# A Multiple System Performance Monitoring Model for Web Services

Yong Yang, Dan Luo, and Chengqi Zhang

Data Sciences & Knowledge Discovery Lab,
Center for Quantum Computation and Intelligent Systems.
Faculty of Engineering & Information Technology,
University of Technology, Sydney
{yongyang,dluo,chengqi}@it.uts.edu.au

**Abstract.** With the exponential growth of the world wide web, web services have becoming more and more popular. However, performance monitoring is a key issue in the booming service-orient architecture regime. Under such loosely coupled and open distributed computing environments, it is necessary to provide a performance monitoring model to estimate the likely performance of a service provider. Although much has been done to develop models and techniques to assess performance of services (i.e. QOSs), most of solutions are based on deterministic performance monitoring value or boolean logic. Intuitively, probabilistic representation could be a more precise and nature way for performance monitoring. In this paper, we propose a Bayesian approach to analyze service provider's behavior to infer the rationale for performance monitoring in the web service environment. This inference facilitates the user to predict service provider's performance, based on historical temporal records in a sliding window. Distinctively, it combines evidences from another system (For example, recommendation opinions of third parties) to provide complementary support for decision making. To our best of knowledge, this is the first approach to squeeze a final integrated performance prediction with multiple systems in Web services.

## 1 Introduction

Web services have become a promising technology for e-business and e-commerce. In such an open and volatile environment, possibly a service provider may not deliver the quality of service (QoS) it declared [1]. In the online business world, it is necessary for participants to estimate each other's performance and predict behavior of web services before initiating any commercial transactions [2,3]. Therefore it is important to monitor the service provider's performance before and during enjoying its so-claimed services.

Normally a service's performance is temporary and variable [4]. It can hardly be accurately indicated from information available because of not only incompleteness of data but also instinct uncertainty involved. For instance, the service may be delayed by incoming real-time tasks. In other words, giving a probabilistic description is more practical than a deterministic answer when estimating the performance [5,6]. The performance monitoring model proposed in this paper allows users to predict the future behavior of a service in such a way:

L. Cao et al. (Eds.): ADMI 2010, LNCS 5980, pp. 149–161, 2010.

The probability that downloading a DVD movie of *Avatar* for *bronze* membership service within 15 minutes is at least 90%.

Our model is based on Bayesian network [7,8,9,10]. A "Bayesian network" (BN) or "belief network" is a probabilistic graphical model whose nodes represent events and arcs depict probabilistic dependency relations among the events via a directed acyclic graph. Take diseases diagnosing in a medical domain as an example. A Bayesian network could represent the probabilistic relationships between diseases and symptoms, and compute the probabilities of various diseases from shown symptoms. Similarly by employing a Bayesian network, we can predict service's future behavior based on the observed historical evidences.

Indeed, most of Qos assessment algorithms tend to ignore user-specific preferences (different subitem service levels) [11,12,13]. For example, a download website service may provide quicker speed for paid members and much limited bandwidth for anonymous users. For example, at *Easy-share* [14] and *MEGAUPLOAD* [15] a premium user is given much higher priority. With regard to premium users, there are 1TB online storage with *filemanager*, whereas for non-members with none. Therefore, simply taking an average of them would neglect the relevance of user's individual feature/situation, resulting in an incorrect assessment of Qos.

Another issue worth considering is service' performance monitoring can be complemented by opinions from third parties [16,17,18]. For example, if we ignored the fact that another user left a message saying the movie is indistinct and favorless, we probably be on the road to "ruin". In other words, the performance monitoring computing can be more comprehensive when taking into account evidences from another system. In a real world situation under various restrictions, only part of evidences might be observed by the user himself or herself whereas others are done by another system such as recommendation, which surely provides useful supplementary information for decision-making. As far as we know, opinions on performance from another system could be an important complement and should be fused to get a combination value for performance monitoring computing.

