposal distribution function, the proposal distribution function can be defined freely (e.g., uniform distribution).
(2) *Weight update:* the weights of particles are recursively updated and normalized.

**Algorithm 2:** Resampling Algorithm

Let CDF be the Cumulative Density Function.
Initialize CDF: $c_1 = 0$;
**for** $i = 2 : \mathcal{N}_s$ **do**
  Construct CDF: $c_i = c_{i-1} + w_t^i$;
**end**
Start at the bottom of the CDF: $i = 1$
Draw a starting point: $u_1 \sim \mathcal{U}[0, \mathcal{N}_s^{-1}]$
**for** $j = 1 : \mathcal{N}_s$ **do**
  Move along the CDF: $u_j = u_1 + \mathcal{N}_s^{-1}(j-1)$
  **while** $u_j > c_j$ **do**
    $i = i + 1$;
  **end**
  Assign sample: $x_t^{j\star} = x_t^i$ ;
  Assign weight: $w_t^j = \mathcal{N}_s^{-1}$ ;
  Assign parent: $i^j = i$ ;
**end**

(3) *Resampling:* when implementing the generic particle filter, after a few iterations, most of particles have negligible weight. In other words, the weight is only concentrated on a few particles. The resampling process stochastically discards particles with negligible weight, and replaces them with the particles with large weights.

Algorithm 1 shows the detailed algorithm of the generic particle filter. The resampling algorithm, which is also called *systematic resampling* and is simple to implement, is shown in Algorithm 2. Its time complexity is $\mathcal{O}(\mathcal{N}_s)$ where $\mathcal{U}[a, b]$ is the uniform distribution on the interval $[a, b]$. Interested readers are referred to [Kitagawa 1996] for more details.

### 2.4. Motivating Example

To illustrate the motivations of our approach, we take the widely-used *travel agency service* (Figure 3) as an example. When a user travels to a place, she usually needs to book her flight, accommodation, and perhaps needs also to rent a car. A travel agency service plays like a composition engine, which composes different Web services to fulfill the user's travel requirements. Ideally, the travel agency service should be highly available so that users can use it for their travel planning whenever they need it. Since the travel agency service relies on other Web services (e.g., flight booking service and hotel booking service), high availability of the travel agency service essentially means the selection of highly available component Web services.

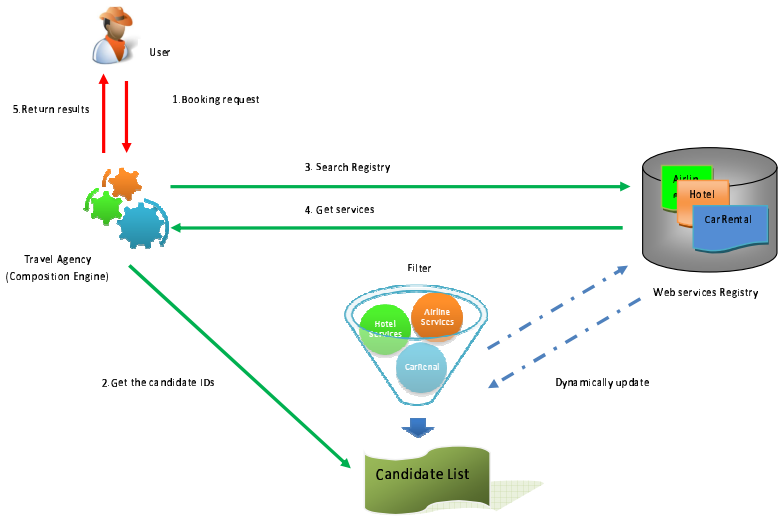


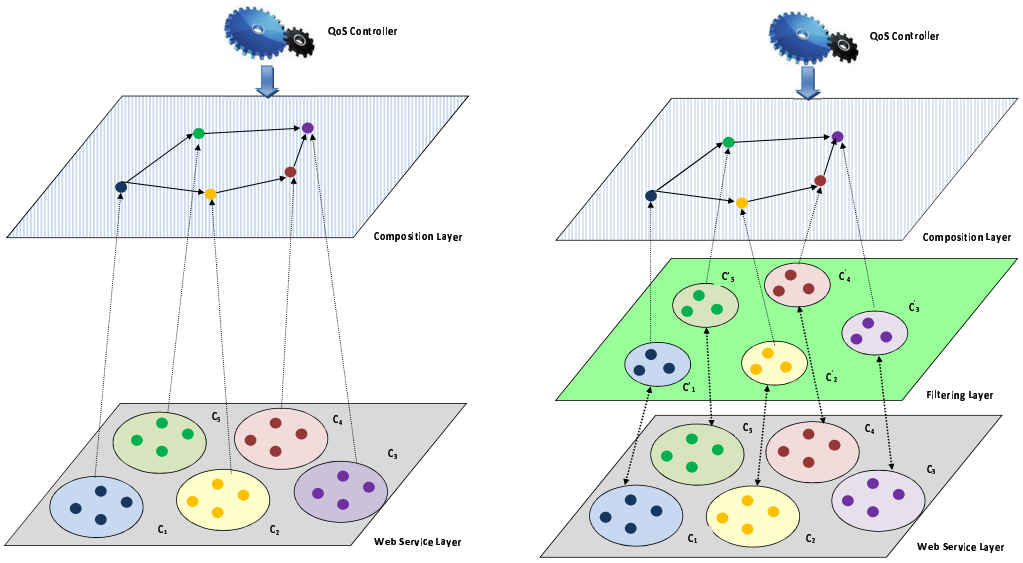

Fig. 3. A Travel Agency Scenario

However, it is quite common that there might be a large number of Web services (normally with very different QoS) providing same services (e.g., flight booking). Searching such a large service pool is not only time-consuming, but more importantly, unreliable since the availability of Web services dynamically changes over time. Our approach (Section 3) dynamically maintains the availability information of Web services and effectively filters Web services with high availability (i.e., reduce the search space), thereby improving the quality of composition Web services.

