QoS-Driven Service Matching Algorithm Based on User Requirements

Mengying Guo(✉) and Xudong Yang

School of Computer Science, Beijing University of Posts and Telecommunications, Beijing, China
{mengying_1204,xdyang}@bupt.edu.cn

Abstract. Quality of Service (QoS) is an important factor which should be considered in service matching. There are two problems in most existing solutions. Firstly, most QoS models are static model described by determinate values or probability distributions, ignoring the impact of time factor. However, most QoS attributes are time-dependent, such as response time and reliability. Secondly, the service selection criteria of most QoS-driven service matching algorithms are based on service performance, but user requirements and the load of services are not considered. In this paper, we propose a Time-Segmented QoS Model (TSQM) to dynamically model QoS. Based on this model, a Service Matching algorithm based user QoS request and Priority (QPSM) is proposed. The priority of user requests is used to control the load of the services. Simulation results show that the algorithm can achieve a higher response rate and a better effect of load balancing.

Keywords: Service matching · QoS · Dynamic QoS model
Service model · Load balancing

1 Introduction

SOA (Service-Oriented Architecture) has provided a possibility for IoT (Internet of Things) systems to build distributed applications by loosely coupled services [1]. IoT services can be provided for different systems as web services by this way. Selecting services in numerous registered services has become difficult with the number of IoT services increasing rapidly [2]. The characteristics of IoT services determine that service function and service quality must be taken into account simultaneously when performing service matching. QoS (Quality of service) measured in different criterions such as delay, response time, reliability, availability, cost, etc. [3], has been a crucial factor in selecting services from numerous services with the same functions. The results of service matching depend not only on the matching degree to user requirements but also on the QoS attributes of the service itself. QoS-aware service selection is a complex multi-criterion decision problem, which is called NP-hard problem, and it is still a challenging research [4].

J. Vaidya and J. Li (Eds.): ICA3PP 2018, LNCS 11336, pp. 17–27, 2018.
https://doi.org/10.1007/978-3-030-05057-3_2

There have been many reasonable selection models and effective matching algorithms for QoS-aware service selection. In these models and algorithms, service matching is considered as an optimization problem based on service selecting and the objective is to find the best service. However, the fact that actual requirements of users are not considered is unacceptable for some users, because the matched services may have the best overall performance but cannot satisfy the user requirement for a certain QoS attribute. Another problem of these models is that the QoS attributes are only represented with single-valued model or probabilistic model and the influence of time is not taken into account. Because the service QoS attributes dynamically change with time and user load, the static model cannot accurately represent the QoS values. Thereby the static model will seriously affect the accuracy of matching results.

In this paper, by splitting time and dynamically modeling each time period, we propose a Time-Segmented QoS Model (TSQM) which can represent QoS attributes more accurately. Based on our model, a Service Matching algorithm based user QoS request and Priority (QPSM algorithm) is proposed. In this algorithm, the single QoS performance and comprehensive QoS performance provided by services are considered simultaneously. The load of the service is controlled according priority, so that the purpose of balancing user load on each service can be achieved. The rest of the paper is organized as follows. Section 2 introduces the related work of service matching technology. Section 3 details the TSQM model and the QPSM algorithm. Section 4 shows the simulation results to prove the feasibility and effectiveness of the QPSM algorithm. Section 5 concludes this paper.

2 Related Work

QoS-based service matching can usually be divided into two relatively independent processes, service selection and service ranking [5]. Service selection ensures the most basic functional requirements and QoS requirements of users or systems. Service ranking is a further optimization on this basis. The model and algorithm of service selection can be divided into service-function-based selection and service-quality-based selection according to different selection criteria. In service-function-based model, the concepts such as semantics or ontology are used to build service models [6,7]. The service-quality-based selection can be divided into single QoS performance selection model and comprehensive QoS performance selection model [5]. The service-quality-based selection can also be divided into single value model and probability-based selection model [8–10].

