User-Perceived Service Availability: A Metric and an Estimation Approach

Lingshuang Shao[1,2] Junfeng Zhao[1,2,*] Tao Xie[3] Lu Zhang[1,2] Bing Xie [1,2] Hong Mei[1,2]
[1] School of Electronics Engineering and Computer Science, Peking University, China
[2] Key Laboratory of High Confidence Software Technologies, Ministry of Education, Beijing 100871, China
[3] Department of Computer Science, North Carolina State University, USA
{shaolsh04, zhaojf}@sei.pku.edu.cn, xie@csc.ncsu.edu, {zhanglu, xiebing, meih}@sei.pku.edu.cn

Abstract

Web-service-related techniques have become popular to improve system integration and interaction. In distributed and dynamic environment, web services' availability has been regarded as one of the key properties for (critical) service-oriented applications. Quality of Service (QoS), including availability, has been regarded by IEEE as a user-perceived property. However, based on our investigation of monitoring invocation records of real web services, existing availability metrics, which were proposed in traditional domains, have not addressed the "user-perceived" characteristics. Based on analyzing the limitations of the existing availability metrics, we propose a status-based user-perceived service availability metric and a corresponding estimation approach. Experiments on monitoring and analyzing the invocation records of real services demonstrate that the new metric and the corresponding estimation approach could lead to a feasible estimation on web services' availability from the user side.

1. Introduction

As an emerging technology to improve system integration and interaction, Web-service-related techniques have become a popular topic in the research community. Service-oriented methodologies have been regarded as promising solutions for future distributed applications [1, 18, 8]. Meanwhile, because of the distributed and dynamic nature of web services, many researchers propose that quality of services (QoS) should be addressed for successfully building critical service oriented applications [14]. Availability, one of the key properties of web services QoS, has been explored intensively by various researchers [4, 6, 10].

* Corresponding author

QoS, including availability, has been regarded as user-perceived properties [5]. From our practical experiences on running and monitoring web services, we find that because of different network factors and user behavior, web service availability measured at the server side and the user side are often different. Therefore, the measurement at the server side is not sufficient for service users, although server-side measurement has been intensively addressed.

However, existing availability metrics [3, 10, 19, 20] have not addressed the characteristics of "user-perceived". Traditionally, availability can be presented as the "up" or "down" of the system at a single point of time. As an extension of this definition, availability can also be presented as the average percentage of time [3]. In traditional domains, such as hard disk availability measurement, the "time percentage" metric is better than "single point of time" metric. However, for user-perceived metrics of web services, because the round trip time of invoking services is very long (comparing to accessing hard disks) and using some services even needs to be paid, the frequency of invoking web services cannot be very high. Then the "time percentage" metric, which implies that accessing is frequent enough that can be regarded as continuous.

We propose that service availability metrics should be estimated based on the success rate of service invocations (according to the "single point of time" metric). However, this "single point of time" metric has its own limitation. As proposed by Brown and Patterson [3], from the perspective of service users, there is much difference between encountering 10 continuous failures and discrete failures during 1000 one thousand invocations.

In our previous work [16], we conducted an experiment for monitoring services on the Internet and collected about 50,000 invocation records. From statistical analysis of these records, we observe several implicit patterns of service-invocation successes and failures. These characteristics depict that simply averaging historical results is not appropriate. In addition, these characteristics provide potential capability for helping estimate availability. However, previous metrics and estimation approaches for availability have not accommodated these characteristics, facing limitations when being used.

To accommodate the characteristics of failures and successes of service invocations, we propose a novel availability metric and a corresponding availability estimation approach in this paper. The basic idea of this metric is that the service is running in different statuses and each status has its own features of encountering failures. We setup a three-status model and specify corresponding features for each of the status. With the status model and the features, we propose a novel estimation approach for service availability perceived by users.

We conduct a two-fold evaluation for this status-based metric and the corresponding estimation approach. First, to evaluate the usefulness of this metric, we analyze the features of interactions between services and service users. Then we conduct a study to show that this new metric can reveal more useful information than traditional metrics. Second, to evaluate the estimation approach, we keep collecting invocation records of various services, such as Google web page search and Amazon S3. From this study, we observe that the estimation approach can lead to a reasonable result on measuring service availability.

