QoS Prediction for Reliable Service Composition in IoT

Gary White, Andrei Palade, Siobhán Clarke

School of Computer Science and Statistics,
Trinity College Dublin, Dublin, Ireland
{whiteg5,paladea,siobhan.clarke}@scss.tcd.ie

Abstract. An Internet of Things (IoT) environment should respond to users' requirements by providing access to a number of component services, which can be used to create applications. To meet quality of service (QoS) requirements, these applications must be able to automatically adapt to changes in the QoS of their component services. In this paper we show how matrix factorisation (MF) based collaborative filtering can be used in a self-managing, goal-driven service model for task execution in the IoT. QoS prediction enables the goal-driven model to make adaptation decisions, allowing execution paths to dynamically adapt. This reduces failures and lessens re-execution effort for failure recovery. Results based on a QoS dataset show the suitability of the proposed mechanism in IoT, where providers are mobile and QoS values of services can change.

1 Introduction

The development of Internet of Things (IoT) technology has made it possible to connect various smart objects together through a number of communication protocols. The number of connected devices is predicted to grow to between 26 and 50 billion connected devices by 2020 [1,2]. These devices will lead to a wide variety of services from multiple sources such as home applications, surveillance cameras, monitoring sensors and actuators [3]. These services could potentially be used in applications in many different domains including smart cities, home automation, industrial automation, medical aids and many others [4].

IoT applications make use of these services typically through a Service Oriented Architecture (SOA), where the services from the devices are discoverable, accessible and reusable in an IoT environment [5,6]. An application can be created by a combination of multiple services and flexible service composition, which is useful to tackle potentially complex requests. There are a number of challenges associated with the environment's openness and dynamism, such as flexible service composition for services maintained by different hosts and adaptable composites to create a new service composition when the network topology changes the QoS of services in the environment. To address these challenges, existing works in service composition have used goal-driven reasoning mechanisms

to achieve the user's complex goal, and classic AI backward-chaining to map the request to services available in the environment [7,8].

Apart from the functional requirements, a key requirement for service composition and adaptation in the IoT is to choose the correct set of services, which can also satisfy the non-functional requirements [9]. Traditional composition approaches assume that the QoS values are already known, however in reality user side QoS may vary significantly due to unpredictable communication links, mobile service providers moving out of range and the heterogeneous provider environment [10]. Collaborative filtering uses the QoS values from other users in the environment to make predictions for candidate services [11,12]. This allows for more accurate selection of the best service based on the non-functional requirements such as response time and throughput [13]. This can happen either before, or during the execution, if the QoS of the currently executing service begins to degrade. This is especially important for safety critical services such as those in healthcare, which have strict QoS requirements and where a service failing can have a serious impact.

In this paper we propose an approach for the self-management of dependable systems using collaborative filtering and goal oriented service composition [7], and present our initial results on an established QoS dataset. The remainder of the paper is organised as follows: Section 2 describes the background and related work, Section 3 presents the QoS-driven service composition and execution. Section 4 describes the experimental setup and Section 5 presents the results of the experiments. Section 6 outlines the conclusions and future work based on the results of the paper.

2 Background and Related Work

2.1 Decentralised Service Execution

Service composition is used to create complex applications based on available services provided by the devices in the environment. Dynamic service composition becomes increasingly difficult in IoT where there are a large number of available service providers, which rely on battery-powered, resource-constrained and potentially mobile devices to provide their services [14]. These devices can dynamically change their state due to poor wireless links, awake/sleep duty cycles or battery shortage [3]. In such an environment, a centralised mechanism is a single point of failure, which affects the reliability of the composition [15]. Recent works have used decentralised composition models to distribute the reconfiguration decisions across composition participants at runtime to improve the resilience and performance of the composition [7].

