SARL: A reinforcement learning based QoS-aware IoT service discovery model

Selahattin Kosunäl1, Kubilay Demir2

The IoT environment includes the enormous amount of atomic services with dynamic QoS compared with traditional web services. In such an environment, in the service composition process, discovering a requested service meeting the required QoS is a difficult task. In this work, to address this issue, we propose a peer-to-peer-based service discovery model, which looks for the information about services meeting the requested QoS and functionality on an overlay constructed with users of services versus service nodes, with probably constrained resources. However, employing a plain discovery algorithm on the overlay network such as flooding, or k-random walk could cause high message overhead or delay. This necessitates an intelligent and adaptive discovery algorithm, which adapts itself based on users’ previous queries and the results. To fill this gap, the proposed service discovery approach is equipped with a reinforcement learning-based algorithm, named SARL. The reinforcement learning-based algorithm enables SARL to significantly reduce delay and message overhead in the service discovery process by ranking neighboring nodes based on users’ service request preferences and the service query results.

The proposed model is implemented on the OMNet simulation platform. The simulation results demonstrate that SARL remarkably outperforms the existing approaches in terms of message overhead, reliability, timeliness, and energy usage efficiency.

K e y w o r d s: internet of things, services discovery, reinforcement learning

1 Introduction

A typical IoT application targets at handling the complex user requests through a set of different atomic services provided by IoT entities [7]. To create such a complex application with success, service-oriented computing (SoC) architecture has an advantage of a fully flexible framework that enables efficient usage of the services. A key advantage of SOC is its high speed, scalability and cost-efficiency when developing and deploying complex IoT applications [8]. However, this makes the discovery and selection of appropriate services a vital task for a particular task. Moreover, a huge number of IoT devices composing diverse atomic services are naturally deployed in harsh environments with limited energy sources and wireless connectivity. This dynamic nature results in highly time-varying quality of service (QoS) for atomic services providing similar functionality. In terms of QoS, therefore, satisfying user requirements is a challenging issue. This requires exploring QoS values of the services in near real-time. On the other hand, the lifetime of the IoT devices in terms of energy capacity is also prolonged while satisfying the requirements of the users.

The existing services discovery and selection approaches is divided into two distinct categories, namely centralized and decentralized methods. A centralized unit collects QoS values through aggressive probing in the centralized methods [9-11]. However, this type of strategy suffers from single point failure of the centralized unit. In addition, the aggressive probing faces a huge amount of message overhead. Another drawback is the insufficient measurements [12, 13]. On the other hand, the decentralized approaches broadcast queries to explore the required service information via structured or unstructured overlay networks [14-16]. A balanced data distribution is achieved by the structured P2P discovery methods [14, 15, 17]. Also, these methods provide an efficient query routing mechanism with an intelligent organization of the topology, considering constraints for the data distribution and exploiting distributed hashing tables (DHT). DHT-based approaches have the potential to increase the scalability of service discovery, but they may not provide the availability of all published service information. This is however causing several problems for dynamicity of the overlay network [17, 18]. The approaches running on unstructured overlay networks are designed by the themes of constrained-flooding and broadcasting-based search algorithms for the service discovery. The main problems with these approaches are the massive amount of message overhead and inefficient usage of energy sources when considering resource constrained IoT devices [14].

To address these problems, a novel services discovery approach with the theme of reinforcement learning, called SARL, is proposed for IoT environments. An underlying feature of SARL is its mechanism to discover the QoS information of the requested services on an unstructured overlay network constructed with all users rather than
directly contacting with service nodes with constrained resources.

To carefully select the neighbor users to send the queries, SARL employs a fully distributed reinforcement learning (RL) strategy to rank the neighbors with respect to their responses against the past queries. With RL approach, a user can assign a weight value for each neighbor. Therefore, SARL selects a certain number of neighbors with the highest weights to send the queries. A critical effort should be placed on the careful and accurate calculation of the weight values due to inherently time varying IoT environment. After the completion of each service query, the weight value of the associated neighbor is updated considering some QoS metrics such as the time to complete service query, the number of hop of the query and the QoS level of discovered service. These QoS level metrics in the weight calculation is used to increase the weight value more aggressively with high QoS values returned. This enables SARL to avoid selecting services that can fail with slightly decrease of QoS value during the execution. The QoS level metric in the weight calculation is used to increase the weight value more aggressively with high QoS values returned. This enables SARL to avoid selecting services that can fail with slightly decrease of QoS value during the execution.