This paper makes the following three main contributions: (1) identifying the limitations of traditional metrics of user-perceived service availability and proposing a status-based metric, (2) proposing an estimation approach for our metric, and (3) providing empirical studies to evaluate the metric and the approach.

The rest of the paper is organized as follows. Section 2 describes the background of collecting service invocation records and characteristics of service failures and successes. Section 3 presents the limitations of existing metrics and proposes a status-based metric for web service availability. Section 4 describes the estimation approach. Section 5 presents the evaluation of the metric and the estimation approaches. Section 6 discusses the related work. Section 7 concludes this paper and outlines our future work.

2. Background: Characteristics of Service-Invocation Failures and Successes

In our previous work [16], we have collected and analyzed invocation records of five services. We have found out some possible patterns of service invocation failures and successes. Because these identified patterns are important basis for our new metric and estimation approach, we next briefly present these patterns as a background.

The invocation records are collected from monitoring five real web services. These services are Google Web Page Search, Yahoo!, Amazon S3, and SearchWS and LetterSOAP published at www.xmethods.net. The client programs are developed with Java JDK 1.5 and Apache Axis 1.4. Some parameters should be set before the client programs are launched. In our experiment, there are two types of parameters: (1) invocation parameters and (2) invocation configuration parameters. These parameters are set according to the rules described in our previous work [15].

2.1. Three Typical Patterns of Service-Invocation Successes and Failures

In our previous work, we finally collect about 10,000 invocation records for each of the monitored service. We intensively go through these records and analyze the patterns of service-invocation successes and failures [16]. These patterns occur repeatedly for almost all the services and we believe that they are representative. There are three patterns: continuous successes, transient failures and continuous failures.

Continuous Successes. In the monitored services, encountering continuous successes is the most prevalent pattern. Except for one service, SearchWS service, which has never been accessed successfully, all the other services have presented the characteristic of being stable up.

Transient Failures. When analyzing the services' invocation records, we find that some services may encounter unstable failures. This pattern can be further divided into two categories: first, a client program may encounter transient invocation failures but successfully access the service for next several invocations; second, transient invocation successes and failures happen alternatively.

This pattern has been discovered in several services. We classify cases with less than continuous three failed invocations as transient failures.

Continuous Failures. Some services have encountered continuous failures during our study. Because the invocation records are collected at the client side, this pattern presents the failures of either the service or network. In user-perceived metric, failures of the service or network are not explicitly distinguished, because user-perceived metric aims to estimate "user-perceived availability", which means that the measurement should be based on the user-side data and should consider both service running and network factors.

These patterns show the runtime characteristics of service invocation successes and failures. A remarkable advantage of discovering these patterns is that it can efficiently help improve the accuracy of estimating a service's availability. Existing metrics have not accommodated these characteristics, which hinder them in being sensitive to the real situation of service-availability.

3. Status-Based Service Availability Metric

The patterns of service failures and successes described in Section 2 show that services are running in different statuses. In this paper, we propose a three-status model, which assumes that services are running in three statuses: Stable Up, Transient Down, and Short-term Down. We next describe the definition of these statuses and explain why we adopt this model.

- Stable Up (SU): This status means that the service is running stably and no invocation failure happens. From the invocation records collected by us, services provided by reputable software enterprises are running in this status at most occasions.
- Transient Down (TD): This status indicates that encountering failures during invocation is not predictable and it may be recovered by a simple "retry". As shown in Section 2.1, most of the monitored services have been involved in this status.
- Short-term Down (STD): This status means that a service becomes down due to specific factors. From the perspective of service users, they encounter continuous failures. This status is common for web services. For example, periodical server restarting can cause the service to be unavailable for several minutes. Because user-perceived availability also takes into consideration of the network status, it is even more likely for service users to encounter short-term network failures.

