Journal of Cleaner Production 212 (2019) 1345–1356

Contents lists available at ScienceDirect

Journal of Cleaner Production

journal homepage: www.elsevier.com/locate/jclepro

# LISA: Lightweight context-aware IoT service architecture

Sarada Prasad Gochhayat [a], Pallavi Kaliyar [a], Mauro Conti [a], Prayag Tiwari [b], V.B.S. Prasath [c], Deepak Gupta [d, *], Ashish Khanna [d]

[a] *Department of Mathematics, University of Padova, Italy*
[b] *Department of Information Engineering, University of Padova, Italy*
[c] *Department of Electrical Engineering & Computer Science, University of Cincinnati, OH, USA*
[d] *Maharaja Agrasen Institute of Technology, Delhi, India*

A R T I C L E I N F O

*Article history:*
Received 13 October 2018
Received in revised form
6 December 2018
Accepted 9 December 2018
Available online 13 December 2018

*Keywords:*
Internet of things
Context-aware
Push-based services
Web services
Automatic query generator

A B S T R A C T

Internet-of-Things (IoT) promises to provide services to the end users by connecting physical things around them through Internet. The conventional services build for web are primarily based on the pull technology, where the user actively engages with system to get the services. However, in IoT environment, the services are based on push-based, where information and value added services will be pushed towards the user. Unless, these push-based services are properly managed they would overwhelm the user with unnecessary information, thus, it will soon start annoying the user.

In this paper, we propose a lightweight context-aware IoT service architecture namely *LISA* to support IoT push services in an efficient manner. In particular, LISA filter and forward the most important and relevant services to the users by understanding their context. To achieve its goals, LISA formulates a user model to resolve local decision making by using agents and available web services paradigm. The proposed user model describes the user in an abstract way by considering the context and profile information of the user. For evaluation, we simulate LISA by considering an IoT tourist guide system as a use case scenario, and we show the performance of the our user model concerning precision and recall metrics. The results of our preliminary experiments confirm that LISA successfully reduces the information provided to the user by selecting only the most relevant among those. The evaluation shows that LISA can extract services for a user by selecting from 15000 services with precision upto 0.3 and recall upto 0.8, and it can be further optimized by tuning the user-specific design settings. Additionally, our approach shows improvements in query processing time which also includes the query generation time.

## 1. Introduction

Internet-of-Things (IoT) paradigm blurs the boundaries between physical objects and computational intensive devices by connecting them through Internet (Aksu et al. (2018); Conti et al. (2017)). It promises to provide user-centric services by considering both, the user context and the user profile information (Kim et al. (2009); Recker et al. (2003)). The existing IoT services have some limitations such as: execution of the tasks within the user devices that has limited computational power and memory (Chiu and Leung (2005)), the information provided to the user might be rigid (Kenteris et al. (2009)), lack of consideration of the time, device, or network constraints of the user (Hashemian and Mavaddat (2005)), and the most important, the user is overwhelmed by the information as not all the services or data pushed towards her is of interest. Thus, the techniques of service creation and dissemination needs to consider the above mentioned limitations as well as demand of new system design (Douzis et al. (2018)), i.e., shift from conventional pull based web services to push-based web services.

The conventional web services are built on pull-based applications, where the user formulates a particular query to get the services. Here, a query is a set of the most relevant terms for a service. The problem with this approach is that the user explicitly involves with the system to get the service. However, in IoT domain, the services will be push-based where service providers will push the service towards the user. For example, in IoT advertisement (Aksu

* Corresponding author.
*E-mail addresses:* sarada1987@gmail.com (S.P. Gochhayat), pallavi@math.unipd.it (P. Kaliyar), conti@math.unipd.it (M. Conti), prayag.tiwari@dei.unipd.it (P. Tiwari), support@elsevier.com (V.B.S. Prasath), deepakgupta@mait.ac.in (D. Gupta), ashishkhanna@mait.ac.in (A. Khanna).

https://doi.org/10.1016/j.jclepro.2018.12.096
0959-6526/© 2018 Elsevier Ltd. All rights reserved.

et al. (2018)), the service provider will send notifications about different advertisements toward the user. The existing push-based systems like SMTP-based email, in which the sender pushes the email to the SMTP email server which the receiver downloads (Duan et al. (2005)), are inadequate in IoT domain and are computationally heavy. Moreover, as there will be thousands of IoT services around the user, these services will create information burden to the user and it will also consume network resources. Hence, there is a need to design a lightweight service architecture for IoT environment (Gudla and Bose (2016); Bormann et al. (2012)) which considers the above mentioned requirements of IoT services.