This paper exploits the problem of performance monitoring prediction by user's previous behavior evidences. The theory of Bayesian network gives us robust prediction theoretical tool, which combines graph theory and probability theory. We can use it to flexibly predict service's future performance on different dimensions, which provides more accurate predictions given the circumstances. The use of Bayesian networks is similar to that of expert system technologies. A Bayesian network represents beliefs and knowledge about a particular class of situations. Given a Bayesian network (i.e. a knowledge base) for a class of situations and evidence (i.e. observations or facts) about a particular situation in that class, conclusions can be drawn in such a situation. Besides, a service may provide some functional and value-add features. These different level of services can be modeled as different status of corresponding variables in the network. And the conditional probability tables of nodes express the predicted performance in the form of probability, which provide sound basis to facilitate the user's decisions. In the meantime, when selectively integrating the cases from external systems into our Bayesian network, opinions from another system are fused to compute the complementary performance monitoring.

To sum up, this paper makes the following main contributions:

- A Bayesian network is introduced and the performance monitoring is represented by a probability value. Furthermore, performance monitoring under different dimensions is discussed and a Bayesian network performance monitoring model is proposed.
- We design a multiple-system combination performance monitoring algorithm to fuse evidences from both internal and external systems. It is capable of providing more complementary and performance trustworthy predictions.

## 2 Probabilistic Performance Monitoring Estimation in Bayesian Networks

In this section, we will first give a brief definition to our Bayesian network. Then a typical application scenario (a file download web service) will be described as an illustration. After formalizing Bayesian networks on two different levels, we will discuss performance monitoring in a multiple-system environment.

### 2.1 Overview of the Performance Monitoring Model

First of all, we define a Bayesian network generally. A Bayesian network consists of:

- A set of variables and a set directed edges between variables
- Each variable has a finite set of states
- The variables together with the directed edges from a directed acyclical graph (DAG) [19].
- For each variable A with parents $B_1, .., B_n$ there is an attached conditional probability table Pr(A|$B_1, ..., B_n$)

The fundamental rule for probability calculus is $P(a \mid b)P(b) = P(a, b)$, where $P(a, b)$ is the probability of the joint event $a \bigwedge b$, $P(a \mid b)$ is the probability of a given b.

Remembering that probabilities should always be conditioned by a context c, the formula should yield the well known Bayes's rule

$$P(b \mid a) = \frac{P(a \mid b)P(b)}{P(a)} \tag{1}$$

. Bayes's rule conditioned on c reads

$$P(b \mid a, c) = \frac{P(a \mid b, c)P(b \mid c)}{P(a \mid c)} \tag{2}$$

$P(b)$ is the prior probability of hypothesis $b$; $P(a)$ is the prior probability of evidence a; $P(b \mid a)$ is the probability of b given a. $P(a \mid b)$ is the probability of a given b.

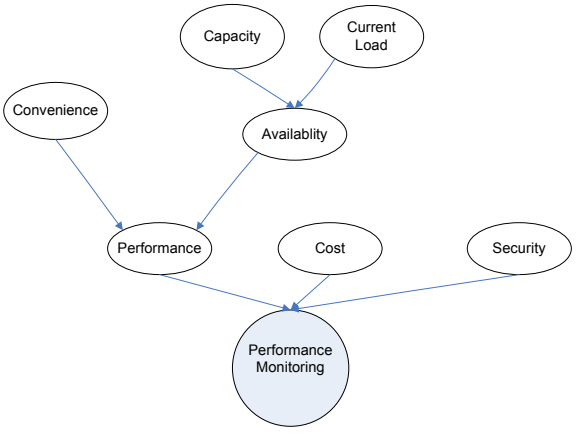


**Fig. 1.** Bayesian network performance monitoring model

In causal networks, the basic concept in the Bayesian treatment of certainties is *conditional probability* [20]. Whenever a statement of the probability $P(a)$ of an event $a$ is given, then it is given conditioned by other known factors.

Let's take file provider as an example. In the online world, file providers' competence are not uniform. For example, some file providers may link to a high-speed fabric network, while others connect through a slow modem. Some file providers might like music, so they share a lot of music files. Some may be interested in movies and share more movies. Some may be very picky about file quality, so they only keep and share files with high quality. Therefore, the file provider's capability can be presented in various aspects, such as performance, cost and security.