This three-status model has two advantages. First, this model is simple. We believe that a service's status is dividable. For example, short-term down can be further divided into many statuses, such as short-term down and long-term down. However, adding more statuses calls for more rules to differentiate them, making the model and the corresponding estimation algorithm complex. Second, this status model is practical. In the reliability engineering community [9, 17], the service statuses are always described as stable up, transient down, and persistent (permanent) down. Researchers contributed much work on analyzing availability with this service status model, which is shown to be understandable and useful.

Based on these three statuses, we define a new service-availability metric as a union of two concepts: the time percentage of being in each one of these statuses and the success rate in each one of these statuses. For example, the service availability that is measured with our metrics can be represented as in Table 4.

	SU	TD	STD
Time Percentage	0.3	0.6	0.1
Success Rate	1	0.95	0

Table 1. An Example of Status-Based Service Availability Metric

4. Status-Based Availability Estimation Approach

Using the status-based metric can address the problem of differentiating "discrete failures" from "continuous failures". However, this metric is more complex than traditional metrics and calls for a practical estimation approach for computing service availability. To address this issue, we propose an estimation approach for the metric. This estimation approach tries to give a long-term measurement of service availability. The approach takes historical invocation records as input and produces the value of the metric.

We next first define the related concepts formally and then describe the details of the estimation approach.

4.1. Definitions

The processes of availability estimation can be regarded as handling a sequence of web service invocation records and producing the estimated value(s). The formal definitions of related concepts are listed below:

1. $I_t \in \{0, 1\}, t \in N$.

I_t is binary denoting the result of a service invocation at time point t (t can be presented as an integer). The value of I_t can be 0 or 1; 1 indicates that the invocation is successful and 0 indicates the contrary.

2. $R_{1,n} = \{I_{t_1}, I_{t_2}, I_{t_3}, ..., I_{t_n}\}, \forall i, \forall j, i \in N, j \in N, t_i \in N, i < j \leftrightarrow t_i < t_j$.

$R_{1,n}$ is a sequence of I_t, sorted by ascending time. $R_{1,n}$ is a sequence of invocation results, which record the invocation results during a specific time period. $|R_{i,j}|$ denotes the "length", number of invocation results, of the sequence.

3. $R^v = \{R_{k_1,k'_1}, R_{k_2,k'_2}, ..., R_{k_t,k'_t}\}, \forall i, j, 1 \leq i \leq t \wedge 1 \leq j \leq t \rightarrow R_{k_i,k'_i} \cap R_{k_j,k'_j} = \emptyset$

R^v is a set that contains a set of R_{k_i,k_j}, where every two elements have no intersection. R_{k_i,k_j} is a "subsequence" of R^v if $R_{k_i,k_j} \in R^v$.

4. $S = \{s_1, s_2, s_3, s_4\}$ is a set of service running statuses, in which $s_i (i \in N, 1 \leq i \leq 4)$ is an independent status. In our context, $s_i (i \in N, 1 \leq i \leq 4)$ respectively denotes the "Stable Up", "Transient Down", "Short-Term Down" and "Long-term Down" statuses.

5. $S_{R_{i,i'}} \in S$ is the service running status during time period t_i and t'_i with the invocation record sequence $R_{i,i'}$.

4.2. Estimation Approach

The input of the estimation approach is a sequence of invocation records. The goal of the estimation approach is to calculate the service availability values under different statuses. So the problem of estimation can be divided to two sub-problems: how to identify service running status-based on invocation records and how to calculate the availability value under each different status. To address these problems, the estimation approach consists of three steps: (1) dividing the historical invocation records into fragments, each of which is regarded as a status, (2) identifying features of each status, and (3) estimating the values of availability metrics under each status.

Invocation-Record Division. Given all the historical invocation records $R_{1,n}$, we first divide them into k small fragments. The parameter k is set manually according to n: if n is large (such as 10,000), k is set large (such as 100); otherwise, k is set small. Each fragment contains an approximately equal number of records: Meanwhile, we take into consideration of the time internal between records, records with comparatively shorter time interval are divided into the same fragment. Below is the algorithm of division:

Input: $R_{1,n}$ and k

Output: R^v

1. $number \leftarrow \llcorner n/k \lrcorner$

2. Scanning $R_{1,n}$ orderly, put every $[n/k]$ records into a different sequence $R_{i,j}$. The remaining records are put into the last sequence. Then $R_{1,n}$ is divided into k fragments.