## 3. THE PARTICLE FILTER BASED APPROACH

Figure 4 shows the basic idea of our approach. Specifically, we propose to add a *filtering layer* between Web service layer and composition layer (right side of Figure 4). The layer of Web services contains several service communities and each of them consists of Web services with similar functionalities. Each community may be heavily populated with Web services.

The filtering layer is essentially a subset of service communities in the Web service layer, which consists of Web services with high availability that will directly involve in service compositions. The Web services are selected based on the accurate estimation and ranking algorithm described in this section. It should be noted that the relationship between Web service communities and the filtering layer is *dynamic* and *adaptive*. Our approach dynamically adjusts the members in the filtered service communities where degrading Web services are automatically and transparently replaced with Web services of better availability from the Web service layer.




(a) Existing approaches (b) Our proposed approach

Fig. 4. Web Service Selection in Compositions

### 3.1. Particle Filtering Based Estimation

Web services' availability state is highly dynamic and therefore needs an adaptive approach to monitor and track each Web service's state. This is important to conduct optimized selection algorithms for composite Web services such as [Zeng et al. 2003; Alrifai et al. 2010]. We apply the particle filtering technique to make accurate estimation of Web service's availability state, which serves the foundation for dynamically optimized selection of Web services in composition.

We consider that the changes of availability of Web services are uncertain. The availability modeling function is non-linear and the noise (Section 2.3) can not be guaranteed as a Gaussian distribution. Particle filter can improve the performance over the established non-linear filtering approaches since it provides optimal estimation in non-linear and non-Gaussian state space models, as well as estimation of non-linear models without making any assumption on the measurement noise distribution. Particle filter can estimate a system states sufficiently when the number of particles (estimations of the state vectors that evolve in parallel) is large.

The particle filter refers to belief using a number of particles. There are two main steps in the particle filter algorithm: *prediction* and *update*. Particle filters realize Bayes filter updates according to a sampling procedure, often called $sequential\ importance\ sampling\ with\ resampling$ [Fox et al. 2003]. Whenever new observations $z_t$ are discovered, the filter predicts the state using

**Algorithm 3:** Particle Filter based Algorithm

1. **Initialization:** compute the weight distribution $\mathcal{D}_w(a)$ according to IP address distribution.
2. **Generation:** generate the particle set and assign the particle set weight, which means $\mathcal{N}$ discrete hypothesis

— generate initial particle set $\mathcal{P}_0$ which has $\mathcal{N}$ particles, $\mathcal{P}_0$ = $(p_{0,0}, p_{0,1}, ...p_{0,\mathcal{N}-1})$ and distribute them in a uniform distribution in the initial stage. Particle $p_{0,k} = (a_{0,k}, weight_{0,k})$ where $a$ represents the Web service availability.

— assign weight to the particles according to our weight distribution $\mathcal{D}_w(a)$.

3. **Resampling:**

— Resample $\mathcal{N}$ particles from the particle set from a particle set $\mathcal{P}_t$ using weights of each particles, refer to Algorithm 2.

— generate new particle set $\mathcal{P}_{t+1}$ and assign weight according to $\mathcal{D}_w(a)$

4. **Estimation:** predict new availability of the particle set $\mathcal{P}_t$ based on availability function $f(t)$.
5. **Update:**

— recalculate the weight of $\mathcal{P}_t$ based on measurement $m_a$, $w_{t,k}$= $\prod(\mathcal{D}_w(a_{t,k}))(\frac{1}{\sqrt{2\pi\phi}})exp(-\frac{\delta a_{t,k}^2}{2\phi^2})$, where $\delta a_{t,k} = m_a - a_{t,k}$

— calculate current availability by mean value of $p_t(a_t)$

6. Go to step 3 and iteration until convergence

$Bel^- \leftarrow \int p(x_t|x_{t-1})Bel(x_{t-1})dx_{t-1}$. And then the filter will correct the predicted estimation using $Bel(x_t) \leftarrow \alpha_t p(z_t|x_t)Bel^-(x_t)$, where $Bel(x_t)$ is a probability distribution over $x_t$.