In previous work, we define a service composition mechanism that uses a goal-driven reasoning approach to model the capabilities of service providers in the environment and dynamically bind their offered services to an abstract composition request [8]. Apart from functional requirements and resilient execution, a service composition mechanism needs to satisfy the users' non-functional requirements. This requires QoS management to select the best set of concrete

services in a composition and to replace these services if the QoS requirements of the application are not satisfied [16]. Most QoS-aware service composition approaches assume that the QoS values of service candidates are already known and usually are provided directly by service providers or through third-party registries (e.g., UDDI) [17]. A service provider cannot give user-specific QoS as it can vary based on the user location and time of invocation [10]. Users can store their user-side QoS in the service registry and use the QoS values from other users in the environment to make predictions for unknown QoS values by collaborative filtering.

2.2 QoS Prediction

Traditional QoS prediction approaches in IoT have focused on the QoS prediction of currently executing services through time-series analysis [18,19,20]. These approaches rely on the user executing the service to generate the values, which can be used for time-series prediction. However, they make no estimates for the QoS values of candidate services, which could be switched to at runtime. This is a problem in IoT as due to the large number of candidate services it would be too time consuming to invoke even a subset of the functionally equivalent services during a runtime service composition.

An alternative approach to predicting QoS, inspired from recommender systems, is to use QoS information of similar users to make predictions about the QoS from possible services, by collaborative filtering [11,21]. These approaches use matrix factorisation to allow the user to receive QoS predictions of services that they have not invoked, based on QoS values from similar users. Using the QoS from other similar users gives more information about candidate services, which can be chosen either at design time or during runtime service execution. This also has the additional benefit that we can gather the QoS information without harming the performance of the infrastructure with needless invocations of services to retrieve QoS values.

Some existing service composition approaches use collaborative QoS prediction in a cloud environment and only consider web services [21,22]. Other mechanisms have used QoS prediction at design time to estimate changes in the QoS values at runtime [23]. We propose that these approaches can be used at the edge of the network in a heterogeneous IoT environment with services from a number of different devices.

3 QoS-Driven Service Composition & Execution

Figure 1 shows our middleware architecture, which is introduced in this section with focus on the Service Composition & Execution Engine and the QoS Monitor [8]. The main components are the Request Handler (RH), the Service Registration Engine (SRE), the Service Discovery Engine (SDE), the QoS Monitor and the Service Composition & Execution Engine (SCEE). The RH establishes a request/response communication channel with the user and forwards the request