3. Repeat the following procedure until R^v remains unchanged. Scanning all sequences in R^v orderly and comparing the records at the boundary of each sequence. Suppose that there are two neighboring sequences $R_{i,j}$ and $R_{i',j'}$ and the adjacent records are I_{t_j} and $I_{t'_i}$, if ($|t_j - t'_i| < |t'_i - t_{i'+1}|$), then move I'_i into $R_{i,j}$, or else, move I_i into $R_{i',j'}$.

4. Scanning all sequences in R^v orderly. For $R_{i,j} \in R^v$, if $\exists R_{i',j'} \in R_{i,j} \wedge \forall I_k, I_k \in R_{i',j'} \rightarrow I_k = 0 \wedge |R_{i',j'}| > t$, then break sequence $R_{i,j}$ into three sub-sequences: $R_{i,i'-1}$, $R_{i,i'-1}$, $R_{i,i'-1}$. In this step, t is a manually defined threshold value to identify whether the service status is short-term down: when the length of continuous failures is more than t, then the service status can be regarded as short-term down.

In these steps, Step 3 divides the invocation records into sequences. Step 4 separates continuous failures sequences from the original sequence.

Status Identification. One of the key tasks in our estimation approach is to identify the service running status according sequence $R_{k,j}$. This status identification is realized as applying a sequence of rules, according to our basic definition for the three statuses:

1. If there is no invocation failure record in $R_{k,j}$, then this sequence is marked as status "Stable Up",

2. If there is no successful invocation record in $R_{k,j}$, then this sequence is marked as status "Short-term down",

3. Others are marked as "Transient down".

4.2.1. Availability Estimation After the historical records have been divided into sequences and marked with running status, the next task is to estimate the availability value. Our estimation consists of three steps: (1) calculating the success percentage for each sequence, (2) calculating the weighted average of success rates for status "Transient Down", and (3) calculating the time percentage for each status.

Calculating Success Rate. Because each status has different characteristics, this step defines different estimation strategies for each status.

Stable Up: According to the definition of status "SU", the estimated value of availability is "1".

Short-term Down: Similarly, according to the definition of status "STD", the estimated value of availability is "0".

Transient Down: The service may have been in the "transient down" status for several times. For every time when the service runs in this status, there are several invocation records being recorded and the availability value can be calculated as Equation (1) [10], where N_a denotes the number of total invocations and N_f denotes the number of encountered failures.

$$Ava_{td} = (N_a - N_f)/N_a \tag{1}$$

Weighted Averaging the Success Rate for Status "Transient Down"

For the divided sequences R^v, there are many sequences marked with status "Transient Down", and each sequence has a different success rate. To produce an overall evaluation for the success rate in status "Transient Down", we propose a "weighted average" technique to calculate the overall success rate, and the weighting value is the time elapsed for each sequence. For each sequence $R_{k,j}$, the time elapsed in this sequence is calculated by $|t_j - t_i|$. Suppose that the sequence is $R^v = \{R_{k_1,k'_1}, R_{k_2,k'_2}, ..., R_{k_t,k'_t}\}$, the calculation for overall success rate is based on Equation (2):

$$Ava^o_{td} = \frac{\sum_{i=1..t \wedge S_{R_{i,i'}}=TD} |t_{k'_i} - t_{k_i}| * Ava^{i,i'}_{td}}{\sum_{i=1..t \wedge S_{R_{i,i'}}=TD} |t_{k'_i} - t_{k_i}|} \tag{2}$$

Time Percentage of Each Status. The calculation of the time percentage of each status is based on Equation (3). In this equation, the denominator is the total time duration of the invocation and the numerator is the time duration of one of the three statuses where s_i denotes a status, such as "SU", "TD" or "STD".

$$\frac{\sum_{i=1..t \wedge S_{R_{i,i'}}=s_i} |t_{k'_i} - t_{k_i}|}{\sum_{i=1..t} |t_{k'_i} - t_{k_i}|} \tag{3}$$

5. Evaluation

To evaluate the new metric and the estimation approach, we design empirical studies to collect web services invocation records and evaluate whether our approach can provide a comparatively accurate estimation on web services availability. In our studies, we collect totally about 200,000 records on invoking 20 different web services deployed on the Internet. We compare our estimation approach with traditional time-period-based estimation approach. The results demonstrate the feasibility and benefits of our approach.