In our approach, we model the availability of a Web service $i$ at time $t$ as $x_i(t)$, which maintains the probability distribution of the service availability at $t$. The state transition of Web service $i$'s availability can be represented as:

$$x_i(t+1) = g(x_i(t)) + \phi_i(t) \tag{4}$$

where $g(x_i(t))$ denotes the nonlinear transition of service $i$'s availability and $\phi_i(t)$ denotes the noise to service $i$'s availability. We can further define the observation equation of the Web service $i$'s availability as:

$$z_i(t) = h(x_i(t)) + \delta_i(t) \tag{5}$$

where $z_i(t)$ is the observation value of service $i$'s availability, $h(x_i(t))$ is the observation function, and $\delta_i(t)$ is the observation noise. In our particle filtering approach, the posterior distribution of $x_i(t)$ can be inducted as the belief $Bel(x_i(t)) = \{x_i(t), w_i(t)\}$, $i = 1, 2, ..., \mathcal{M}$, where $w_i(t)$ are the different weight values, which indicate the contribution of the particle to the overall estimation, also called *important factors* ($\sum w_i(t)$ = 1).

Algorithm 3 shows steps to summarize the particle filtering process. Firstly, we initialize a uniformly distributed sample set representing a Web service's availability state. We assign each sample a same weight $w$. Secondly, when the availability changes, the particle filter calculates the measure-

**Algorithm 4:** Overall Adaptive Filtering Algorithm

**Input**: initial availability values, $\alpha$, $\tau$.

**Output**: predicted availability, referencing availability, candidate list.

1. Read in the initial parameters;
2. Calculate each values for Web service $a_{ij}(s,t)$ in Web service community $j$ at time $t$;
3. Predict the availability state of next time slot using particle filter (Algorithm 3);
4. Looking up database and calculate the *mean* values of availability $\mathcal{H}$.
5. Calculating the reference availability $\mathcal{R}$.
6. Update the top $k$ candidate list in each Web services community for every time interval $\tau$;
7. Go to step 2, and iterating.

ment by adjusting and normalizing each sample's weight. These samples' weights are proportional to the observation likelihood $p(z|x)$. The particle filters randomly draw samples from the current sample set whose probability can be given by the weights. Then we can apply the particle filters to estimate the possible next availability state for each new particle. The prediction and update steps will keep running until convergence.

We calculate the weight distribution by considering the bias resulted from the routing information between users and targeting component Web services (e.g., routing-hops between the user and the component Web service or whether user and targeting services are in the same IP address segment). The Sequential Importance Sampling (SIS) algorithm is a Monte Carlo method that forms the basis for particle filters. The SIS algorithm consists of recursive propagation of the weights and support points as each measurement is received sequentially. To tackle the degeneracy problem, we adopt a more advanced algorithm with resampling [Arulampalam et al. 2002]. It has less time complexity and minimizes the Monte-Carlo variation. The resampling algorithm is given in Algorithm 2.

### 3.2. The Dynamic Filtering Algorithm

Based on the algorithm mentioned above (i.e., Algorithm 3) , we can sort the top $k$ Web services with high availability according to the monitoring and prediction. We call this estimated availability $\mathcal{E}_i$. In addition, for the overall filtering algorithm, we also take the history information on availability $\mathcal{H}_i$ into account, on top of the estimated availability by using the particle filter technique. The historical fluctuation of Web services availability has important impact on the current availability of the services. We call this historical fluctuation $\mathcal{H}$ impact as *availability reputation*. The most common and effective numerical measure of the center tendency is using the $mean$, however, it is sensitive to the extreme values (e.g., outliers) [Han and Kamber 2006]. In our work, we define the final availability of a Web service as *reference availability* $\mathcal{R}$, which is calculated using:

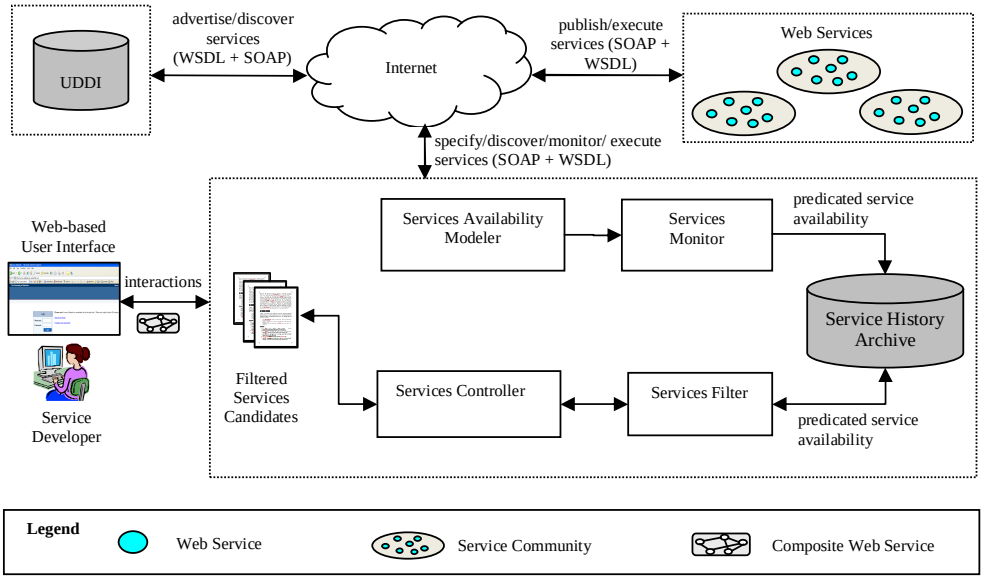



Fig. 5. Architecture of the prototype system

$$\mathcal{R}_i(\tau) = \sigma\mathcal{E}_i(\tau) + (1-\sigma)\mathcal{H}_i(\sum_{1}^{\tau-1}(\tau-1)) \tag{6}$$

where $\tau$ is a time span which can be defined by users and $\sigma \in [0,1]$ is the weight and users can assign different weight based on their different preference between predication and history of service availability. For example, if $\sigma$ is 1, the availability of a Web service will totally depend on the estimation value obtained by the particle filtering algorithm. Here, the historical values can be considered as the smoother for the reference availability $\mathcal{R}$. Finally, we summarize the overall particle filter algorithm for Web service selection in Algorithm 4.