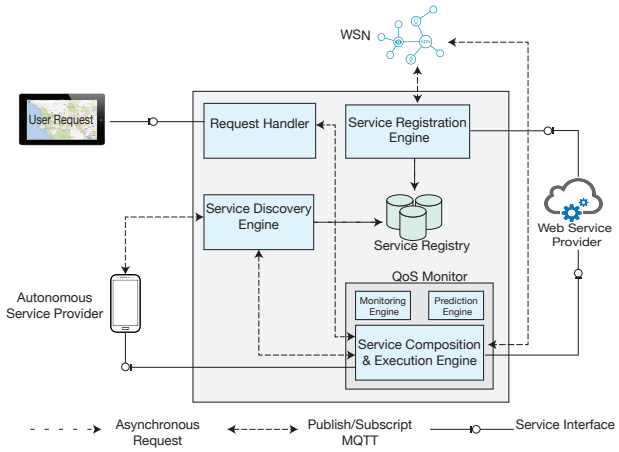


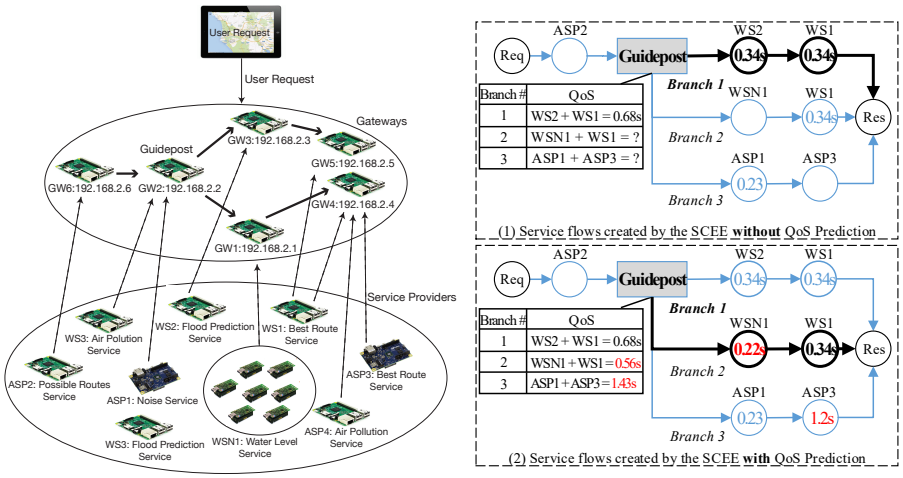
Fig. 1: Middleware Architecture

to the other middleware components. The SRE registers the available services in the environment. The SDE uses the backward-planning algorithm to identify the concrete services, which can be used to satisfy the request and sends this list of services to the SCEE. The QoS monitor is used to monitor these services and can predict possible candidate services to switch to if one of the services begins to degrade, using the prediction engine. The SCEE will use these services to create a response for the request. This process as well as how it manages the execution in a dynamic IoT environment is explained in the following section.

3.1 Service Composition & Execution

The SCEE is responsible for the composition and execution of services discovered by the SDE. Figure 2b illustrates a list of available services in the environment identified to satisfy a user request, which was received from the RH. These services are retrieved by the RH from the environment in Figure 2a, with services provided from different service types including web services (WS), wireless sensor networks (WSN), and autonomous service providers (ASP), who are independent mobile users with intermittent availability. The SCEE creates a list of service flows based on the concrete service providers received from the SDE. The flows are then merged based on the service description. If two or more services in the flow have the same input, the SCEE creates a guidepost to enable the invocation of one of these services based on QoS requirements (see Figure 2b).

An execution guidepost $G = \{R_{id},\ D\}$ is a split-choice control element of the composition process and maintains a set of execution directions D for a composition request R_{id}. These execution directions will be referred to as branches. Each element in the set D is defined $d_j = \{id,\ w,\ q\}$ where j $\leq |D|$. The set w represents the services in the branch and id represents the identifier of the branch. The value q reflects the branch's aggregated QoS values [7], which can be calculated



(a) Small Scale Demonstration

(b) Service flows selection without (1) and with (2) QoS Prediction

Fig. 2: Sample QoS Data

according to predefined formulas [24]. The branch that maximises/minimises an objective function will be selected by the guidepost during execution.

In this work, we consider the response time and throughput of each branch. The formula in Equation 1 calculates the response time by aggregating the response time value of each component service in a sequential flow [24]. In this formula, rt_i is the response time of service i. The throughput value is calculated using the formula in Equation 2, which selects the lowest throughput $min(th_i)$ offered by the services in a sequential flow [24]. These formulas require the response time and throughput values of each service component in the flow to calculate the aggregate values. It is possible that these values could not be calculated if the required QoS data is missing or is out-of-date.

$$Response\ Time\ (RT) = \sum_{i=1}^{n} rt_i \tag{1}$$

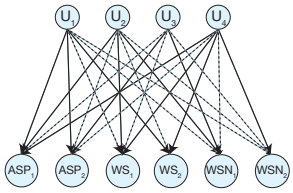
$$Throughput\ (T) = min(th_i) \tag{2}$$

To address this problem, the QoS monitor uses QoS prediction to predict QoS values across each branch stored in the guidepost. Figure 2b shows the flows created by the SCEE for User 4 (U_4 in Figure 3). The response time values recorded during service discovery phase were 0.34s for service provider WS_2, 0.34s for WS_1 and 0.23s for ASP_1. The response time values for WSN_1 and ASP_3 were not recorded. When the execution reaches Guidepost G, we can only aggregate the response time values for Branch 1 (Part 1 in Figure 2b), which is not optimal. If the composition selects the branch with the lowest reported QoS

values, it will select Branch 3, which is also not optimal. Only when we use the predicted values for the missing service QoS does the composition choose the optimal Branch 2 (Part 2 in Figure 2b).