The Bayesian network performance monitoring model is depicted in Figure 1, which looks like a upside down tree. The root node denotes the service capability or performance monitoring value for this service, and element nodes denote users' various requirements to each Qos property. For simplicity, we assume that these Qos properties do not influence each other (That is, all the metrics are independent when estimating service's performance). In order to use BN to predict the service capability, some continuous features, such as availability and current load, should be discretized. Two approaches are presented to discretize continuous features in this article. The first approach simply specifies different levels for each Qos property in advance. For example, three levels, $0 \sim 0.5$ as low, $0.5 \sim 0.8$ as medium and $0.8 \sim 1$ as high, are specified to availability property. The second approach adopts statistical threshold.

As for the example, we simply divided the performance monitoring after each invocation into two categories: *Positive,Negative*. Likewise, *Cost* has three values: *premium membership, member, non-member*. They represent user's grades (different levels of membership) regarding to the server.

The nodes under the root node represent the file provider's capability in different aspects. Each of them is associated with a **conditional probability table** (CPT). Its CPT is showed in Figure 2. Each column follows one constraint, which corresponds to one value of the root node. The sum of values of each column is equal to 1.

| | $PerformanceMonitoring = Positive$ | $PerformanceMonitoring = Negative$ |
|---|---|---|
| $Premium$ | *p(C="Pr"\|T="Po")* | *p(C="Pr"\|T="Ne")* |
| $Member$ | *p(C="Me"\|T="Po")* | *p(C="Me"\|T="Ne")* |
| $Nonmember$ | *p(C="No"\|T="Po")* | *p(C="No"\|T="Ne")* |

C=cost,T=Performance Monitoring.

**Fig. 2.** CPT table of performance monitoring node

$p(C = \text{“}Pr'' \mid T = \text{“}Po'')$ is the conditional probability with condition that an invocation quality is judged to be positive. It measures the probability that user's levels involved in a download, given the downloading performance is satisfying. For example, the bronze user's given performance is excellent is 0.8. After each download, the network is updated. It can be calculated according to the following formula:

$$P(C = \text{“}Pr'' \mid T = \text{“}Po'') = \frac{P(C = \text{“}Pr'', T = \text{“}Po'')}{P(T = \text{“}Po'')} \tag{3}$$

$P(C = \text{“}Pr'', T = \text{“}Po'')$ is the probability that the service capability is excellent and the availability requirement is high in the past invocations.

$$P(C = \text{“}Pr'', T = \text{“}Po'') = \frac{m}{n} \tag{4}$$

m is the number of invocations that the service capability is excellent and the user's availability requirement is high. n is total invocation numbers of the service.

The root element P(T="Po") represents the probability that the service performance is regarded to be positive.

$$P(T = \text{“}Po'') = \frac{p}{n} \tag{5}$$

p is the number of invocations that invocation is regarded to be good.

P(C="Pr") is the probability that service'availability is Premium level, which is calculated according to the following formula:

$$P(C = \text{“}Pr'') = \frac{r}{n} \tag{6}$$

r is the number of invocations that the service's availability is premium level.

The conditional probability functions can be computed before next invocation as follows:

$$P(T = \text{“}Po'' \mid C = \text{“}Pr'') = \frac{P(C = \text{“}Pr'' \mid T = \text{“}Po'')P(T = \text{“}Po'')}{P(C = \text{“}Pr'')} \tag{7}$$

These predictable information is extremely important for a user's decision making. For example, we can clearly tell in what probabilities the bronze or premium level of service can have some quality of service, which is very helpful for the user to decide whether it is worth to choose them.

The variables can have internal relationship between them. For example, some of the web servers will lose its availability during the peak even for its remarkable capacities. That is, *availability* is jointly determined by *capacity* and *current-load*. We use *hierarchical* structure to reflect and represent these features.

### 2.2 General Theory

Many existing probabilistic models use naive Bayesian networks. We extend it to hierarchical structure, which is believed to be more precise to the real world. Based on the analysis of factors contributing to web service's performance, the formalized structure for our performance monitoring model is shown in Figure 3. $M_{11}, ..., M_{1a}$ represent all the aspects for evaluation of performance monitoring. Any leaf nodes in the tree may be defined recursively. However we limit our discussion to two level Bayesian networks, as any complex BN graphs can be equivalently splitted into a combination of two level models.