Service function is one of the requirements that should be satisfied in the process of service matching. The fundamental purpose of service matching is to select the most appropriate service for the user based on the service request from the user. More and more models describe and define services based on semantic web and ontology to understand the functional requirements of users more intelligently. A new resource model describing IoT resources in multi-dimensional was proposed in [6]. Based on this model, a resource matching algorithm, that select suitable resource according the similarity between semantic web matching resources, was also proposed. In [7] authors proposed a QoS-based dynamic service composition method in semantic IoT. According to the context-added QoS ontology, after the dynamic semantic annotation of the services in semantic internet of things, the candidate service sets are dynamically selected and combined to provide more accurate services.

Service quality is another requirement that should be satisfied in the process of service matching. The QoS attributes of services will significantly impact on the comprehensive evaluation of services. Therefore, QoS-based service selection is an available scheme of IoT service selection. In most studies, such as [8,9], single-valued model or probabilistic model are usually used to model each dimension of QoS, and the optimal services are selected according to the comparison of service performance. In the process of QoS-aware service matching, not only the overall performance of the service but also each user requirement of QoS should be considered. In [10] authors proposed a service discovery algorithm based on a multi- stage service matching algorithm. In this algorithm, each QoS attribute is assigned a different weight and the QoS constraints are determined according to user requests. Finally, the most suitable service is selected. The QoS of web service dynamically changes with factors such as network condition, user load and time. Static model constructed solely from historical data cannot accurately represent the dynamic changes. Therefore, the time factor must be considered when modeling.

3 Service Model

In a complete process of service matching, the function and quality of service should be taken into consideration. Assume that the virtual service set S is known and all services in the virtual service set S can satisfy the functional requirements requested by the user. Next, the QoS modeling and service matching will be discussed further.

3.1 Time-Segmented QoS Model Definition

The TSQM model is a time-segmented QoS-based model. According to changes of QoS attributes over time, the QoS change period can be divided into some time periods with different intervals and the QoS model can be constructed separately in each time period.

Definition. The TSQM model for a service can be represented as a triple (ET, P, QM), where

- $ET = [T_0, T_0 + T)$ is the effective period of QoS, T_0 is the start time of effective period, T is the time period of QoS attribute updated.
- $P = \{P_1, P_2, \cdots, P_N\}$ is the time period of ET, $P_i = [t_i, t_{i+1})$ and $\bigcup_i P_i = ET$.

- $QM = \langle Q_1, Q_2, \cdots, Q_n \rangle$ is a sequence of QoS models, $Q_i = (f_{DELAY_i}, f_{REST_i}, f_{REL_i}, f_{USA_i}, f_{COST_i})$ is the QoS vector of the time period P_i, and f_{DELAY_i}, f_{REST_i}, f_{REL_i}, f_{USA_i}, f_{COST_i} represent the probability distribution function of delay, response time, reliability, availability, and cost.

Given a service, the QoS model of the service can be represented as $Q(t) = (f_{DELAY_t}, f_{REST_t}, f_{REL_t}, f_{USA_t}, f_{COST_t})$, where $t \in [t_i + kT, t_{i+1} + kT), k = 0, 1, \cdots$

The TSQM model shows that the QoS of the service changes with time. The model can be flexibly extended according to different user requirements, and the number of QoS attributes in each time period can be one or more. In this paper, delay, response time, reliability, availability and cost are selected as the QoS attributes.

3.2 Detailed Description of the Model

QoS Model. A QoS model of a service contains k QoS attributes. These attributes can be 5 non-functional attributes defined in the TSQM model, and they can also be extended according to user requirements. The QoS of service S_i corresponds to a set of QoS vectors consisting of a probability distribution function at each time period. In order to compare the QoS performance more easily, the probability distribution function in each time period should be converted into a determined value using the 999 criterion (choose a value that 99.9% of the data satisfies as the QoS value of the current time period), i.e., $f_{QoS_i} \rightarrow q_i$.