3.2 Collaborative QoS Prediction

Accurate QoS prediction of candidate services is a fundamental component of goal driven service composition. Incorrect predictions may cause compositions and adaptations to have worse QoS, which could lead to SLA violations. On-line prediction approaches have been proposed to detect service failure and QoS deviations of the currently executing services [25,26]. Other approaches propose to collecting QoS values by invoking the candidate services, but this is not scalable in IoT due to the large amount of candidate services [27].



(a) Invocation Graph for IoT services

	ASP1	ASP2	WS1	WS2	WSN1	WSN2
U1	0.9	0.23		0.12		
U2	0.43		0.23		0.33	
U3		0.30			0.37	
U4	0.23		0.34	0.34		0.24

(b) User-service Matrix

	ASP1	ASP2	WS1	WS2	WSN1	WSN2
U1	0.9	0.23	0.25	0.12	0.22	0.3
U2	0.43	0.23	0.23	0.25	0.33	0.36
U3	0.80	0.30	0.38	0.36	0.37	0.15
U4	0.23	0.23	0.34	0.34	0.22	0.24

(c) Predicted User-service Matrix

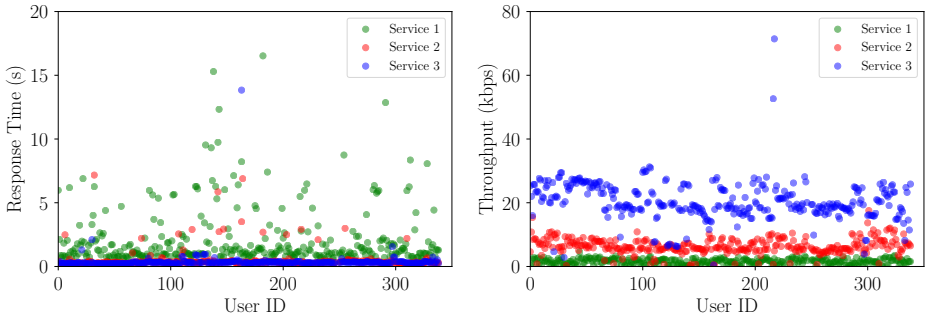
Fig. 3: Demonstration of QoS Prediction in IoT

To provide QoS values on m IoT services for n users, one needs to invoke at least $n \times m$ services, this is almost impossible in an IoT environment where we expect a large number of services and users. Without this QoS information, the service execution engine cannot select the optimal components based on their QoS and must make a choice based on whatever QoS values are available. This leads to choosing potentially non-optimal services, which can cause service degradation at runtime and service execution errors. Figure 4 shows that the same service can have different values even for the same quality factors such as response time and throughput for different users and comes from a real life dataset [10].

We use collaborative filtering, which identifies users who share similar characteristics (e.g., location, response time, etc.) to make predictions of what QoS they will receive when executing a service. The QoS value of IoT service s observed by user u can be predicted by exploring the QoS experiences from a user similar to u. A user is similar to u if they share similar characteristics, which can be extracted from their QoS experiences on different services by collaborative filtering. By sharing local QoS experience among users these approaches can predict the QoS value of a range of IoT services including ASP, web services and WSN even if the user u has never invoked the service s before [28].