The all aspects regarding to performance monitoring is denoted as set of aspects $\Psi = \pi_1, \pi_2, ..., \pi_p$. The set of agents: $A = a_1, a_2, ...$ and $B = b_1, b_2, ...$ denotes the agents who have direct relationship with the target service. $C = c_1, c_2, ...$ denotes the agents who have indirect relationship with the target service. D denotes the rest of them.

The set of ratings p can be viewed as a mapping from $A \times \Psi$ to a real number between 0 and 1:

$$p = A \times \Psi \rightarrow [0, 1] \tag{8}$$

The chain rule for Bayesian networks. Let $U = \{A_1, ..., A_n\}$ be a universe of variables. Then the joint probability distribution $P(U)$ is the product of all potentials specified in BN

$$P(u) = \prod_i P(A_i \mid pa(A_i)) \tag{9}$$

where $pa(A_i)$ is the parent set of $A_i$.

We can use Eq. 9 to estimate the uncertainty of variables we are interested in. Tractability is not a yes or no issue. Intuitively, these equation still hold even for hierarchical network. After each interactive with this web service, the performance monitoring network is updated. By cumulative effects of the evidences, we can predict service's future behavior. This procedure is named *training*. Furthermore, with this prediction, we can build user-based performance monitoring model for this services. Based on user's history, service can offer user-specified intelligent service which is greatly convenient for the user and assist them with the proper decision.

There are algorithms for probability updating in bayesian networks, however, basically probability updating in NP-hard. This means that some models have an updating time exponential in the number of nodes.

### 2.3 Representing Multiple-System Combination Performance Monitoring in Bayesian Networks

When computing the performance, it is important to combine the evidences from another system, such as recommendations. In other words, the predicted performance can be refined by considering recommendations.

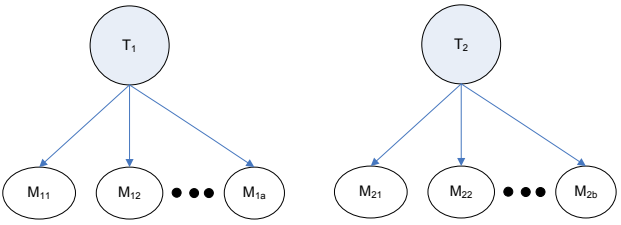


**Fig. 3.** Two level bayesian networks

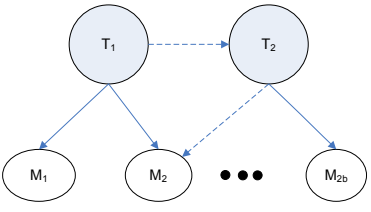


**Fig. 4.** Bayesian network with combination performance monitoring

We use our bayesian network to alert the user's requirements whether there is any change about certainty. This network can be used to follow how a change of certainty in one variable may change the certainty for other variables. We propose a set of rules for that way of reasoning. The rules are independent of the particular calculus for uncertainty.

For example, service 1 is a movie download web service. The opinions from users may leave comments for it (e.g., the downloading speed is rather slow or this movie is poor-quality). Afterwards that information clearly is helpful for users to estimate the performance monitoring. Internal and external evidences are merged and thereby the performance monitoring model can be refined to Figure 4. The solid lines represent all the direct invocations with the target service. In contrast, the solid lines represent indirect experiences with target services.

$$T = (1 - \theta)T_B + \theta T_C \tag{10}$$

where $\theta \in [0, 1]$. The weight $\theta$ represents the importance of these two probabilities respectively and are decided by the personal characteristics of the entity.