For clear expression, the below-mentioned QoS attributes default to QoS attributes within a certain time period. The QoS attributes of service S_i can be represented as a vector, i.e., $Q_i = (q_{i1}, q_{i2}, \cdots, q_{ik})$, where q_{ik} is a value converted from the probability distribution function of the k-th QoS attribute. We assume that the virtual service set consists of n candidate services, $S = \{S_1, S_2, \cdots, S_n\}$, and their corresponding QoS attributes can be represented as an $n \times k$ matrix.

$$M = \begin{bmatrix} q_{11} & q_{12} & \cdots & q_{1k} \\ q_{21} & q_{22} & \cdots & q_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ q_{n1} & q_{n2} & \cdots & q_{nk} \end{bmatrix} \tag{1}$$

Because of the differences in the range of QoS values and the effect on the comprehensive service performance, the QoS values should be normalized by the min-max normalization [11]. According to the impact on the comprehensive performance of the service, QoS attributes can be classified into positive effect attributes and negative effect attributes. The larger value of positive effect attributes (such as reliability, availability, reputation and other attributes) or the smaller value of negative attributes (such as cost, response time, and other attributes), the better overall performance of the service. Assuming that the range of q_i is $[min(q_i), max(q_i)]$, positive and negative effect attributes should be normalized by formula (2) and (3) respectively.

$$q_i^{'} = \begin{cases} \frac{q_i - min(q_i)}{max(q_i) - min(q_i)}, & max(q_i) - min(q_i) \neq 0 \\ 1, & max(q_i) - min(q_i) = 0 \end{cases} \tag{2}$$

$$q_i^{'} = \begin{cases} \frac{max(q_i) - q_i}{max(q_i) - min(q_i)}, & max(q_i) - min(q_i) \neq 0 \\ 1, & max(q_i) - min(q_i) = 0 \end{cases} \tag{3}$$

All QoS values are distributed between $[0, 1]$ after normalization. The comprehensive performance of the service is enhanced with the increase of each QoS value, that is, the larger the QoS value, the better the service performance.

Service Request. A service request sent from the user to the service platform when the service discovery is performed can be represented as $Req = \{Q_{req}, M_{req}\}$, where $Q_{req} = (\alpha_1, \alpha_2, \cdots, \alpha_k)$ is a QoS request vector and $\alpha_1, \alpha_2, \cdots, \alpha_k$ represent the user's expected values for k attributes $q_{i1}, q_{i2}, \cdots, q_{ik}$. The QoS values in the request vector, $\alpha_1, \alpha_2, \cdots, \alpha_k$, should be normalized by formula (2) or (3), so we can get $\alpha_1^{'}, \alpha_2^{'}, \cdots, \alpha_k^{'}$. Then Q_{req} is converted to $Q_{req}^{'}$. The priority vector is $M_{req} = (m_1, m_2, \cdots, m_j)$, $j \in \{1, 2, \cdots, k\}$, and j means the j-th attribute in Q_{req} as the priority attribute of the request Req. M_{req} including one or more priority attributes is defined by the user requirements, which fully reflects the user's preference for the QoS attributes of the target service. The user requirement emphasizes the importance of the j-th attribute $q_j^{'}$ in the target service. And $q_j^{'}$ is expected to satisfy the requirement of $\alpha_j^{'}$ in $Q_{req}^{'}$ as much as possible, i.e., $q_j^{'} \geq \alpha_j^{'}$.

Priority. The priority of the service request depends on $\alpha_j^{'}$ in the QoS request vector $Q_{req}^{'}$. Suppose h is the user's expected value of a certain QoS attribute, i.e., $h = \alpha_j^{'}$. The priority of the request can be calculated by formula (4).

$$Prior(h) = \begin{cases} 1, & h \in [0, T_1) \\ 2, & h \in [T_1, T_2] \\ 3, & h \in (T_2, 1] \end{cases} \tag{4}$$

T_1 and T_2 are single performance thresholds that is used to determine the priority of the service request. The values of T_1 and T_2 are in the range of [0, 1], and $T_1 \leq T_2$. The priority of the service request Req can be divided into three levels of 1, 2, and 3, which respectively represent the low, medium, and high of the priority. According to the request priority, different matching strategies are selected. The matching strategy set can be represented as $MS = \{MS_H, MS_M, MS_L\}$, where MS_H,MS_M and MS_L respectively indicate the matching strategies of different priority.