The RL mechanism repeats the selection of the neighbor users with high weight values. However, losing one of these neighbor users is a case encountered in practice which requires to move on to the other user. The candidate neighbor users with relatively low weight values might not be selected/used for a long time, so that their up-to-date weight values are probably missing. To tackle this issue, the proposed RL strategy is continually incorporated with a process of exploration (picking different users with a certain probability at random) and exploitation (pulling the user with the highest weight value). This will give a likelihood to all neighboring users to be selected to keep up to date their weight values. Therefore, a failed request will have more impact on the weight value.

To strengthen the merits of the proposed RL strategy against the dynamic nature of IoT, we investigate the \( \epsilon \)-greedy strategy which ensures a robust level of adaptation. Win-or-Learn-Fast (WOLF) strategy is integrated into the RL strategy to avoid a user to spend much time to quit a currently unsuitable neighbor user despite many successive successful requests. Therefore, a failed request will have more impact on the weight value.

Our approach is implemented on a realistic environment using OMNeT simulation tool [20]. The simulation results validate the superior of the proposed approach comparing with the existing studies.

In this paper we provide the literature review, including recent services selection approaches, some widely used RL-based applications and existing resource discovery mechanisms. Further, details of service and network models, and underlying features and basic operation of the proposed strategy are described. Performance evaluations in comparison to well-known algorithms are presented to indicate the advantages of the proposed idea.

2 Related work

This part intends to present an overview of state-of-the-art studies. To depict the background of the work more precisely, we divide this section into three distinct fields. We first focus on the existing services selection algorithms specifically proposed for IoT environments. An emphasis is then placed on the typical RL-based applications to reveal the importance of the RL strategy. Several query-based resource-discovery methods are also introduced.

2.1 Service selection approaches for IoT

This section provides an overview of service selection approaches specifically proposed for IoT applications. Many studies have been proposed to handle a complex task of the users through an intelligent strategy of exploring services satisfying the user requirements, which are thoroughly studied in recent surveys [21, 22]. Our focus is placed on the recent state-of-the-art service selection methods with QoS awareness.

A recent study, called EQSA, focuses on the energy-efficiency while providing the desired QoS level [9]. The main mechanism of EQSA is built on conserving the residual energy with a high degree of services availability. The initial step of the EQSA is to explore the all services that provide the user-defined required QoS level, which approaches the service selection issue as a multi-objective optimization problem. In the next stage, an elimination process is applied to reduce the number of the services preselected in the first step. To do this, a threshold value is applied to eliminate the services, that is calculated with the best quality value reduced by the user preference. The services passing the threshold value are finally compared to extract the best services. For this comparison, an utility value is assigned to each service to assure a relative dominance with considerations of energy level, QoS values and user’s preferences. Another work relies on the similar methodology comprising of pre-sorting, filtering and final-sorting [23]. An underlying feature of this study is
that it allows users to select a priority level for each QoS attribute. A recent paper treats the energy consumption issue as a separate part of the problem, distinguishing it from QoS attributes [24]. Therefore, the problem of energy consumption and providing the intended functionality of QoS are handled separately and an optimization model is presented to model the problem. The problem is then solved efficiently using a pulse strategy [25].

2.2 RL-based applications

Reinforcement Learning (RL) has been recently applied to a wide range of applications in the scope of wireless communications and networking. A RL-based medium access control (MAC) protocol for wireless sensor networks (WSNs) is proposed to enhance the QoS performance in terms of channel throughput, end-to-end latency, and energy-efficiency [26]. It basically dedicates a transmission slot for each node removing the problem of packet transmission collisions. Each node learns independently the behavior of its neighbors which results in a perfect scheduling among all nodes. In underwater networking, a similar strategy is applied to handle key drawbacks specifically for underwater acoustic sensor networks [27]. It designs an efficient operation to address the problems encountered by time synchronization. A significant performance improvement in channel utilization and delay is achieved. A critical duty of cellular systems is to efficiently manage spectrum to maintain an acceptable QoS level among users in particularly voice calls and data transmissions. Due to inherent dynamic nature of such a cellular system, an intelligent RL-based approach is proposed to develop a dynamic spectrum access improving the performance of cognitive cellular systems [28].