The evaluation includes two aspects: (1) evaluating whether the status-based metric proposed in this paper is useful for service users, and (2) evaluating the usefulness of our estimation approach with a case study.

In this section, we first briefly describe the service running records collected by us and then describe the evaluations.

5.1. Data Sets

To evaluate our approach, we apply our estimation approach on several real web services. The setup for collecting invocation records has been described in Section 2. We run these client programs for about three month and each client program collects about 20,000 invocation records. Each invocation record contains the begin and end time of the invocation and the request and response messages. Services are regarded as unavailable, when (1) they fail to deliver the response message in pre-set time interval, or (2) the response message shows that the connection error occurs during invocation.

5.2. Evaluation of the Status-Based Metric

As stated in Section 2 and 3, the new metric proposed has conveyed much more information about the service running than traditional availability metrics. Comparing to the traditional metrics at a single point of time, the new metric can reveal the continuity of encountering failures by explicitly dividing service running statuses into three categories. In this section, we design a study to show the usefulness of the new metric. This study shows the limitations of traditional metrics at different application scenarios and how the new metric can address these limitations.

From the experiences of using web services, there are two major factors that may impact how service users choose services. The application scenario (the context of using the services) and the interaction pattern between service users and services. These two factors are not always independent. For example, the application scenario may to some extent impact the interaction pattern. However, in most occasions, these two factors are independent and we next analyze them.

In different application scenarios, service users may emphasize different aspects of service availability. For example, in a safety-critical environment, service failures cost much and the service users may care most about whether the service can be stably running. Specifically, they care primarily about the time percentage of stable up and whether transient down or short-term down makes no difference for them. However, in another application scenario, the way to choose service may be totally different. For example, in a non-critical application, such as checking some sports news, service failures may impact user experiences but not cost much. Service users may not care much about whether the service is always running stable, if transient failures do not frequently happen, they are also acceptable. Compared to the critical-application scenario, the time percentage of being in Stable Up and Transient Down should both be taken into consideration.

The other factor impacting the way of choosing service is the interaction pattern between service users and services. From the perspective of interaction frequency, the interaction pattern can be classified into two categories: continuous and discrete interactions.

The continuous interaction indicates that the service users access services frequently. If the service users has successfully access the service once, it is likely that these service users will access this service again short time later. Sometimes, those two nearby invocations are not independent. The discrete interaction indicates that the time interval between two invocations from one service users is very uncertain, likely being very long.

(a) Service 1

	SU	TD	STD
Time Percentage	0.8	0.14	0.06
Success Rate	1	0.89	0
Total Success Rate	0.871		

(b) Service 2

	SU	TD	STD
Time Percentage	0.85	0.01	0.14
Success Rate	1	0.91	0
Total Success Rate	0.866		

Table 2. Service Availaiblity

Our next example shows how application scenarios impact the choices of services and why our new metric can help convey more useful information. Table 2 shows two real web services'^{1,2} availability measured by our proposed metric and the total success rates. The functionality of both these two services are in-time currency exchange rate querying. The total success rate of these two services are so close that according to traditional metrics, these two services are of the same availability. However, with the status-based metric, service users can differentiate these two services according to application scenarios and interaction patterns.

First, let us assume that there is a financial company preparing for integrating a currency exchange service. Because this functionality strongly impacts this company's core business, engineers suggest that they should choose a service that is the most stable because the failures, either transient or short-term, make big impact on the running of their core business. At the occasion, the financial company would prefer to choose a service with more time percentage in the stable up status. In this case, Service 2 is preferred.