We give a demonstration based on a subset of the implementation in Figure 2a, where we have a number of different service providers, who are able to provide functionally equivalent services from heterogeneous devices. We model this as a bipartite graph $G = (U \cup S, E)$, such that each edge in E connects a vertex in U to S. Let $U = \{u_1, u_2, ..., u_4\}$ be the set of component users and $S = \{ASP_1, ASP_2, ..., WSN_2\}$ denote the set of IoT services and E (solid lines) represent the set of invocations between U and S. Given a pair $(i, j), u_i \in U$ and $c_j \in S$, edge e_{ij} corresponds to the QoS value of that invocation. Given the set E the task is to predict the weight of potential invocations (broken lines).

We visualise the process of matrix factorisation for the demonstration in Figure 3b, in which each table entry shows an observed weight in Figure 3a. By using the latent factor model [29] a number of algorithms first factorise the sparse user-component matrix and then use $V^T H$ to approximate the original matrix, where the low-dimensional matrix V denotes the user latent feature space and the low-dimensional matrix H represents the low-dimensional item latent feature space. The latent feature space represents the underlying structure in the data, computed from the observed features using matrix factorisation. As the matrices V and H are dense it is then possible to use a neighbourhood-based collaborative method, as shown in Figure 3c.



(a) Response Time for Users (b) Throughput for Users

Fig. 4: Sample QoS Data

4 Experimental Setup

4.1 Dataset Description

Invoking thousands of IoT services in the wild is difficult because some of the services may have limited range and may not be publicly available on the Internet. To evaluate the prediction quality of these approaches, we use a QoS dataset, which consists of a matrix of response time and throughput values for 339 users by 5,825 services [10], which would be provided by the monitoring component.

4.2 Metrics

The predictive composition process uses the predicted values generated through collaborative filtering to choose the optimal flow. The composition process without the predicted values has access to the percentage of QoS values given by the matrix density. Once the optimal flow has been selected we report the actual values based on the original data. Two metrics are considered in this work: response time and throughput. The response time for each branch is aggregated using Equation 1. The branch which has the lowest response time is used in the composition. The throughput value for each branch is aggregated using Equation 2. The branch which has the highest throughput is used in the composition.

4.3 Performance Comparison

We compare the performance of the flow generated using the predicted values from the CloudPred collaborative filtering algorithm to the flow generated using no predicted values [11]. We show the optimal service composition to compare how each of the compositions approach the optimal solution. To evaluate this, we divided the available set of 5,825 services for each user into sub-sets of services each of size 50. We used each sub-set to create branches that can be used in a guidepost. The number of services in a branch was set to 20. The value of each branch was calculated by aggregating the QoS value assigned to each service in the branch. These values were stored in the guidepost and used for branch selection during composition execution. The branch selection uses a function, which minimises the response time and maximises the throughput. This allows the branch with the smallest response time to be selected and follow the execution of the guidepost. We conducted the experiment for a number of different matrix densities from 10% to 90% to show how the results of both approaches change as they get access to more data about the users. The results are shown for 1 user and an aggregation of 100 different users.

5 Results

Figure 5a and Figure 5b illustrate the response time of service composition, with and without prediction, for 1 and 100 users. In Figure 5a we see the response time results for a number of different matrix densities for one user. In this case the prediction composition shows a large improvement compared to the composition that has no prediction. As the matrix density increases we can see that the composition without prediction gets closer to the optimal value as it has access to more QoS values. Figure 5b shows the response times for the service flows averaged over 100 users. This graph follows the same trend as one user with a larger difference in response time between the composition approaches.