Another application area improved with RL-based techniques is the cognitive radio. Similarly, efficient allocation of the spectrum among the cognitive radio users in dynamic environments is investigated in [29].

2.3 Resource discovery

In unstructured P2P networks, an intelligent way of exploring the pertinent resources is required to satisfy the user query. Taking the dynamic nature of the network into consideration, smart strategies would potentially provide flexible and robust mechanism against the network dynamicity. Several traditional methods have been exploited to find the required resource. Flooding allows each peer to unconditionally broadcast a request to its neighbors until a certain resource is discovered [30]. It is well-known that flooding-based methods ensure the fastest exploration with a simple structure at the expense of high number of redundant query packets. Several variations to pure flooding have been proposed to improve the performance. To reduce the high message overhead, controlled flooding is proposed to permit peers to select a specific number of neighbors at random, instead of forwarding query to all neighbors as in pure flooding [31]. A $k$-walker random walk is an improvement of controlled flooding in which peer creating the query selects $k$ neighbors randomly [32]. This is then followed by forwarding the query to a random neighbor continuously until the desired purpose is achieved or Time-To-Live (TTL) is exceeded.

The random selection of neighbors is obviously inefficient, so that an intelligent strategy is inevitable. Basically, recent studies have attempted to rank the neighbors based on previous experience. The objective is to select $k$ neighbors which probably provide the desired resource. In [33], a simple RL-based strategy is proposed to find $k$ relevant neighbors with a basic reward mechanism. It assigns a reward value of 1 for a successful query and 0 for a failed query. In this way, the cumulative reward for a particular neighbor is aimed to be maximized. Therefore, the neighbors with the high reward values will be eventually selected for a query. Another efficient way of exploring the required resource is the utilisation of a basic Ant Colony Optimization (ACO) which allows to find the associated resource with the shortest time [34]. It essentially let the ants to drop pheromone of the successful way. This will route the other ants to follow the previous trail instead of selecting a route at random. By treating the peers as ants, ACO-based solutions have a potential to provide effective resource discovery mechanisms. To further improve the performance of pure ACO in terms of load balancing among the peers, an extended and improved version of ACO, called inverted ACO (IACO), is presented in [35]. Learning-automata, which is a simple decision-making strategy in highly dynamic environments, is utilised as a distributed resource discovery algorithm which provides a recursively accelerated structure of the resource discovery for queries with a multi-attribute range [36]. It significantly reduces the average number of peers required to be visited to locate the requested resource with a high level of successfully returned queries, employing an exceptionally low control message overhead.

3 System model

We will introduce the IoT architecture and models underlying the proposed service discovery method for the IoT environment.

3.1 IoT architecture with service management

The layered architecture of IoT basically relies on the layers of perception, network, and application [37]. Due to ever increasing number of IoT devices with a high level of heterogeneity, new abstraction layers are required to be incorporated into the IoT architecture depending on the application requirements. A new layer is essentially employed for service-oriented applications, called service management layer [38]. The overall architecture is given in Fig. 1. The overall mechanism provides a flexible and efficient operation for successful completion of complex IoT applications. The perception layer is responsible for collecting and pre-processing information required.
through a set of physical sensors. The information in digital form is then transferred to the upper layer. The network layer transmits the collected data, which is so-called atomic services, to the service management layer using an available communication technology.

IoT applications may need to acquire a wide spectrum of the services from the resource constrained IoT devices. To do this, a list of the atomic services for a IoT application is created in association with their properties. Those atomic services are provided by selected services from among all atomic services. This structure enables to create complex application by composing the atomic services, namely composite services, as an atomic service can carry insufficient information/functionality individually. The composite services are linked with the requesting users via application layer. It may be noted that service management layer permits to also create cloud-based services thanks to historical and analytical data extracted from the atomic services.

3.2 Network model of SARL

Due to many available atomic services in a practical IoT environment, each user could have a limited number of services’ information. Therefore, users should be able to share their services information with each other. In this way, all users have the possibility to access the information about requested services meeting QoS requirements.

To do this, after initiating a service request, the user starts to search for another user holding the required service. This drives us to build an overlay network of users including a set of diverse users, each of which maintains the information of the services invoked previously. We consider that each user is assigned to a certain number of neighbors at random. Discovering the required services with satisfied QoS requirements through the neighboring user nodes is the focus of this study.