QoS Performance Evaluation Value. QoS performance evaluation value is classified to request performance evaluation value QoS_{req} and service performance evaluation value QoS_{ser}. QoS_{req} is selected by the expected QoS value from user and it can be represented as $QoS_{req} = \left|Q'_{req}\right|^2 = \alpha_1'^2 + \alpha_2'^2 + \cdots + \alpha_k'^2 = \sum_{i=1}^{k} \alpha_i'^2$, where $Q'_{req} = \left(\alpha'_1, \alpha'_2, \cdots, \alpha'_k\right)$ is the QoS request vector after normalization. The QoS_{ser} of service S_i can be represented as $QoS_{ser}(i) = \left|Q'_i\right|^2 = q_{i1}'^2 + q_{i2}'^2 + \cdots + q_{ik}'^2 = \sum_{j=1}^{k} q_{ij}'^2$, where $Q'_i = \left(q'_{i1}, q'_{i2}, \cdots, q'_{ik}\right)$ is the QoS attribute vector after normalization.

The Utility of Service Matching. $U(i)$ is the utility of the service matching algorithm when the service S_i is selected as the target service satisfying the request *Req*. It is classified to single performance utility value $U_S(i)$ and comprehensive services utility value $U_C(i)$. $U_S(i)$ is the ratio of a certain QoS attribute of *Req* to that of S_i, and can be represented as formula (5). $U_C(i)$ is the ratio of the overall performance evaluation value of *Req* to that of S_i, and can be represented as formula (6). $U(i)$ is the weighted sum of $U_S(i)$ and $U_C(i)$, and it can be represented as formula (7).

$$U_S(i) = \begin{cases} h/q'_{ij}, & h < q'_{ij} \\ q'_{ij}/h, & h \geq q'_{ij} \end{cases} \tag{5}$$

$$U_C(i) = \begin{cases} QoS_{req}/QoS_{ser}(i), & QoS_{req} < QoS_{ser}(i) \\ QoS_{ser}(i)/QoS_{req}, & QoS_{req} \geq QoS_{ser}(i) \end{cases} \tag{6}$$

$$U(i) = \mu \times U_S(i) + (1 - \mu) \times U_C(i) \tag{7}$$

The μ is weighted factors in the range of [0, 1]. The impact of $U_S(i)$ and $U_C(i)$ on $U(i)$ can be adjusted through μ. In the matching process, the greater utility, the more matched with the user requirements the service is.

4 Service Matching Algorithm

The QoS-based service matching algorithm can be roughly classified to two methods: single-QoS performance matching and overall-QoS performance matching. In the QPSM algorithm, service selection and matching are performed according to user-defined priority attributes and QoS. So the most suitable service to user requirements can be matched.

QPSM algorithm is proposed as Algorithm 1. The main idea of the algorithm is selecting the corresponding matching strategy according to the priority of user request, and selecting the service that is most suitable to the user. The priority of user request is determined by the specified priority attributes, and the different matching strategies are adopted according to the priority. When the request priority is determined as a high priority, the target service must satisfy