## 4. IMPLEMENTATION AND EXPERIMENTS

In this section, we discuss the implementation of the proposed approach and also report on some experimental results.

### 4.1. System Implementation

The proposed approach has been implemented as a Web-based prototype system. Figure 5 shows its main modules. The `Service Availability Modeler` is responsible for building the Web service availability model. In particular, we have built a Particle Filter for each Web service and tagged a multiple Particle Filter according to each Web service community. The `Services`

`Monitor` is responsible for estimating and predicting the availability of Web services using the availability model. The module takes as inputs the availability values for each Web service and outputs the updated estimation of the availability for each Web service. The new estimation is also stored in a database (i.e., `Service History Archive`) that keeps the historical availability for each service.

The `Services Filter` is responsible for ranking the Web services of a community and recommend the top $\mathcal{N}$ Web services with the highest availability in real-time. In order to do so, the `Service Filter` retrieves the historical information from the archive and calculates the reference availability of Web services using Equation 6. Finally, the `Services Controller` is responsible for running the selection algorithm to generate the candidate Web services lists and update the candidate pool (e.g., replacing the degrading Web service with the one with better quality). As a consequence, Web service composition only needs to interact with this small number of candidates, which guarantee the high availability of the generated composite Web services.

The prototype system has been implemented in Java and is based on state-of-the-art technologies like XML, SOAP, WSDL, and UDDI [Curbera et al. 2002]. Java2WSDL, a tool provided by *Apache Axis*[1], is used to generate WSDL descriptions from the Java class files so that all the components of the system can be invoked as Web services. Services are deployed on Apache Axis. In our implementation, we use *Apache Tomcat*[2] as a Web server where Apache Axis is deployed.

### 4.2. Experimental Results

In this section, we present four experimental results. The first one studies the estimation accuracy of our approach. The second experiment compares the availability of composite Web services with and without our approach. The third experiment studies the impact of $\sigma$ on the error rate of estimation accuracy. The last experiment studies the performance in composing Web services using our particle filter based approach. For the experiments, we simulated 500 Web services of five different Web service communities (i.e., 100 Web services for each service community). We set the failure probability for the Web services as 3.5 percent, which complies with the findings in [Kim and Rosu 2004].

*Estimation Accuracy.* The purpose of this experiment is to study the accuracy of our availability estimation approach. In the experiment, we simulated Web services' availability fluctuation and tracked their fluctuation of availability for 50 time steps (each time step counted as an *epoch*). The actual availability of Web services and corresponding estimated availability using our particle filter

[1]http://ws.apache.org/axis/index.html.
[2]http://jakarta.apache.org/tomcat/.

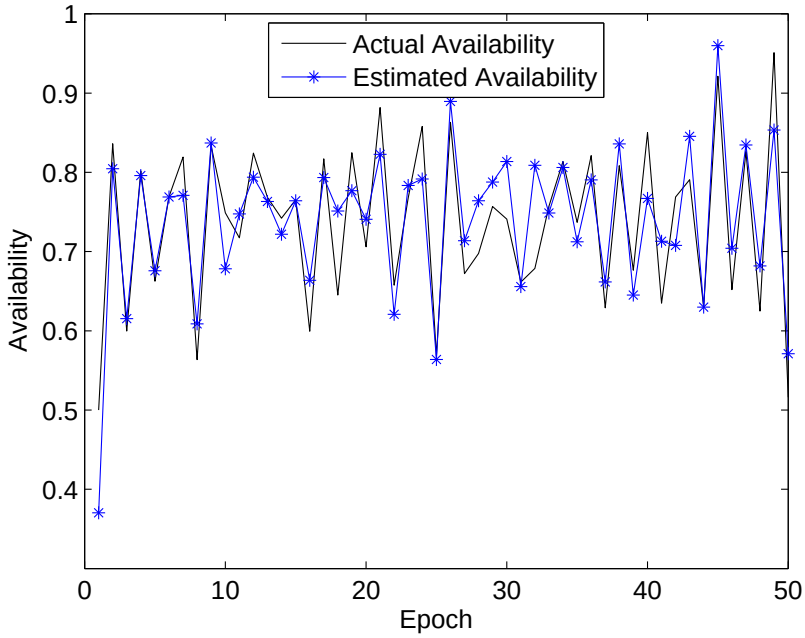



Fig. 6. Availability of a Web Service: Actual Availability vs. Estimated Availability

approach were collected and compared. Figure 6 shows the result of one particular Web service. From the figure, we can see that our approach works well in tracing and predicting the availability of Web services.