We consider that the overlay network consists of user nodes but not service nodes. User nodes are mostly mobile phone or web users. However, a service node also become a user by requesting a service from another atomic service. However, for simplicity, we exclude this satiation in our implementation.

On the other hand, we consider that an atomic service node could be a single IoT device or, as in fog and edge computing, a modem/router, which collects data from IoT devices and fulfils a pre-processing. However, in this work we do not focus on services types and level but handling their heterogeneity and dynamicity to provide a smooth service provision. Our network model is depicted in Fig. 2.

4 SARL: Services discovery mechanism with reinforcement learning algorithm

SARL consists of two mechanisms: (1) The main P2P-based discovery mechanism, and (2) The RL-based algorithm, which significantly improves the discovery mechanism.

4.1 P2P-based service discovery mechanism

SARL’s main goal is to discover atomic services meeting QoS requirements defined by the requesting IoT application. To do this, SARL constructs an overlay with user nodes, which runs as middleware on the users. To discover the information about requested atomic services, SARL simply send k query to its k neighbor user nodes, and they forward those queries if they do not contain the information. This forwarding continues until either finding the information (HIT) or expiring time-to-live (TTL) duration, as depicted in Fig. 2. To discover the services, this mechanism is simple but powerful and resource efficient especially comparing with the existing approaches discovering the information directly contacting with service nodes with constrained resources. Moreover, by using
the RL-based algorithm (see the following section), this discovery mechanism is considerably empowered.

4.2 Reinforcement learning-based algorithm

4.2.1 The proposed RL-based algorithm

To improve the efficiency of service discovery mechanism in this paper, a weight-driven and linear type of RL strategy is applied to maximize the value function through a pre-defined reward and punishment. The key idea relies on intelligent selection of the neighbor users based on the previous experience of the users in replacement of the classical random selection. For this purpose, a user maintains a weight table assigning a weight value for each service of its neighbor users. The weight value indicates the success level of a user to be selected as a next hop. The weight values are initially assigned to 0 on start-up. Upon completion of a request, the associated weight value of the selected neighboring user for the requested service is updated with the following recently proposed, model-free and stateless linear RL algorithm [29]

$$W_{i,j}^{t+1}(S_m) = W_{i,j}^t(S_m) + \gamma R,$$

where $W_{i,j}^{t+1}$ represents the weight value from user $i$ to $j$ for the $m^{th}$ abstract service at time $t + 1$. $R$ indicates the current reward function returning two typical values, namely reward and punishment. In this study, a successful request will take a reward of 0.1 and a failed request will take a punishment of -0.1. There is no consensus for the selection of values of $R$ which makes it an application-dependent property. A series of successful requests through the same neighbor user will essentially result in the increment of the related weight value. This will therefore increase the probability of the associated user to be selected as one of the preferred neighbor users.

Due to the high dynamic in IoT environments, each query should not have the same effect on the weight value. Therefore, after a successful completion of a query, the level of learning should be carefully arranged according to the dynamics of IoT environment. For instance, a query completed with a high latency will have a low increment on weight value. For this purpose, we introduce a new coefficient called learning efficiency, $\gamma$, to determine to what extent the recently acquired information will impact on the learning speed. To better understand the idea of $\gamma$, it may be considered the learning rate, $\alpha$, in Q-Learning. It is typically set to a small constant value, which controls the speed of weight (Q-value) update. High values of both $\gamma$ and $\alpha$ increases the weight more aggressively. The main difference between the concepts of $\gamma$ and $\alpha$ is that the value of $\gamma$ is a non-constant value which is targeted to reflect the IoT dynamics on the weight update. Its value is dynamically calculated depending on the carried information of each query.

To derive the learning efficiency precisely, critical features of the information obtained will be considered. This study introduces three fundamental dynamics of IoT environments, which are likely to be increased depending on the application requirements, i.e., number of hops, query latency, and the QoS values of the service found. Firstly, number of hops: a request is forwarded hop-by-hop until it finds the required QoS values or TTL is terminated. It is important to take into consideration the number of hops a request visited. Secondly, latency: the time taken from generation to completion of a request is another key feature. The first two features are also significant to ensure a good level of energy efficiency. Thirdly, QoS values: several candidate services up to $k$ is discovered at different extents in terms of QoS. For instance, a candidate service may ensure the highest service quality for all QoS attributes except for one attribute providing lowest quality. In this case, the overall performance of this service may be easily corrupted with a small performance reduction on the lowest-quality attribute during the execution of the service. This necessitates to update the weight value in association with the obtained QoS values of the candidate services. Finally, the learning efficiency is the combination of the three features