With respect to modeling the recommendation, we use the following formula to its estimator:

$$p(\hat{\theta}) = Beta(c_1, c_2) \tag{11}$$

where $c_1$ and $c_2$ are parameters determined by prior assumptions and $\hat{\theta}$ is the estimator for the true proportion of a's approval of b based on invocation between them. Assuming that each invocation's approval probability is independent of other invocations between a and b, the likelihood of having p approvals and (n-p) disapprovals, which can be modeled as:

$$L(D|\theta) = \theta^p(1-\theta)^{n-p} \tag{12}$$

$$p(\theta|D) = Beta(c_1 + p, c_2 + n - p) \tag{13}$$

$$E[\theta|D] = \frac{c_1 + p}{c_1 + c_2 + n} \tag{14}$$

$$\sigma^2_{\theta|D} = \frac{(c_1 + p)(c_2 + n - p)}{(c_1 + c_2 + n - 1)(c_1 + c_2 + n)^2} \tag{15}$$

How is the prior belief about $\theta$ created? There are two ways:

$$p(\theta) = \begin{cases} 1 & 0 < \theta < 1 \\ 0 & otherwise \end{cases} \tag{16}$$

For the Beta prior, values of $c_1 = 1$ and $c_2 = 1$ yield a uniform distribution.

After each invocation, the client updates $\theta$ and corresponding knowledge in the Bayesian network. It is a new case for this learning. But it is often a problem for updating the CPTs in the Bayesian network because the initial counts are kept when the network try to accommodate new cases. Particularly when the conditional probabilities in the environment change over time, the accumulated counts will prevent the network from new changes. Therefore, the vacuous counts will build up to make parameters too resistant to change. It is known that the performance monitoring of service provider is on a temporary basis as the behavior of service provider is constantly changing with time. The value of past performance monitoring does not represent the current performance monitoring. To this end, we set up a parameters k as the **window size** of the learning procedure. All the case are updated in the fixed size window. When the cases in the window is greater than k, the newest case is put into the window and the oldest one are swapped out. The performance monitoring in Eq. 10 is described below. In order to avoid the duplicate cases observed by indirect and direct users, a hash table is used to record what has been observed by $B$ and avoid any duplicates.

**Algorithm 1.** Bayesian network performance monitoring

**Input**: Initial setting, structure of Bayesian network,invocation records.
**Output**: $PerformanceT$

(a). setup and system initiation. Set the window size. Create a table HashTl for the store the cases objected.
(b). compute $T_B$ using bayesian network.
(c). compute $T_C$ for recommendation from others.
(d). compute $T$ for this invocation.
(e). search Hash(case) in HashTl, if there is a hit, ignore it and return step 2. Otherwise add a new record to HashTl and do the invocation.
(f). After each invocation, the learning algorithms is used to update the Bayesian network and compute T.

Probability learning algorithm of bayesian network is described below (We assume that the network itself is constructed using a priori knowledge):

**Algorithm 2.** Bayesian network learning

**Input**: $T_n$
**Output**: $T_{(n+1)}$

(a). compute the probability of $T_{B(n+1)}$
(b). if $|T_{Bn} - T_{B(n+1)}| < |T - T_{B(n+1)}|$ then
    n++;
    compute Eq. 13
  else
    n++;
    p++;
    compute Eq. 13.
  end if
(c). compute the new $\theta$ and $T_{(n+1)}$.

Then we take priority of evidence from set B because people trust themselves intuitionally. But on the other hand, they probably can not observe all the cases from the target service. Under that circumstantiates, the recommendation plays an important role in the performance monitoring estimation. In order to speed up the classification of cases, a hash table data structure is designed in table 1.

**Table 1.** Hash table H for case history

| $id$ | $casesummary$ | $Time$ | $HashValue$ |
|---|---|---|---|
| 1 | ... | *081022:2456* | *FFEFs3va* |
| 2 | ... | *081022:2456* | *df12456* |
| 3 | ... | ... | ... |

## 3 Analysis

A Bayesian network serves as a model for networks occurring in trust estimation, and the relations in the model reflect causal impact between events. The reason for building these computer networks is to use them when taking decisions. The probabilities provided by the network are used to support decision making. To that end, we can distinguish forward-looking and backward-looking approaches. Although the examples presented in this paper are very elementary only, they serve as building blocks for these types of decisions.

When making decision in a complex situation, a proper loss function as well as a set of possible actions have to be taken into consideration. We make a distinction

here between intervening actions, which force a change of state for some variables in the model, and non-intervening actions, whose impacts are not a part of the model. Intervening action may also force the direction of causality to be inverted. It is beyond the scope of this paper to extend much beyond this notion. The interested reader is refereed to [8] for further study.