Second, let us assume that a service users wants to find some currency exchange rate querying service for checking the currency exchange rate several times per day, indicating that the interaction pattern is discrete. The total success rates of these two services are not much different. However, with status-based metrics, this service users may find that though Service 1 has smaller time percentage in Stable Up, this service has much more time percentage in Stable Up and Transient Down statues than Service 2.

Service 1 is impacted much by the transient instability but Service 2 is more likely to stop its service for a comparatively long time (from the table, we can observe that Service 1 has more time percentage in stable up and transient down, and comparatively less time in short-term down). At this occasion, Service 1 may be preferred because the service users may recover from the failure by a simple retry and a transient failure has not much impact on the user experience. But short-term down of the service may handicap the service users on getting the service.

In summary, in many occasions, a simple binary metric at single point time is not sufficient for helping service users to choose the best services. Our status-based metric is more applicable for conveying much more useful information.

5.3. Study for Estimation Approach

In this section, we present a real case to evaluate the feasibility of the estimation approach: whether the estimation approach can address the real situation of service running.

On December 26th, 2006, the submarine cable between China and America was interrupted because of an undersea earthquake. The repair work lasted three weeks and during that period, Chinese people felt it was very hard to visit web sites hosted in America. Meanwhile, most of our monitored web services become unavailable for us in that period.

The QoS data collected by us covers part of this special time: we had collected the QoS information from November 10th, 2006 to January 25th, 2007. This time period covers some days before the submarine cable was broken and early days after the submarine cable had been completely recovered. We next show the availability metric of the well-known Google Web page Search Service at that time period in Table 3 as an example. According to the news report of the cable-repair process, we separate the data for four phases: first, the time before the earthquake happened (before Dec. 25th, 2006, marked as Phase 1); second, the early days of which the earthquake happened (from Dec. 26th, 2006 to Jan. 9th, 2007, marked as Phase 2); third, the last days of the cable repair work (from Jan. 10th to Jan. 28th, 2007, marked as Phase 3); fourth, the time after the submarine cable had been completely recovered (after Jan. 28th, 2007, marked as Phase 4).

In Table 3, "SU" denotes the Stable Up status, "TD" denotes the Transient Down status, and "STD" denotes the Short-term Down status.

In Phase 1, Google web page search was shown to have very good capability of providing the service.

In Phase 2, when the earthquake happened, the time percentage of transient down and short-term down of Google service increased. We review related news reports at those days and find out that Google had paid some Information Service Provider for transferring its data through another route. Because the new route could not provide as much as band width as the original submarine cable, it causes much short-term down and transient down for the service.

In Phase 3, it is apparent that Google service has become better in providing services; however, it does not recover to the status before the earthquake happened. Referring to related news reports, we can find that at the time period, although the repair work of the submarine cable is mostly done, the cable remains very unstable and the communication was sometimes interrupted because of ongoing repair work.

In Phase 4, the Google service had been totally recovered and was shown to have even better capability of providing service then before the earthquake.

From this real study, we can observe that the estimated results are consistent with the real world situation. This real case study shows that our estimation approach is feasible and can reveal much more valuable information of service running.

1 `http://www.freewebs.com/jimmy_cheng/CurrencyExchangeService.wsdl`

2 `http://www.xignite.com/xCurrencies.asmx?WSDL`

Phase 1	SU	TD	STD
Time Percentage	0.872	0.0	0.129
Success Rate	1	–	0
Phase 2	SU	TD	STD
Time Percentage	0.243	0.013	0.734
Success Rate	1	0.93	0
Phase 3	SU	TD	STD
Time Percentage	0.662	0.176	0.161
Success Rate	1	0.95	0
Phase 4	SU	TD	STD
Time Percentage	0.947	0.052	0.0
Success Rate	1	0.92	0

Table 3. Availability Estimation for Google Web Search Service

6. Related Work

6.1. Definition and Measurement of Availability

Availability is now increasingly getting more and more attention in research community. Patterson et al. point out that in new century, availability and maintainability would be regarded as more important than performance[3, 11].