Figure 5c and Figure 5d show how QoS prediction affects the throughput of the composed services. Figure 5c shows the results for one user, when the density is less than 20% the prediction approach shows a clear improvement with

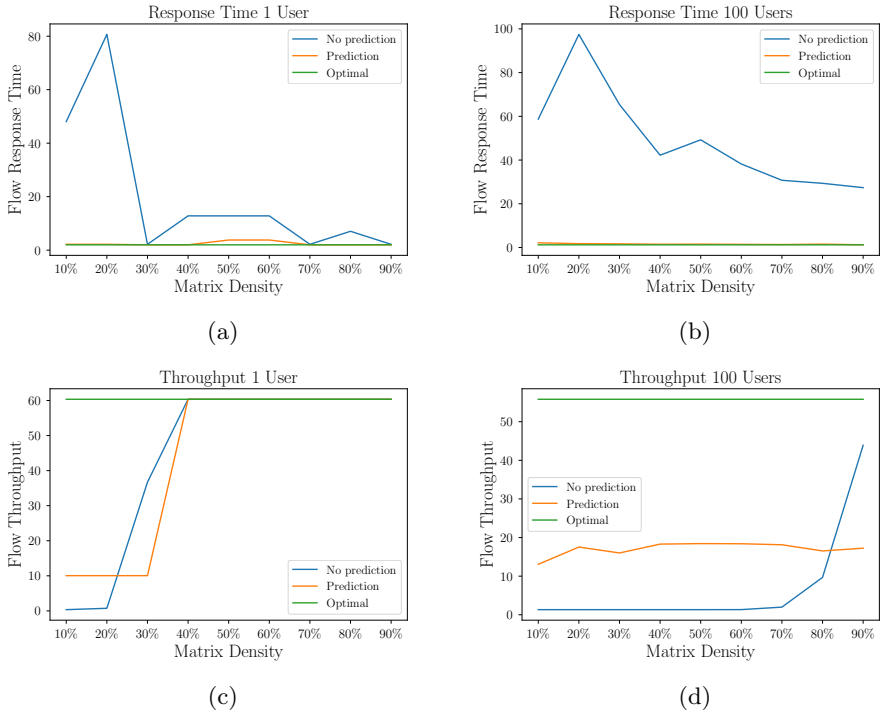


Fig. 5: Impact of prediction on service composition flows

a greater throughput value. However at 30% density, we see that composition without prediction actually performs better than the predictive approach. When the matrix density is greater than 40%, both approaches are able to find the optimal composition. Figure 5d shows the throughput values averaged over 100 users. In this case, we can see that the predictive approach has a much larger throughput value than the approach without prediction when the density is less than 80%. When the matrix density is greater than this, the composition without prediction improves and almost reaches the optimal throughput value.

5.1 Threats to validity

To assess the validity of the results, we take into account areas where bias could have been introduced, following the guidelines introduced by Peterson and Gencel [30]. In particular, we consider the interpretive validity and repeatability of the experiments.

Interpretive Validity The interpretive validity is the extent to which the conclusions from the experiments are reasonable given the data. In the results section we present the results for each of the matrix densities to allow the reader to

validate the conclusions. We show the results for an individual user as well as an aggregation of 100 users to show a comprehensive evaluation of the approaches.

Repeatability Repeatability allows for the complete repetition of the experiment to verify the results and requires detailed reporting of the research method. In the experimental setup section, we give a detailed description of the data, metrics and the approaches that were used. In Section 3 we describe the goal-driven composition algorithm and collaborative filtering algorithm, which allows for the repeatability of the experiment to validate the results. The goal-driven service composition mechanism was introduced by Chen et al. [7] and an implementation of this mechanism in an IoT scenario is presented in our previous work [8]. This should enable future proposals to reproduce our results and develop new QoS prediction models for service composition in a decentralised setting based on our source code, which is available on request. In future work we will test the prediction composition in a real life environment, which although reducing the repeatability will increase the internal validity of the results.

6 Conclusion and Future Work

The results of the experiments have a number of interesting conclusions, which can be used to outline future work. The results from the response time dataset are encouraging with predictive composition showing clear improvement over all the matrix densities. For the throughput dataset, the predictive approach showed clear improvement for low matrix densities (<= 80%) when averaged over 100 users, but for densities greater than this, the composition without prediction values produced better results. There is a clear difference in the response time and throughput results, with the response time for the predictive composition approach selecting close to the optimal flow for most densities. When averaged over 100 users, the throughput results for the predictive composition do not get closer to the optimal value as the density increases. This can be attributed to the different data scales, which can introduce larger errors for the throughput predictions [28], that can result in selecting the wrong branch.