A Bayesian network is also useful for illustrating the effects of different scenarios. For that purpose, stochastic simulations are useful. It can then be analysed what the effects are if some actions are taken. For example, the effects of lowering a threshold on the *availability* feature of a service may be illustrated.

Finally, Bayesian networks can fit well into a built-in hierarchical model. Hierarchical structure is very useful considering the complex relationships among real world factors.

## 4 Case Study

In this section we present the work done as a case study on real-life data. To evaluate the effectiveness of our approach in estimating trust probability in different web service scenarios, we implement a prototype in Java. The system architecture is depicted in Figure 5.

We have carried out several experiments to study the effectiveness of our model. The environment is file downloading web services running on a Pentium computer with 3.40 GHz dual CPUs. The RAM is 1 G. and LAN connection is 100 M bps Ethernet card. Operation system is windows XP with SP3 pack. The service's performance we predicted is described as a trust value (boolean variable, positive or negative).

The first study is to check whether our model can identify user-specific preference. We invoked the service for 100 times with difference system trust. Table 2 shows the

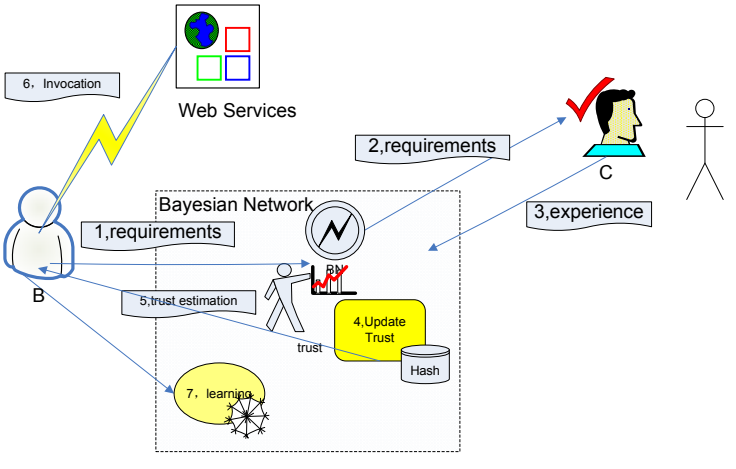


**Fig. 5.** System architecture

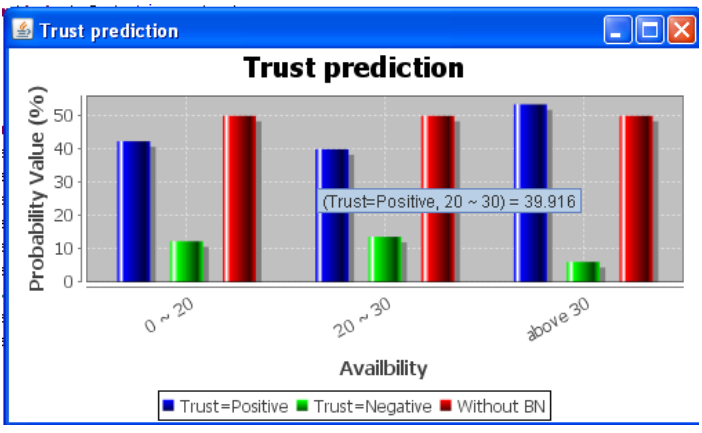


**Fig. 6.** Prediction result of bayesian network

**Table 2.** Service with two different levels

| | *Availability* | | |
|---|---|---|---|
| | *Min* | *Max* | *Average* |
| *Level1* | *0* | *20* | *10* |
| *Level2* | *20* | *30* | *25* |

two different service levels with availability feature. The user-specific duration was modeled into 2 levels: $0 \sim 20$ (level 1) and $20 \sim 30$ (level 2). When we did the 1000 invocations, we calculated the posterior probability distribution with 3 intervals: (0,20),[20,30),[30,$\infty$).