$$\gamma = \begin{cases} e^{-(\psi-1)/\alpha_1} e^{\phi/(\psi-\sigma-1)/\alpha_2} \xi & \text{for reward} \\ 1 & \text{for punishment} \end{cases}$$

where $\psi$ represents the number of hop for a service request, $\sigma$ is the time taken to explore the requested service, $\phi$ is the pre-defined maximum limit of time for finding the service and $\xi$ is the current status of QoS values of the explored service. The aim of usage exponential constant $e$ in the equation 2 is to provide an exponential increase while closing to the "best" values. The alpha coefficients also enable tuning of influence of the individual components such as the number of hops, $\Psi$, and the latency, $\sigma$.

In order to reflect the influence of the QoS values of the $k$ services returned to each corresponding neighbor user, we simply look at the volume of the QoS values by introducing a new minimum limit (ML) value which is the exceeding of the user-defined QoS threshold with certain rate. For instance, assuming that the user-defined threshold is 50 and the value of exceeding threshold is 20%, the ML will be 60. In this respect, $\xi$ will take 1 if all QoS values are greater than the ML, and 0.9 otherwise. This is to decrease the weight increment when one or more QoS value is sufficient, but at a critical level, which may corrupt the overall performance for the next service request. The value exceeding threshold can be chosen depending on the properties of the QoS attributes, such as it can be set to a very small value for reliability which is typically expected to have a high threshold 99%. 

S. Kosunalp, K. Demir: SARL: A REINFORCEMENT LEARNING BASED QoS-AWARE IoT SERVICE DISCOVERY MODEL
4.2.2 Convergence, exploration and exploitation

A drawback of the RL strategy in a case of losing a successful neighbor user is the high amount of successive failures required to reduce the weight value to enable the user to explore a new neighbor user. It may be noted that the value of $\gamma$ is equal to 1 for a failed request. This is to make the proposed RL scheme compatible with Win-or-Learn-Fast (WOLF) principle [28]. WOLF allows a user to learn faster when a request fails, and more slowly for a successful request. In this way, it will avoid the user to spend a lot of time to give up an unsuitable neighbor user after a high number of consecutive successful requests. This idea ensures a good level of robustness to protect the system from infrequently becoming unavailable services. Consequently, the whole process described until now is the nature of a plain RL-based system. The target point is to provide a quick convergence, whereby some neighbor users are accepted as converged when their weight values reach a unique threshold value. After convergence, the system can adapt to environmental and network changes with the basic functionality of the RL mechanism.

A common disadvantage of RL algorithms is the imbalance between exploration and exploitation. A RL agent is expected to explore its environment by obtaining enough knowledge through repetition of previous actions. Exploitation is performed to capitalise on the experience already obtained through exploration. Each agent should achieve a certain level of exploration which triggers exploitation stage. Therefore, an intelligent mechanism should be essentially investigated to control the trade-off between exploration and exploitation. A prominent method, called $\epsilon$-greedy strategy, performs the exploration and exploitation simultaneously, since the action with the highest weight value is always preferred resulting in the updating its weight value after the completion of the action. This type of strategy, after convergence, stabilises on the actions with high weight values for a long time of period, giving no attention to other actions. This is however not an efficient way in highly dynamic conditions such as an IoT environment. The proposed RL algorithm employs an extended version of greedy strategy known as $\epsilon$-greedy with a fixed value (0-1) of $\epsilon$. The basic idea is to let users to select their neighbor users with the highest weight value with probability $1-\epsilon$ allocating the probability $\epsilon$ for random selection. With $\epsilon$-greedy policy, the exploration duration can be controlled by choosing an appropriate $\epsilon$ value, which permits a sufficient level of exploration after convergence. The choice of $\epsilon$ value depends on the desired exploration level and it is typically set to 0.03 in all experiments of this work to allow exploration for 3% of the time after convergence.

In our scenario, we accept a neighbor user as converged if its weight value approaches to 1. After convergence, the weight value is traditionally only updated in exploration. Hence, the maximum limit of the weight value is 1. Considering the dynamic structure of the IoT, the availability of services is highly time-varying which drives us to add a new property to the whole process of the proposed RL approach. In exploitation stage of a converged user, we propose to update the weight value only for punishment. This is to reduce the weight value more aggressively if the availability of the service is losing. Note that the speed of the weight value reduction depends only on the punishment value which can be arranged by the functionality of specific application environments. By doing so, the proposed strategy generates a quick convergence to the ideal operation of the network while providing a reasonable response to environmental changes.