In IoT, because of the large number of services available, we would expect that only a small percentage would have relevant, up-to-date QoS information in the time period required (e.g., within 15 mins). This makes the results for low matrix densities from 10%-30% particularly important, as they illustrate that we need only a small number of users in the environment to report values to achieve almost optimal results.

As future work, we will evaluate the overhead introduced by the QoS prediction mechanism when dynamically composing IoT services in a real-world environment. We also aim to investigate a number of alternative techniques for preprocessing the data and conducting similarity comparison between the users and the different types of service providers (e.g. data smoothing, clustering, PCA, etc.).

Acknowledgments

This work was funded by Science Foundation Ireland (SFI) under grant 13/IA/1885.

References

1. H. Bauer, M. Patel, and J. Veira, "The internet of things: Sizing up the opportunity," *McKinsey*, 2014.
2. D. Evans, "The internet of things: How the next evolution of the internet is changing everything," *CISCO white paper*, vol. 1, p. 14, 2011.
3. A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE Communications Surveys Tutorials*, vol. 17, no. 4, Fourthquarter 2015.
4. P. Bellavista, G. Cardone, A. Corradi, and L. Foschini, "Convergence of manet and wsn in iot urban scenarios," *Sensors Journal, IEEE*, vol. 13, no. 10, 2013.
5. N. Ibrahim and F. L. Mouel, "A survey on service composition ware in pervasive environments," *arXiv preprint arXiv:0909.2183*, 2009.
6. V. Raychoudhury, J. Cao, M. Kumar, and D. Zhang, "Middleware for pervasive computing: A survey," *Pervasive and Mobile Computing*, vol. 9, no. 2, 2013, special Section: Mobile Interactions with the Real World.
7. N. Chen, N. Cardozo, and S. Clarke, "Goal-driven service composition in mobile and pervasive computing," *IEEE Transactions on Services Computing*, vol. PP, no. 99, pp. 1–1, 2016.
8. C. Cabrera, F. Li, V. Nallur, A. Palade, M. Razzaque, G. White, and S. Clarke, "Implementing heterogeneous, autonomous, and resilient services in iot: An experience report," in *A World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2017 IEEE 18th International Symposium on*. IEEE, 2017.
9. A. Metzger, C. H. Chi, Y. Engel, and A. Marconi, "Research challenges on online service quality prediction for proactive adaptation," in *2012 First International Workshop on European Software Services and Systems Research - Results and Challenges (S-Cube)*, June 2012, pp. 51–57.
10. Z. Zheng, Y. Zhang, and M. R. Lyu, "Investigating qos of real-world web services," *IEEE Transactions on Services Computing*, vol. 7, no. 1, pp. 32–39, Jan 2014.
11. Y. Zhang, Z. Zheng, and M. R. Lyu, "Exploring latent features for memory-based qos prediction in cloud computing," in *2011 IEEE 30th International Symposium on Reliable Distributed Systems*, Oct 2011, pp. 1–10.
12. J. Zhu, P. He, Z. Zheng, and M. R. Lyu, "Online qos prediction for runtime service adaptation via adaptive matrix factorization," *IEEE Transactions on Parallel and Distributed Systems*, vol. PP, no. 99, pp. 1–1, 2017.
13. G. White, V. Nallur, and S. Clarke, "Quality of service approaches in iot: A systematic mapping," *Journal of Systems and Software*, vol. 132, pp. 186 – 203, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S016412121730105X
14. A. Palade, C. Cabrera, G. White, M. Razzaque, and S. Clarke, "Middleware for internet of things: A quantitative evaluation in small scale," in *A World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2017 IEEE 18th International Symposium on*. IEEE, 2017.
15. J. Bronsted, K. M. Hansen, and M. Ingstrup, "Service composition issues in pervasive computing," *IEEE Pervasive Computing*, vol. 9, no. 1, 2010.