Research on availability can be found in many communities, such as computer architecture, network, and distributed computing. Brown and Patterson [3] have identified three different definitions of availability: binary metric at single point of time, binary metric at percentage of time, and time-dependent metric.

Researchers also addressed the problem of estimating machine availability in distributed environments. Brevik et al. [2] classify estimating approaches into two categories: parametric and non-parametric. In the non-parametric category, two methods, termed with Resample Method and Binomial method, are proposed.

In this paper, we reconsider the availability metric in service-oriented environment. We propose that the metrics of the server side's availability and the user side's availability should be differentiated. We analyze the characteristics of user-perceived service availability and identifying the limitations of traditional metrics for measuring web services' availability.

6.2. Web Service Availability

In web service domain, availability is one of the key QoS properties. Many researchers propose applications, such as (dynamic) service selection, service composition, SLA-aware middleware, and self-reconfiguration service deployment platform based on estimated service availability.

In the web service domain, two major definitions, binary metric at single point of time and binary metric at percentage of time, have been addressed. The expression of "single point of time" metric is showed in the Equation (4) [7, 10]. In Equation (4), Ava is the probability that that services will be available for use. N_a is the total times the client program has invoked the services, N_f is the number that the client program fails to access the services.

$$Ava = (N_a - N_f)/N_a \tag{4}$$

Zeng et al. [20] adopt the "percentage of time" metric as Equation (5), in which Ava is the calculated value of availability and T_{up} is the total amount of time in which the services are available during the last T seconds.

$$Ava = T_{up}/T \tag{5}$$

Equation (6) shows another similar equation for calculating availability proposed by Rosenberg et al. [13].

$$1 - T_{down}/T_{up} \tag{6}$$

In Equation (6), T_{up} is the uptime of the monitoring device (the device monitors the running of the service)[12], which is much different from the "last T" seconds in Equation (5).

Although there exist various metrics, the basic characteristics of web services' availability have not been explored. According to the definition of IEEE [5], quality of service (QoS), including availability, should represent the perception of service users. From this point of view, existing metrics are not appropriate for user-perceived availability.

The implicit assumption of the "time period" definition is that the system is accessed frequently so that the user-perceived service status at a single point of time can be regarded as the status in that short time period. In fact, in many research communities, such as software RAID systems [3], measurement in this way is reasonable because of the high-frequency accesses.

The limitation of adopting the "time percentage" definition in web service domain is that the assumption of the high frequency of accessing web services is not valid. First, round trip time of invoking services contains time for complex business handling and SOAP message (un)wrapping. In some cases, a long time interval interaction lasting more than one minute is possible (e.g., invoking a complex business process). Second, using some web services needs to be paid for invocations, which poses the barriors for the frequent accesses.

Our research differs from the preceding previous researches in two major aspects: (1) the new metric addresses the features of service invocation successes and failures, and (2) the new metric has a corresponding estimation approach, which is evaluated with real empirical data.

Doshi et al. [4] emphasize that availability is dynamically changing and propose a Bayesian Learning algorithm to keep up the estimated availability with the changing environment. Their measurement is based on counting the occurrences of the successful and failed invocation records, and new records have been assigned more weight. Their approach is evaluated with simulated data.

In our previous work [16], we proposed a status-based dynamic availability estimation algorithm. This algorithm adopts the same three-status model for dynamically estimating service status and the probability of invoking services successfully for the next invocation.

Both Doshi's work and our previous work aim to improve the accuracy of dynamically estimating the probability of being successful for the next invocation availability. Our new approach addresses a totally different issue: measuring long running service availability, which aims to make the overall evaluation on service running.

7. Conclusion and Future Work

Web services' availability has been regarded as one of the key properties for service-oriented applications. Starting from analyzing the basic features of web service availability, and based on observations on monitoring service running, we propose a new metric for web service availability. Further more, we propose an approach to estimating service availability according to the new metric. Empirical studies demonstrate that the new metric and estimation mechanisms can produce a more feasible estimation of web service availability.

In our future work, we plan to make a survey on whether the new metric is applicable for choosing services and setting up SLA. We also plan to extend the data set in the study.