5 Evaluation

In this work, we propose a service discovery mechanism (SARL) for the IoT environment, which runs on users of the services. Since both user and service nodes have constrained resources, such as limited battery and computation capacity, to provide a realistic assessment, we should evaluate the proposed systems in terms of the following metrics, associated with the constrained resources: message overhead, latency, reliability and network life time.

5.1 Evaluation metrics

- Imposed additional message overhead may cause network congestion or energy shortage (each additional wireless transmission leads to a decrease in the battery level). Since message overhead should be minimized in such a service discovery system running on user nodes, mostly mobile phones.
- The time to discover the requested service is of great importance, as up-to-dateness of the discovered QoS information of the service depends on the discovery time of services. We therefore compare the proposed approach with the existing works concerning the latency.
- In our evaluation, reliability refers to the rate of successful query, which returns the information of service meeting the QoS requirements.
- Finally, we evaluate the proposed system regarding the imposed energy consumption during the service discoveries.

5.2 Implementation

To implement the proposed approach, we exploit Omnet++ [44]. In our implementation, unless otherwise stated, 200 user and service nodes are deployed. Each user, 24 times in a day, sends queries to look for three different services composing an IoT application. Each simulation is run for around a year. We also repeat each simulation run until obtaining stable results. Transmission latency/data-rate of each link between two nodes is generated in a random manner between 20-100 ms/1-10 Mbps, respectively.

On the other hand, the QoS attributes defined by the IoT application for the services are abstracted into three type of QoS attributes: (1) positive quantitative
attributes, (2) negative quantitative attributes, and (3) qualitative attributes (Boolean 1 or 0). We select a representative attribute from each type to decrease the search complexity. The queries sent by users contain the requested minimum values for the three selected QoS attributes. When SARL found “k” services with different QoS, the services with QoS values farthest from the minimum QoS values are chosen.

Since creating a large-scale IoT deployment is a difficult task, and there is no QoS dataset for IoT environments and services in the literature, we syntactically generate the QoS values of the IoT services based on values reported in the literature [9, 45]. As representatives of the three type QoS attributes, we select availability (positive), latency (negative), and SSL (Secure Sockets Layer) certificate checking (Boolean). While availability values are uniformly produced between [0.80 - 0.9999], latency values are uniformly generated in the interval [0.30 - 0.50] after being normalized. We decide whether the SSL certificate exists in a service node in an arbitrary manner demonstrating with 1 (exist) or 0 (not). To produce a realistic fluctuation in the QoS values of the services, we multiply them by a number between [0.95 - 1.05] every day. On the other hand, as described in [9], battery level of each device are randomly generated in the interval [0.7 - 1.0] in a normalized form. As our proposal focus on mostly service discovery, in our evaluation, we compare our approach with the existing approaches in terms of their energy consumption during the service discovery. Therefore, for hopping a query over each node, a constant amount of energy is declined from the corresponding user node or service node (for the centralized approaches, probing the service nodes is considered as service discovery).

5.3 Assessment of effects of the mechanisms on the network

We compared the proposed mechanism with the following existing mechanisms: (1) Central Approaches: This approach is common in the literature, and relies on periodic probing of the services by a server to obtain QoS information or exploits just historic QoS values of the services [9, 10, 46, 47]. Each user queries the server containing QoS information of all services to ask a service covering the required functionality and QoS. (2) Constrained Flooding: While flooding approach simply broadcast the query to entire network to find the resource, constrained flooding approaches restrict the spreading of the broadcast with adding Time-to-Live (TTL) value [48, 49]. In our implementation we limited number of neighbors for broadcasting for each node on the network to restrain an aggressive message overhead increase in the network. (3) K-Random-Walk (K-RW): Random walk (RW) is another resource discovery method. RW leads to much lower message overhead versus the flooding approaches but much higher latency. K-RW approach decreases the high latency in the RW by rising the number of walkers (k walker).