16. R. Calinescu, L. Grunske, M. Kwiatkowska, R. Mirandola, and G. Tamburrelli, "Dynamic QoS management and optimization in service-based systems," *IEEE Transactions on Software Engineering*, vol. 37, no. 3, pp. 387–409, 2011.
17. Z. Zheng, H. Ma, M. R. Lyu, and I. King, "Collaborative web service qos prediction via neighborhood integrated matrix factorization," *IEEE Transactions on Services Computing*, vol. 6, no. 3, pp. 289–299, July 2013.
18. Z. Ye, S. Mistry, A. Bouguettaya, and H. Dong, "Long-term qos-aware cloud service composition using multivariate time series analysis," *IEEE Transactions on Services Computing*, vol. 9, no. 3, pp. 382–393, May 2016.
19. A. Amin, L. Grunske, and A. Colman, "An automated approach to forecasting qos attributes based on linear and non-linear time series modeling," in *2012 Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering*, Sept 2012, pp. 130–139.
20. A. Amin, A. Colman, and L. Grunske, "An approach to forecasting qos attributes of web services based on arima and garch models," in *2012 IEEE 19th International Conference on Web Services*, June 2012, pp. 74–81.
21. W. Lo, J. Yin, S. Deng, Y. Li, and Z. Wu, "An extended matrix factorization approach for qos prediction in service selection," in *2012 IEEE Ninth International Conference on Services Computing*, June 2012, pp. 162–169.
22. Z. Zheng and M. R. Lyu, "Collaborative reliability prediction of service-oriented systems," in *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 1*. ACM, 2010, pp. 35–44.
23. M. Li, J. Huai, and H. Guo, "An adaptive web services selection method based on the qos prediction mechanism," in *Web Intelligence and Intelligent Agent Technologies, 2009. WI-IAT'09. IEEE/WIC/ACM International Joint Conferences on*, vol. 1. IEEE, 2009, pp. 395–402.
24. N. B. Mabrouk, S. Beauche, E. Kuznetsova, N. Georgantas, and V. Issarny, "Qos-aware service composition in dynamic service oriented environments," in *ACM/IFIP/USENIX International Conference on Distributed Systems Platforms and Open Distributed Processing*. Springer, 2009, pp. 123–142.
25. C. Wang and J. L. Pazat, "A two-phase online prediction approach for accurate and timely adaptation decision," in *2012 IEEE Ninth International Conference on Services Computing*, June 2012, pp. 218–225.
26. D. Geebelen *et al.*, "Qos prediction for web service compositions using kernel-based quantile estimation with online adaptation of the constant offset," *Information Sciences*, vol. 268, pp. 397 – 424, 2014, new Sensing and Processing Technologies for Hand-based Biometrics Authentication.
27. B. Jiang, W. K. Chan, Z. Zhang, and T. H. Tse, "Where to adapt dynamic service compositions," in *Proceedings of the 18th International Conference on World Wide Web*, ser. WWW '09. New York, NY, USA: ACM, 2009, pp. 1123–1124.
28. G. White, A. Palade, C. Cabrera, and S. Clarke, "Quantitative evaluation of qos prediction in iot," in *3rd International Workshop on Recent Advances in the Dependability Assessment of Complex Systems*, June 2017.
29. R. Salakhutdinov and A. Mnih, "Probabilistic matrix factorization." in *Nips*, vol. 1, no. 1, 2007, pp. 2–1.
30. K. Petersen and C. Gencel, "Worldviews, research methods, and their relationship to validity in empirical software engineering research," in *Software Measurement and the 2013 Eighth International Conference on Software Process and Product Measurement (IWSM-MENSURA)*, Oct 2013, pp. 81–89.