*Availability of Composite Web Services.* The purpose of the second experiment is to study the impact of our approach on the availability of composite Web services. We randomly generated composite Web services by composing services from five different communities. We simulated a comparatively significant fluctuation on the availability (i.e., changes in availability) of Web services for 50 different rounds and collected the availability information of the composite services under the situations of i) using our approach and ii) without using our approach. In our experiment, the availability of a composite Web service, $\mathcal{A}_c$, is a product of $e^{\mathcal{A}(s_i,t)}$, where $c$ is a composite Web service, $s_i$ is a component Web service of $c$, and $\mathcal{A}(s_i,t)$ is the availability of component service $s_i$.

Figure 7 shows the availability of a particular composite Web service. From the figure we can see that the availability of the composite Web service is more stable when using our approach. In contrast, without using our approach, its availability is very sensitive to the fluctuations of service availability. The reason is that our particle filter based approach can dynamically predict the availability of component Web services and proactively substitute the services with poor availability.

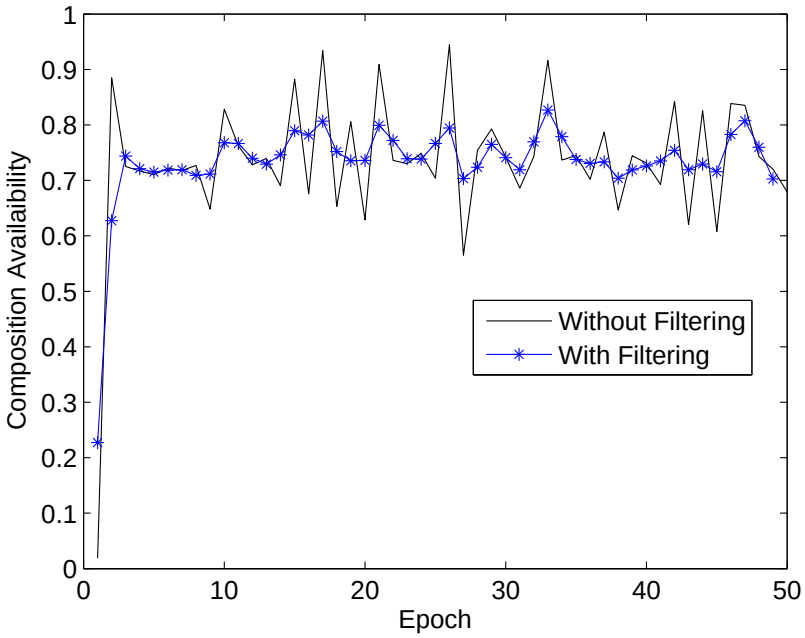



Fig. 7. Availability of a Composite Web Service

*The impact of $\sigma$ for average error rate on accuracy.* This experiment aims to study the impact of $\sigma$ (Equation 6) on the accuracy of availability estimation. In the equation, the value of $\sigma$ represents the weight between the predicted availability and historical availability. In particular, the weight of the historical availability (i.e., 1-$sigma$) is considered to be a smoother. In this experiment, we set the smoother over a range of 0 and 0.8 to show the impact on the accuracy of prediction for component Web services. Figure 8 shows the result of a particular Web service. From Figure 8 we can see that although the error rate stays relatively stable when the smoother is less than 0.2, the average availability error rate increases constantly when the smoother becomes bigger. The reason is that the role of historical data played in the particle filtering prediction process, which is based on the Markov assumption.

We also studied the impact of $\sigma$ on the availability of composite Web services. In the experiment, we set the values of the smoother as 0, 0.2, and 0.5. Figure 9 shows the result of one composite Web service. From the figure we can see that when we take the smoother into account, the availability of the composite Web service is more stable. Interestingly, there are no significant changes when the value of the smoother is set as 0.2 and 0.5. As a result, by combining our findings on estimation

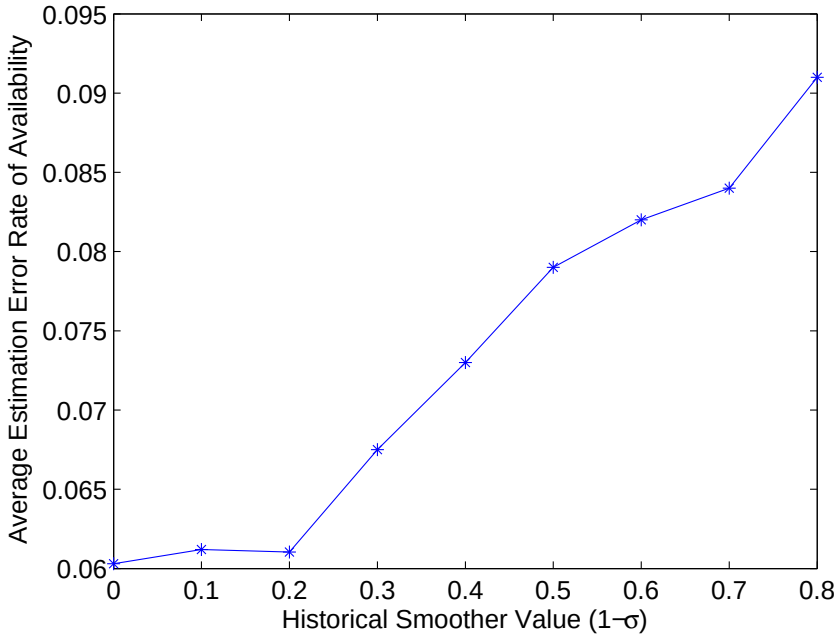



Fig. 8. The Impact of $\sigma$ on Estimated Availability Error of a Component Web Service