Algorithm 1. QoS-based service matching algorithm (QPSM)

```
Input: (1)$S$ // Service Set
       (2)$Req$ // User Requirements
Output: $Ser\_match$ // All services that suit for user
Initialize $Req$, $S$ and its corresponding QoS attribute matrix $M$;
Determine the priority of the request;
Compose priority service set $Ser\_prior : q_{ij}^{'} \geq h$;
Compose the candidate service set $Ser\_wait : QoS_{ser}(i) \geq QoS_{req}$;
while $Req$ is not empty do
    if $Prior(h)=3$ then
        if $Ser\_prior = \varnothing$ then
            $Ser\_match \leftarrow null$
        else
            $Ser\_match \leftarrow$ the largest $QoS_{ser}(i)$ from $Ser\_prior$
        end
    end
    if $Prior(h)=1$ then
        if $Ser\_wait = \varnothing$ then
            $Ser\_match \leftarrow$ the largest $QoS_{ser}(i)$ from $S$
        else
            $Ser\_match \leftarrow$ the minimum $QoS_{ser}(i)$ from $Ser\_wait$
        end
    end
    if $Prior(h)=2$ then
        if $Ser\_prior \neq \varnothing$ and $Ser\_wait = \varnothing$ then
            $Ser\_match \leftarrow$ the largest $QoS_{ser}(i)$ from $Ser\_prior$
        end
        if $Ser\_prior = \varnothing$ and $Ser\_wait \neq \varnothing$ then
            $Ser\_match \leftarrow$ the largest $q_{ij}^{'}$ from $Ser\_wait$
        end
        if $Ser\_prior = \varnothing$ and $Ser\_wait = \varnothing$ then
            $Ser\_match \leftarrow$ the largest $U(i)$ from $S$
        end
        if $Ser\_prior \neq \varnothing$ and $Ser\_wait \neq \varnothing$ then
            if $Ser\_inter = Ser\_prior \cap Ser\_wait \neq \varnothing$ then
                $Ser\_match \leftarrow$ the largest $U(i)$ from $Ser\_inter$
            else
                if $Ser\_union = Ser\_prior \cup Ser\_wait \neq \varnothing$ then
                    $Ser\_match \leftarrow$ the largest $U(i)$ from $Ser\_union$
                end
            end
        end
    end
end
return $Ser\_match$;
```

the priority attributes completely with the user requirements. When the request priority is judged as a low priority, a service with the smallest service performance evaluation value which satisfies the user request performance evaluation value is selected. So the load of the entire service system is balanced and the optimized matching of resources is achieved. When the request priority is judged as a medium priority, the user request and service performance are weighed, and the service selection is determined by the utility of service matching.

Ser_match, a matching service set, is composed of services selected by priority attributes. When the number of priority attributes is more than one, a conflict of matching policy selection may occur. The merging of matching services is to merge the services in *Ser_match* and finally the most suitable service is selected for the user. Algorithm 2 shows the whole procedure of matching service merging.

Algorithm 2. Merge matching service

```
Input: $Ser\_match$ // Matching Service Set
Output: $Ser\_result$ // The most suitable service for users
Initialize $\alpha^{'} \in \{\alpha_1^{'}, \cdots, \alpha_k^{'}\}, i \in \{1, \cdots, n\}, j \in \{1, \cdots, k\}$;
for $Ser\_match \neq \varnothing$ do
    if $num(Prior(\alpha^{'}) = 3) \geq 1$ then
        if $num(Ser\_match(q_{ij}^{'} \geq \alpha_j^{'})) \geq 2$ then
            $Ser\_result \leftarrow$ the largest $U(i)$ from $Ser\_match(q_{ij}^{'} \geq \alpha_j^{'})$
        end
        if $num(Ser\_match(q_{ij}^{'} \geq \alpha_j^{'})) = 1$ then
            $Ser\_result \leftarrow Ser\_match(q_{ij}^{'} \geq \alpha_j^{'})$
        end
        if $num(Ser\_match(q_{ij}^{'} \geq \alpha_j^{'})) = 0$ then
            $Ser\_result \leftarrow$ null
        end
    end
    if $num(Prior(\alpha^{'}) = 3) = 0$ then
        if $num(Ser\_match) \geq 2$ then
            $Ser\_result \leftarrow$ the largest $U(i)$ from $Ser\_match$
        else
            $Ser\_result \leftarrow Ser\_match$
        end
    end
end
return $Ser\_result$;
```

5 Experiment Analysis

The main purpose of the QPSM algorithm is to select the most suitable service for the user according to user-defined QoS request. In order to verify the feasibility and effectiveness of this algorithm, it is compared with the other two QoS-based matching algorithms, Single-QoS and Overall-QoS, in four aspects that is response rate, load, average single performance value and overall performance value. All the experiments were conducted on a computer with a 3.2

GHz Intel Core 2 Duo CPU and 12 GB RAM. The data used for the experiment derived from two sources: a data set containing 1000 actual services and 5 QoS values, and a randomly generated user request data set.