8. Acknowledgement

We would like to thank Dr. Ying Pan's valuable instructions on this paper. This research was sponsored by the State 863 High-Tech Program (SN: 2006AA01Z189, 2007AA010301), and National Science Foundation of China (No.60803011,60803010).

References

[1] M. Aoyama, S. Weerawarana, H. Maruyama, C. Szyperski, K. Sullivan, and D. Lea. Web services engineering: promises and challenges. In *Proc. of the 24th International Conference on Software Engineering*, pages 647–648, 2002.

[2] J. Brevik, D. Nurmi, and R. Wolski. Quantifying machine availability in networked and desktop grid systems. Technical Report CS2003-27, U.C. Santa Barbara Computer Science Department, 2003.

[3] A. Brown and D. A. Patterson. Towards availability benchmarks: a case study of software raid systems. In *Proc. USENIX Annual Technical Conference*, pages 263–276, Berkeley, CA, USA, 2000. USENIX Association.

[4] P. Doshi, R. Goodwin, R. Akkiraju, and K. Verma. Dynamic workflow composition using markov decision processes. In *Proc. of the IEEE International Conference on Web Services*, pages 576–582. IEEE Computer Society, 2004.

[5] IEEE. *IEEE Standard Computer Dictionary: IEEE Standard Computer Glossaries.* IEEE, New York, 1990.

[6] A. Keller and H. Ludwig. The WSLA framework: Specifying and monitoring service level agreements for web services. *J. Netw. Syst. Manage.*, 11(1):57–81, 2003.

[7] N. Kokash and V. D'Andrea. Evaluating quality of web services: A risk-driven approach. In *Proceedings of the Business Information Systems Conference*, LNCS. Springer, April 2007.

[8] P. M. M.-S. L. E. Moser and W. Zhao. Building dependable and secure web services. *Journal of Software*, 2(1):14–26, Feb., 2007.

[9] M. R. Lyu and V. B. Mendiratta. Software fault tolerance in a clustered architecture: Techniques and reliability modeling. In *IEEE Aerospace Conference*, pages 141–150, 1999.

[10] M. Mikic-Rakic, S. Malek, and N. Medvidovic. Improving availability in large, distributed, component-based systems via redeployment. Technical Report USC-CSE-2003-515, December 2003.

[11] D. A. Patterson. Availability and maintainability >> performance: New focus for a new century. In *Key Note at FAST*, 2002.

[12] C. Platzer. Personal communication. 2007.

[13] F. Rosenberg, C. Platzer, and S. Dustdar. Bootstrapping performance and dependability attributes ofweb services. In *Proc. of the IEEE International Conference on Web Services*, pages 205–212, 2006.

[14] M. A. Serhani, R. Dssouli, A. Hafid, and H. Sahraoui. A QoS broker based architecture for efficient web services selection. In *Proc. of the IEEE International Conference on Web Services*, pages 113–120, 2005.

[15] L. Shao, J. Zhang, Y. Wei, J. Zhao, B. Xie, and H. Mei. Personalized qos prediction forweb services via collaborative filtering. In *Proc. IEEE International Conference on Web Services, Application Services and Industry Track*, September 2007.

[16] L. Shao, L. Zhang, T. Xie, J. Zhao, B. Xie, and H. Mei. Dynamic availability estimation for service selection based on status identification. In *Proc. IEEE International Conference on Web Services, Application Services and Industry Track*, September 2008.

[17] W. Torres-Pomales. Software fault tolerance: A tutorial. Technical Report Technical Report Tm-2000-210616, NASA, NASA Langley Research Center, October 2000.

[18] A. Tsalgatidou and T. Pilioura. An Overview of Standards and Related Technology in Web Services. *Distrib and Parallel Databases*, 12(2-3):135–162, 2002.

[19] K. Xiong and H. Perros. Trust-based resource allocation in web services. In *Proc. of the IEEE International Conference on Web Services*, 2006.

[20] L. Zeng, B. Benatallah, A. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang. QoS-aware middleware for web services composition. *IEEE Trans. Softw. Eng.*, 30(5):311–327, 2004.