Firstly, we compare SARL with the three-existing works regarding message overhead. We build three simulation setups with 100, 200, 300 service and user nodes

![Fig. 3. Message overhead comparison with the existing works](image1)

![Fig. 4. Latency comparison of the four approaches](image2)

![Fig. 5. The number of hops for access to the QoS information of the requested services](image3)
with 10-15 neighbors to demonstrate effect of increase of the number of nodes. Fig. 3 shows that SARL discovers the requested services information with much less message overhead in comparison to constrained flooding and central approaches. Furthermore, While SARL causes a reasonable increment in the message overhead while climbing the number of service and user nodes, those two approaches result in sharp rises. On the other hand, to provide a fair comparison, for both SARL and K-RW, $k$ is 3 and TTL is 500 ms. Fig. 3 denotes that SARL also outperforms K-RW in each different number of nodes.

Secondly, we test SARL’s latency to reach the QoS information of the requested services. The latency results are also compared with the existing works. Fig. 4 demonstrates that SARL remarkably outperform K-RW and constrained flooding thanks to RL-based routing algorithm, which weights the lower latency links to access the QoS information of the services, even in a dynamic environment. Moreover, we clearly see that although centralized approaches are able to access the information in one hop due to their server-client model, there is relatively small difference between two approaches latency.

The number of hops for access to the QoS information of the services affects both latency and reliability of the service discovery in addition to the message complexity. We compare SARL with two P2P approaches i.e K-RW and constrained flooding. Since central approaches employs server-client model, which leads to one hop, this approach is excluded from the comparison. Fig. 5 denotes that while SARL discovers the service information around 3 hops, K-RW and constrained flooding can find the information around $> 7$ hops and $> 4$ hops, respectively. Since SARL’s RL-based algorithm considers the number of hops according to equation 2, SARL can discover the QoS information in fewer hops compared to the other approaches.

Since most of the user nodes are mobile and battery constrained phones, energy consumption of the approaches is also an important metric in the assessment. In the simulation setup, we consider that each user node has the same amount of energy. During the each forwarding of a message, total energy consumption for the computation and transmission is the same in each node for all the compared approaches. Fig. 6 shows normalized energy consumption of each approach for a simulation run. For centralized approach, we build two simulation setup with probing per 30 s and 60 s to denote effect of probing on the energy consumption. We clearly see in Fig. 6 that SARL outperforms all the approaches by causing minimum energy consumption. Notably, constrained flooding approach could deplete whole energy of the network, which makes this approach impracticable for the service discovery. Furthermore, energy consumption of the central approach subjects to probing frequency. With increasing probing frequency, ie, from 60 s to 30 s, we see that energy consumption significantly rises. Usage of central approaches is also not practicable or not preferred method when we consider its high message overhead on service nodes with constrained resources.

In the light of the above mentioned, we continue our tests on the two approaches outstanding with their performances, ie, SARL and K-RW. Since there is no guarantee to access the QoS information of the requested services in the fist time query using these remaining methods, we test reliability of these methods. The reliability refers to successful query rate in the first time. Fig. 7 demonstrates that while SARL provides high reliability for all k values (ie, $k = 2, 3, 6, 10$), reliability results of K-RW are far below what is required. The reason of outstanding reliability performance of SARL is that each agent is (nodes) learning of places of the wanted information does not allow the service discovery mechanism for a random action leading to walk through undesired places and expire the TTL duration. In the simulation setup of all above tests, $k$ is 3, as SARL provides the required reliability in $k = 3$.

Moreover, simply increasing the number of the query number “$k$” would be seen a solution to increase of the reliability of the K-RW approach. To clarify this issue, we fulfill a simulation to compare two approaches’ message overhead increase while rising k value from 2 to 10. Figure 8 shows that although SARL maintains its
outperforming for the message overhead, both two approaches leads to considerable message overhead increase when arising k values from 2 to 10. Since SARL provides the required reliability and latency with k = 3, which causes reasonable message overhead, highly increasing k to enhance reliability of K-RW does not make K-RW a preferable method.