The purpose of the first experiment is to evaluate the response rate of the algorithm, that is the ratio of successfully matched and returned requests to the total requests. In this experiment, 100 services are selected for matching and 1000 service requests are randomly generated. The response rates of this three algorithms are shown in Fig. 1. As the number of user requests increase, the response rate of each algorithm tends to be stable. The QPSM algorithm outperforms other algorithms with the highest response rate at about 96%. However, the response rate of the Single-QoS algorithm [8] is the lowest at about 88%. The reason for this result is that the Single-QoS algorithm will fail to respond when all candidate services do not satisfy the QoS constraints. The Overall-QoS algorithm [10] will fail to respond when the overall performance is lower than user request performance. In QPSM algorithm, the matching results will be found through a comprehensive consideration of user requirement and service performance.

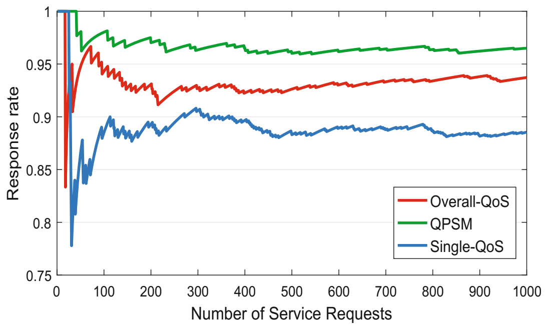

Fig. 1. The response rate of the algorithm with the number of user requests

The second experiment is to evaluate the effect of load balancing, that is indicated by the number of times that services with different QoS performance respond to requests. In this experiment, 5 candidate services with the same function and the different QoS are selected and 1000 service requests are randomly generated. The distributions of service load by using traditional UDDI [5] algorithm and QPSM algorithm are compared. And the load distributions of QPSM algorithm with different single performance thresholds T_1 and T_2 are tested. Figure 2 shows that the QPSM algorithm outperforms the UDDI algorithm in term of load balancing when the number of service requests is the same. The greater difference between T_1 and T_2, the better performance of load balancing. Because the greater difference between T_1 and T_2, the more service requests are judged to be medium priority, and the effect of load balancing is better.

The third experiment is to evaluate the average service single-performance value and the overall-performance value. In this experiment, 1000 services used for matching are selected and 1000 user requests with high demand for response

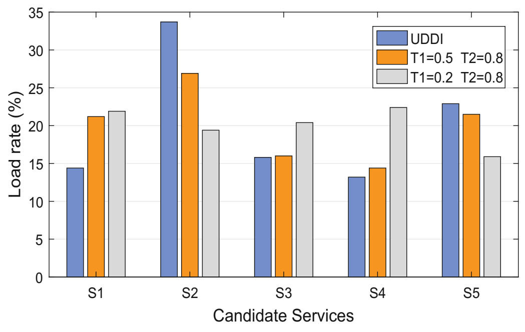

Fig. 2. Distribution of service matching load rate

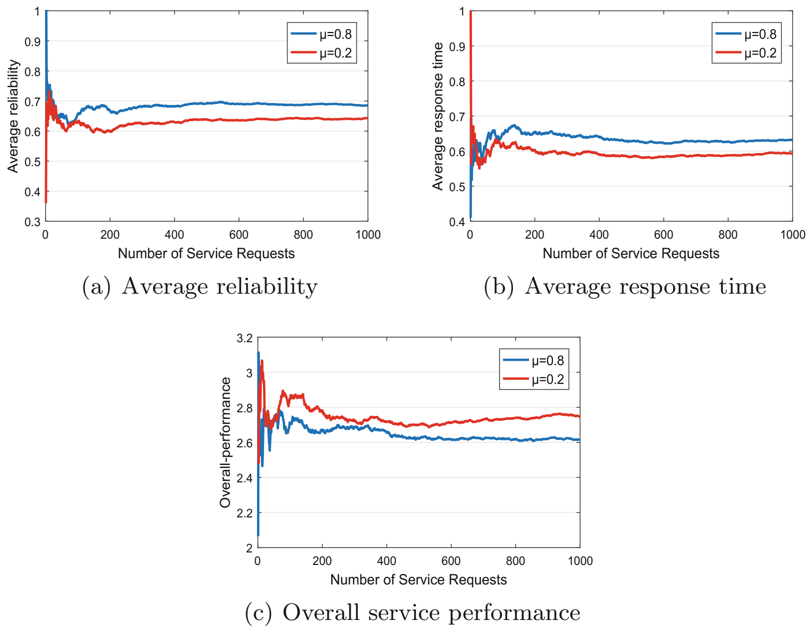

Fig. 3. Service single-performance and overall-performance with the number of user requests