**Fig. 7.** Trust estimation with/without BN

The prediction result by Bayesian network is calculated in Figure 6. It can provide more informative prediction by considering user's preferences. For instance, there is 39.916 percent of chance that overall judgement of this service's performace is positive given that availability is in the interval $20 \sim 30$.

The next experiment is to see if our BN model has improve the performance estimation of the services. The BN model is compared to without BN selection, in Figure 7. Our service provider provides level 1 availability services (within $0 \sim 20$) for 800+ invocations. (The small red squares depict introduced evidences assumed from third parties.) The result shows that BN model presents a more stable and reliable (closer to true value, fitting in with availability *Level 1* range) description about service's performance.

## 5 Conclusion

Bayesian network is a probabilistic model to represent relations between attributes and classes. We have described an application of Bayesian networks, a relatively new technology for probabilistic representation and inference, to web service performance monitoring. The advantages that Bayesian networks bring to the performance prediction task include an intuitive representation of uncertain relationships and a set of efficient inference algorithms. Probabilistic models view performance prediction as a problem of estimating the probability that the service provider matches or satisfies user's requirements. The salient feature of this model is that it can correctly predict the service performance on the consideration of user's specific requirements. Furthermore, this model combines evidences from multiple sources, which places higher level of trust intuitively.

Our network structure is a complete graph. A disadvantage of this model is that the representation of this model is exponential with respect to the number of attributes. Consequently, it is difficult to estimate exponential number of parameters from limited data and perform computations with this model.

As future work, we plan to explore our model into composition of web services.

## References

1. Menascé, D.: QoS Issues in Web Services. IEEE Internet Computing, 72–75 (2002)
2. Dasgupta, P.: Trust as a commodity. Trust: Making and Breaking Cooperative Relations, 49–72 (1988)
3. Baier, A.: Trust and Antitrust. Ethics 96(2), 231 (1986)
4. Zak, P.: Trust: A temporary human attachment facilitated by oxytocin. Behavioral and Brain Sciences 28(03), 368–369 (2005)
5. Hilden, J., Habbema, J., Bjerregaard, B.: The measurement of performance in probabilistic diagnosis. II. Trustworthiness of the exact values of the diagnostic probabilities. Methods Inf. Med. 17(4), 227–237 (1978)
6. Despotovic, Z., Aberer, K.: A Probabilistic Approach to Predict Peers' Performance in P2P Networks. LNCS, pp. 62–76. Springer, Heidelberg (2004)
7. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann Publishers, San Francisco (1988)
8. Jensen, F.V.: Bayesian Networks and Decision Graphs. Springer, Heidelberg (2001)

9. Pearl, J.: Causality: Models, Reasoning, and Inference. Cambridge University Press, Cambridge (2000)
10. Heckerman, D., Geiger, D., Chickering, D.: Learning Bayesian networks: The combination of knowledge and statistical data. Machine Learning 20(3), 197–243 (1995)
11. Welch, L., Shirazi, B.: A Dynamic Real-Time Benchmark for Assessment of QoS and Resource Management Technology. In: The IEEE Real-Time Technology and Applications Symposium, pp. 36–45 (1999)
12. Ito, Y., Tasaka, S.: Quantitative assessment of user-level QoS and its mapping. IEEE Transactions on Multimedia 7(3), 572–584 (2005)
13. Bernardi, S., Merseguer, J.: QoS Assessment via Stochastic Analysis. IEEE Internet Computing, 32–42 (2006)
14. Easy Share, `http://www.easy-share.com/`
15. Megaupload, `http://www.megaupload.com/`
16. Doney, P., Cannon, J.: An Examination of the Nature of Trust in Buyer-Seller Relationships. Journal of Marketing 61, 35–51 (1997)
17. La Porta, R., De Silanes, F., Shleifer, A., Vishny, R.: Trust in Large Organizations. In: Nber Working Paper (1996)
18. Massaro, D., Friedman, D.: Models of integration given multiple sources of information. Psychological Review 97(2), 225–252 (1990)
19. Diestel, R., Diestel, R.: Graph theory. Springer, New York (2000)
20. Jensen, F.: An introduction to Bayesian networks. UCL Press, London (1996)