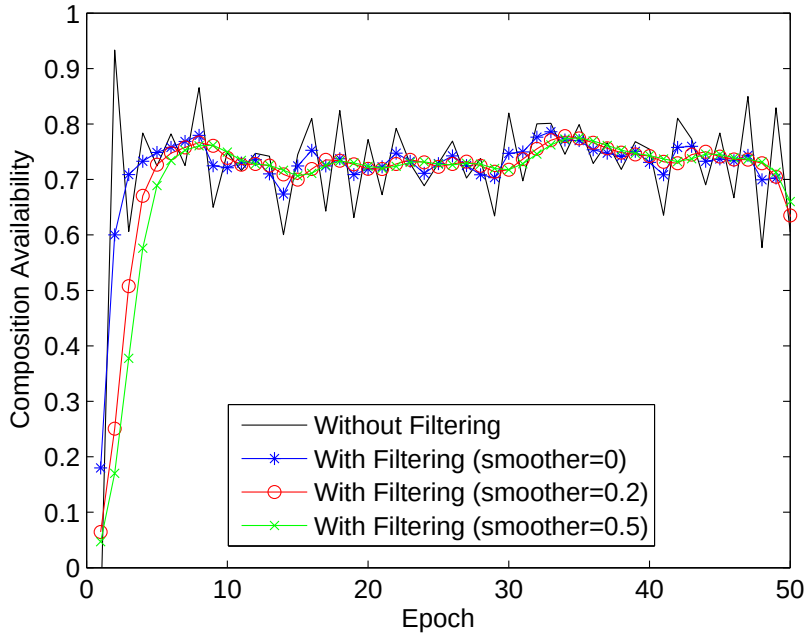



Fig. 9. The Impact of $\sigma$ on the Availability of a Composite Web Service

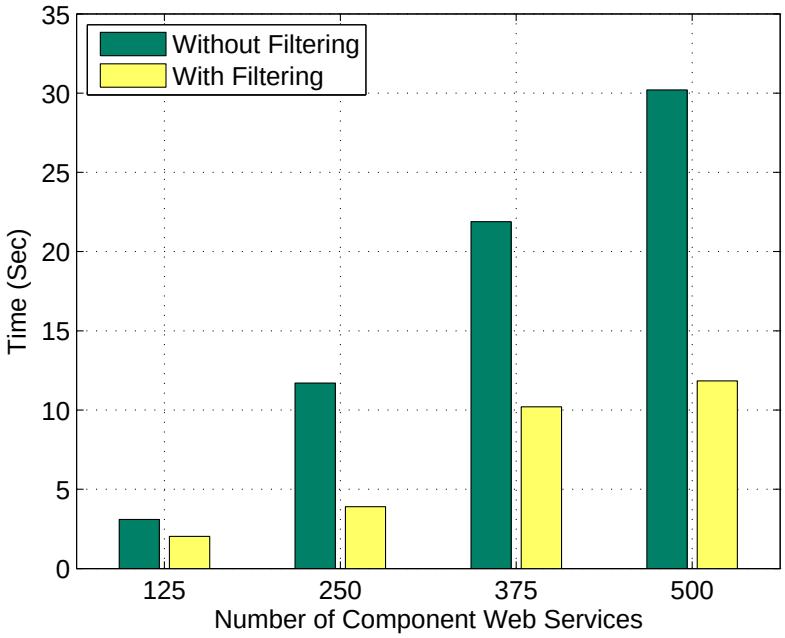



Fig. 10. Time for Composing a Composite Web Service

accuracy of component Web services (Figure 8), in our implementation, we chose to set the value of the smoother as 0.2.

*Time for composing Web services.* This experiment aims at studying the performance of our approach in Web services composition. In this experiment, we set the top-$\mathcal{N}$ Web services where $\mathcal{N}$ is 20, which means the size of candidate list for each Web service community is 20. We further set the number of component Web services to 125, 250, 375, and 500 (i.e., each service community has 25, 50, 75, and 100 Web services respectively). We recorded and compared the time used for composing composite Web services with and without our proposed filtering algorithm. The availability of the composite Web services is manually set in this experiment ($\geqslant 0.80$). It should be noted that in real situation, the requirement of a composite Web service's availability is usually determined by the SLA. All composite Web services produced similar results and Figure 10 shows the result of a certain composite Web service. It can be noticed that the improvement in reducing the execution time is obvious, particularly when the size of service communities becomes bigger. This is due to the smaller size of the filtered service communities with high quality component Web services.