![Fig. 8. Message overhead comparisons with varying k values](image)

5.4 Convergence analysis

SARL defined the convergence in Section 4.2.3 for a particular neighbour user if its weight value converges to a value close to 1. The RL-based approaches are traditionally expected to provide a full convergence in which each user has a certain number (k in our case) of converged neighbour users. The convergence of a particular user is lost if the weight value of one of its neighbours starts to decrease which is likely to occur in dynamic IoT environments. In our implementation, to observe the convergence lost, we alter the QoS value of users daily as explained in Section 5.2. In case of losing one or more converged neighbour users, exploring a new neighbour user to be converged is a critical part of this study. It is worth mentioning that the $\epsilon$-greedy strategy of SARL allocates a small number of queries, controlled by $\epsilon$ value, for non-converged neighbour users. The weight values of these users are also kept relatively up to date. Therefore, SARL spends few queries for converging the weight values of such users to 1. In the performance evaluations of SARL, we observe the time taken in terms of query amount to explore new neighbour users as converged. The average number of queries to find a new converged neighbour user is observed as 10, meaning that SARL mechanism provides a quick convergence recovery.

6 Discussion

Our comprehensive tests show that the approach proposed in this paper to provide an efficient service discovery for the IoT environment remarkably outperforms three existing approaches the K-RW, constrained flooding, and central approaches. Particularly, the simulation results denote that employing constrained flooding and central approach for the service discovery for the IoT environment with constrained nodes is not practicable due to their massive amount of message overhead and energy consumption. K-RW could be a rival against SARL, but its low reliability rates to discover the QoS information of the requested services makes it not preferable method. Thanks to RL-based services discovery algorithm, SARL discovers the requested services with high reliability rate by causing minimal message overhead, latency, and hops.

7 Conclusion

This paper concentrated on the development of a novel resource-efficient services discovery strategy for IoT applications, namely SARL. It permits the exploration of the QoS conditions of service nodes to find services requested by users. The basic operation of SARL lies in a P2P-based resource discovery mechanism, improved by employing a fully distributed reinforcement-learning algorithm. SARL searches the requested services by sending queries to other users by building a user overlay network rather than directly querying atomic services, with constrained resources. Each user maintains a weight table mapping each neighboring user and service type through the RL strategy. The weight table is updated based on the results of service queries that essentially indicate the availability and quality level of the services. Eventually, upon a service request, SARL running on the hosting user selects a specific number of neighboring users with the high weight values to send the queries. Furthermore, to avoid users to spend a lot of time to give up an unsuitable neighbor user with high weight, we equip SARL with Win-or-Learn-Fast (WOLF) principle, which enable users to learn faster when a request fails, and more slowly for a successful request. To fix the imbalance between exploration and exploitation, we also employ $\epsilon$-greedy with a fixed value (0-1) of $\epsilon$, letting the user to select the neighbor user with probability 1-$\epsilon$ which permits a sufficient level of exploration after convergence. Performance evaluations through a realistic simulation environment clearly prove the efficiency of the proposed strategy in comparison to well-known existing approaches.

REFERENCES

[1] L. Atzori, A. Iera and G. Morabito, “The internet of Things: a survey”, Computer Networks, 54(15), pp. 2787-2805, 2010.
[2] S. Li, L. D. Xu and S. Zhao, “The internet of things: a survey”, Information Systems Frontiers, 17(2), pp. 243-259, 2015.
[3] P. Kamalinejad, C. Mahapatra, Z. Sheng, S. Mirabbasi, V. C. Victor and Y. L. Guan, “Wireless energy harvesting for the Internet of Things”, IEEE Communications Magazine, 53(6), pp. 102-108, 2015.
[4] G. Kecskemeti, G. Casale, D. N. Jha, J. Lyon and R. Ranjan, “Modelling and simulation challenges in internet of things”, IEEE Cloud Computing, 4(1), pp. 62-69, 2017.
Selahattin Kosunalp received the BSc degree in electronics and telecommunications engineering from Kocaeli University, Kocaeli, Turkey, in 2009, and the MSc degree in communications engineering and the PhD degree in electronics engineering from the University of York, York, UK, in 2011 and 2015, respectively. He is currently with the Department of Computer Technologies, Bandirma Onyedi Eylul University, Turkey. He is an author of several refereed journal and conference papers and has experience as a Reviewer for several conferences and journals. His research interest lies in wireless sensor networks, medium access control protocol design, energy harvesting technology, internet of things (IoT) and reinforcement learning.

Kubilay Demir received his MSc in Electronics and Telecommunication Engineering in 2010 from University of Kocaeli, Turkey. He obtained PhD at TU Darmstadt, Germany, in the Dependable, Embedded Systems & Software Group. He is currently an assistant professor at Bitlis Eren University. His main research interests include the areas of dependable and secure communication, smart grid, IoT, Reinforcement Learning and Autonomic Systems.