time and reliability are randomly generated. The μ in the service matching utility $U(i)$ is taken as $\mu = 0.2$ and $\mu = 0.8$ respectively. Figure 3 shows that the larger μ, the higher average reliability of the matching service, the shorter response time, and the lower overall service performance value. Because the value of μ determines the proportion of single performance utility value $U_S(i)$ and comprehensive services utility value $U_C(i)$ in the utility of service matching $U(i)$, and affects the final service selection further. The users can select the appropriate μ according to their requirements.

6 Conclusion

Due to the uncertainty caused by the dynamic change of service QoS and the ambiguity of user requirements, there are some limitations in the current service matching algorithms. In order to describe the QoS attributes more accurately, we propose a time-segmented QoS model on the consideration of time. Based on this model, a service matching algorithm based on user QoS request and priority is also proposed. In this algorithm, user requirements and QoS performance preferences is fully considered. And the most suitable service is selected according to user-defined service requests and priorities, which is more suitable for users with specific requirements. Finally, experimental results indicate that the proposed algorithm can achieve a higher response rate and a better effect of load balancing.

References

1. Benslimane, D., Dustdar, S., Sheth, A.: Services mashups: the new generation of web applications. IEEE Internet Comput. **12**(5), 13–15 (2008)
2. He, Q., Yan, J., Jin, H., Yang, Y.: Quality-aware service selection for service-based systems based on iterative multi-attribute combinatorial auction. IEEE Trans. Softw. Eng. **40**, 192–215 (2014)
3. Zhao, S., Wu, G., Zhang, S.: Review of QoS research in SOA. Comput. Sci. **36**(4), 16–20 (2009)
4. Klein, A., Ishikawa, F., Honiden, S.: SanGA: a self-adaptive network-aware approach to service composition. IEEE Trans. Serv. Comput. **7**(3), 452–464 (2014)
5. Guo, D., Ren, Y., Chen, H.: A QoS constrained web service selection and ordering model. J. Shanghai Jiaotong Univ. **41**(6), 870–875 (2007)
6. Zhao, S., Zhang, Y., Yu, L., Cheng, B., Ji, Y., Chen, J.: A multidimensional resource model for dynamic resource matching in internet of things. Concurr. Comput. Pract. Exp. **27**(8), 1819–1843 (2015)
7. Li, L., Liu, N., Li, G.: A QoS-based dynamic service composition method in semantic internet of things. Appl. Res. Comput. **33**(3), 802–805 (2016)
8. Zeng, L., Benatallah, B., Ngu, A.H.H., Dumas, M., Kalagnanam, J., Chang, H.: Qos-aware middleware for web services composition. IEEE Trans. Softw. Eng. **30**(5), 311–327 (2004)
9. Cardoso, J., Sheth, A., Miller, J., Arnold, J., Kochut, K.: Quality of service for workflows and web service processes. Web Semant. Sci. Serv. Agents World Wide Web **1**(3), 281–308 (2004)
10. Jia, B., Li, W., Zhou, T.: A centralized service discovery algorithm via multi-stage semantic service matching in internet of things. In: 2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC), pp. 422–427 (2017). https://doi.org/10.1109/CSE-EUC.2017.82
11. Chen, L., Yang, J., Zhang, L.: Time based QoS modeling and prediction for web services. In: Kappel, G., Maamar, Z., Motahari-Nezhad, H.R. (eds.) ICSOC 2011. LNCS, vol. 7084, pp. 532–540. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25535-9_38