## 5. RELATED WORK

There is a large body of research work related to the topic we discussed in this paper. One important area on achieving high availability of Web services focuses on replication technology [Salas et al. 2006; Serrano et al. 2008; Sheng et al. 2009]. The underlying idea is to spread service replicas over various locations and if needed, direct invocation requests to appropriate replica (e.g., with lower workload). Serrano et al. [Serrano et al. 2008] discuss an autonomic replication approach focusing on performance and consistency of Web services. Salas et al. [Salas et al. 2006] propose a replication framework for highly available Web services. Sheng et al. [Sheng et al. 2009] further developed the idea by proposing an on-demand replication decision model that offers the solution to decide how many replicas should be created, and when and where they should be deployed in the dynamic Internet environment. While these approaches focus on improving service availability through replication, our work concentrates on monitoring and predicting service availability. Our work is complementary to these works in the sense that the estimations provide a good source of information for replication decisions.

Many research projects achieve high availability of Web services based on the concept of *service communities* where Web services are selected based on QoS [Liu et al. 2004; Zeng et al. 2004; Wang et al. 2006; Maamar et al. 2008; Alrifai et al. 2010; Medjahed and Bouguettaya 2011]. The basic idea is that services with similar functionalities are gathered as communities. If a Web service is unavailable, another service will be selected based on QoS. The work presented in [Zeng et al. 2004] is the first of few that focuses on optimizing services selection during composition. The authors advocate that service selection should be carried out during the execution of a composite service, rather than at design time. They further propose a global planning approach to optimally select services based on linear programming methods. In one of the most recent works, Alrifai et al. propose a new approach based on the notion of skyline to select services for composition effectively and efficiently [Alrifai et al. 2010]. However, most approaches assume that QoS is readily accessible and ignore its dynamic nature. In addition, selecting Web services from large communities may have performance issues. Our work focuses on these issues by proposing a particle filter based approach (could say more later).

The work presented in [Guo et al. 2008; Sirin et al. 2004; Rosario et al. 2008; Hwang et al. 2007] are the most similar ones to our work. In [Guo et al. 2008], Guo et al. model a composition process into the Markov Decision Process and use Kalman Filter to tracking the state of composite Web services. Sirin et al. [Sirin et al. 2004] propose a filtering methodology that exploit matchmaking algorithms to help users filter and select services based on semantic Web services in composition process. Rosario et al. [Rosario et al. 2008] focus on Service Level Agreements (SLAs) of composite

Web services and propose a soft probabilistic contracts that consist of a probability distribution for the considered QoS parameter. These soft contracts can be composed to yield a global probabilistic contract for composite Web services. However, these works focus on adaptive composition of Web services and do not pay attention to the availability of component Web services. Finally, Hwang et al. [Hwang et al. 2007] propose a probability-based QoS model for describing QoS values of both atomic and composite Web services. Our approach uses particle filter to precisely predict the availability of Web services in real time and dynamically maintains a subset of Web services with higher availability, from which service developers can choose in their compositions.

## 6. CONCLUSION

Guaranteeing the availability of Web services is a significant challenge due to the varying number of invocation requests the Web services have to handle at a time, as well as the dynamic nature of the Web. Many existing approaches ignore the uncertain nature of service availability and simply assume that the availability information of a Web service is readily accessed. In this paper, we proposed a novel approach to monitor and predict Web service's availability based on particle filter techniques. Furthermore, we developed algorithms to filter Web services from service communities for efficient service selection. The implementation and experimental results validated our approach.

There are a few directions following our work presented in this paper. First of all, we will conduct more experiments to study the performance of the proposed approach (e.g., scalability). We also plan to extend our approach to support other important service dependability properties such as reputation, reliability, and security, which eventually underpins the construction of robust and highly dependable Web services.

## ACKNOWLEDGMENTS

Quan Z. Sheng's work has been partially supported by Australian Research Council (ARC) Discovery Grant DP0878367.

## REFERENCES

ALRIFAI, M., SKOUTAS, D., AND RISSE, T. 2010. Selecting Skyline Services for QoS-based Web Service Composition. In *Proceedings of the 19th International World Wide Web Conference (WWW'2010)*. Raleigh, North Carolina, USA.

ARULAMPALAM, M., MASKELL, S., GORDON, N., AND CLAPP, T. 2002. A Tutorial on Particle Filters for Online Nonlinear/non-Gaussian Bayesian Tracking. *IEEE Transactions on Signal Processing 50,* 2, 174–188.

BENATALLAH, B., SHENG, Q. Z., AND DUMAS, M. 2003. The Self-Serv Environment for Web Services Composition. *IEEE Internet Computing 7,* 1.

CURBERA, F., DUFTLER, M., KHALAF, R., NAGY, W., MUKHI, N., AND WEERAWARANA, S. 2002. Unraveling the Web Services Web: An Introduction to SOAP, WSDL, and UDDI. *IEEE Internet Computing 6,* 2, 86–93.

DOMINGUE, J. AND FENSEL, D. Toward A Service Web: Integrating the Semantic Web and Service Orientation. Service Web 3.0 Project, http://www.serviceweb30.eu.

ELSAYED, A. 1996. *Reliability Engineering*. Addison Wesley.

FOX, D., HIGHTOWER, J., LIAO, L., SCHULZ, D., AND BORRIELLO, G. 2003. Bayesian Filtering for Location Estimation. *IEEE Pervasive Computing 2,* 3, 24–33.

GUO, H., HUAI, J., LI, Y., AND DENG, T. 2008. KAF: Kalman Filter based Adaptive Maintenance for Dependability of Composite Services. In *Proceedings of the 20th International Conference on Advanced Information Systems Engineering (CAiSE'08)*. Montpellier, France, 328–342.

HAN, J. AND KAMBER, M. 2006. *Data Mining: Concepts and Techniques*. Morgan Kaufmann.

HWANG, S.-Y., WANG, H., TANG, J., AND SRIVASTAVA, J. 2007. A Probabilistic Approach to Modeling and Estimating the QoS of Web Services based Workflows. *Information Sciences 177*, 5484–5503.

KIM, S. AND ROSU, M. 2004. A Survey of Public Web Services. In *Proceedings of the 13th International World Wide Web Conference (WWW'04)*. New York, NY, USA.

KITAGAWA, G. 1996. Monte Carlo Filter and Smoother for Non-Gaussian Nonlinear State Space Models. *Journal of Computational and Graphical Statistics 5,* 1, 1–25.

LIU, Y., NGU, A., AND ZENG, L. 2004. QoS Computation and Policing in Dynamic Web Service Selection. In *Proceedings of the 13th International World Wide Web Conference (WWW'04)*. New York, NY, USA.

MAAMAR, Z., SHENG, Q. Z., AND BENSLIMANE, D. 2008. Sustaining Web Services High Availability Using Communities. In *Proceedings of the 3rd International Conference on Availability, Reliability, and Security (ARES'08)*. Barcelona, Spain.

MEDJAHED, B. AND BOUGUETTAYA, A. 2011. *Service Composition for the Semantic Web*. Springer.

NG, B., PESHKIN, L., AND PFEFFER, A. 2002. Factored Particles for Scalable Monitoring. In *Proc. of the 18th International Conference on Uncertainty in Artificial Intelligence (UAI'02)*. Edmonton, Alberta, Canada.

PAPAZOGLOU, M. P., TRAVERSO, P., DUSTDAR, S., AND LEYMANN, F. 2007. Service-Oriented Computing: State of the Art and Research Challenges. *IEEE Computer 40,* 11, 38–45.

RAN, S. 2003. A Model for Web Services Discovery with QoS. *ACM Sigecom exchanges 4,* 1, 1–10.

ROSARIO, S., BENVENISTE, A., HAAR, S., AND JARD, C. 2008. Probabilistic QoS and Soft Contracts for Transaction-Based Web Services Orchestrations. *IEEE Transactions on Services Computing 1,* 4, 187–200.

SALAS, J., PÉREZ-SORROSAL, F., PATIÑO-MARTÍNEZ, M., AND JIMÉNEZ-PERIS, R. 2006. WS-Replication: A Framework for Highly Available Web Services. In *Proceedings of the 15th International Conference on World Wide Web (WWW'06)*. Edinburgh, Scotland.

SERRANO, D., PATIÑO-MARTÌNEZ, M., JIMENEZ-PERIS, R., AND KEMME, B. 2008. An Autonomic Approach for Replication of Internet-based Services. In *Proceedings of the 27th IEEE International Symposium on Reliable Distributed Systems (SRDS'08)*. Napoli, Italy.

SHENG, Q. Z., MAAMAR, Z., YAHYAOUI, H., BENTAHAR, J., AND BOUKADI, K. 2010. Separating Operational and Control Behaviors: A New Approach to Web Services Modeling. *IEEE Internet Computing 14,* 3, 68–76.

SHENG, Q. Z., MAAMAR, Z., YU, J., AND NGU, A. H. 2009. Robust Web Services Provisioning Through On-Demand Replication. In *Proceedings of the 8th International Conference on Information Systems Technology and Its Applications (ISTA'09)*. Sydney, Australia.

SIRIN, E., PARSIA, B., AND HENDLER, J. 2004. Filtering and Selecting Semantic Web Services with Interactive Composition Techniques. *IEEE Intelligent Systems 19,* 4, 42–49.

WANG, X., VITVAR, T., KERRIGAN, M., AND TOMA, I. 2006. A QoS-aware Selection Model for Semantic Web Services. In *Proceedings of the International Conference on Service-Oriented Computing (ICSOC'06)*. Chicago, USA.

YAO, L. AND SHENG, Q. Z. 2011. Particle Filtering based Availability Prediction for Web Services. In *Proceedings of the 9th International Conference on Service-Oriented Computing (ICSOC 2011)*. Paphos, Cyprus.

YU, Q., BOUGUETTAYA, A., AND MEDJAHED, B. 2008. Deploying and Managing Web Services: Issues, Solutions, and Directions. *The VLDB Journal 17,* 3, 537–572.

ZENG, L., BENATALLAH, B., DUMAS, M., KALAGNANAM, J., AND SHENG, Q. Z. 2003. Quality Driven Web Services Composition. In *Proceedings of The 12th International World Wide Web Conference (WWW'2003)*. Budapest, Hungary.

ZENG, L., BENATALLAH, B., NGU, A., DUMAS, M., KALAGNANAM, J., AND CHANG, H. 2004. QoS-aware Middleware for Web Services Composition. *IEEE Transactions on Software Engineering 30,* 5, 311–327.