In an IoT environment, the applications provide personalized and adaptive services to end users. Due to the huge set of devices involved, and therefore, its pervasiveness, IoT is a great platform to leverage for building new applications and services or extending the existing ones. In the past, the web services paradigm is used as it provides the advantages of building distributed applications. Hence, it helps to select and provide various IoT services to the users as per their context. As the number of web services increases exponentially, selecting and providing the appropriate service among the vast number of required services become very difficult. Hence, to search the most relevant service, a correct query needs to be formulated.

### 1.1. Contribution

In this paper, we propose LISA, a lightweight context-aware IoT service architecture for supporting IoT push services in an efficient manner. The main role of LISA is to reduce the communication overhead between user domain and service domain (refer to Fig. 1). First, LISA analyzes: (i) the user context and generate a query on behalf of the user, and (ii) the set of services that are pushed towards the user. Then it selects the most relevant services that need to be forwarded to the user. In the beginning, it formulates a user model, then it uses agents to select the relevant services locally, and subsequently, it exploits the advantages of web services. While existing systems selects the relevant services for the user centrally at the cloud (Yu et al. (2018)), our approach uses agents, which act independently to provide personalized services (Germanakos et al. (2005)) and to perform the tasks distributively. The agents generate the query, fetch the information, and use the formulated user model to adapt the content to provide the required information in the best suitable way. In case of a failure of one agent (mobile or static), another agent is deployed to look after the user needs. The proposed architecture is based on the fact that a significant set of problems can be solved efficiently, if these are addressed locally. By doing so, the architecture provides features such as faster and efficient local decision making and fault tolerance, short service

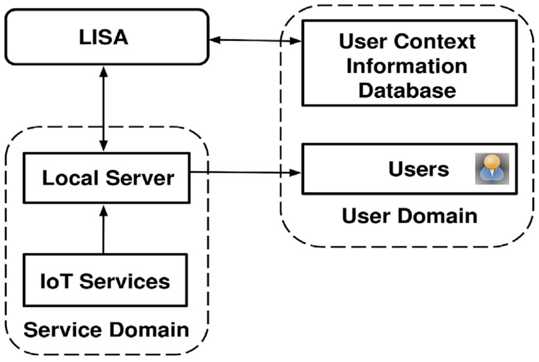

**Fig. 1.** Role of LISA in IoT environment.

selection time, and dynamic integration of new services. To this end, the major contributions of our work are as follow.

- We present the design of an IoT service architecture, which enables the important IoT services to reach the user based on his/her context information. We show that the different modules in our architecture (such as Query Initiator, Automatic Query Generator, Web Service Provider, and Web Service Adaptation) efficiently fulfill the essential requirements of IoT service provisioning that includes, usability, accountability, and adaptability.
- We demonstrate the importance of distributed agents to locally solve the overload problem of IoT push based services. As a result of our distributed approach, we achieve improvements in both, the response time and the bandwidth utilization, the two parameters which are crucial to provide IoT services effectively. In particular, LISA considers the energy limitations and bandwidth usages in the IoT network by delegating its functionality to the local agents.
- Finally, we tested our proposed architecture on a simulated IoT tourist guide system use case scenario to show its effectiveness. The simulation results obtained show the feasibility and correctness of our proposal.

### 1.2. Organization

The rest of the paper is organized as follow. Sections 2 and 3 present the related work and the terminologies used in our proposal. In sections 4 and 5, we present our proposed web service architecture and the IoT user query generation model along with the details of their functionality and the interaction mechanisms between different components of the architecture. The simulation setup and performance evaluation are discussed in Section 6. Finally, conclusion and directions for future work are given in Section 7.

## 2. Related works

In this section, first we present evolution in state-of-the-art for push based applications in web domain and automatic query generation techniques in web services. Then, we specify the need for an efficient lightweight IoT service architecture.

Most of the existing push based approaches have been developed for web browsers and generally require an explicit involvement of the user. In (Gudla and Bose (2016); Gudla et al. (2016)), the authors propose a smart push system with feedback enabled flow control. In the proposed system, a gateway client is installed on the user device which is integrated to the browser app. The gateway client observes the feedback from the user, and the user behavior from the feedback is considered to develop the push system. In (Cho et al. (2016)), the authors propose a feedback concept to maximize the service utilization by preventing the system from delivering unnecessary messages to the users. Thus, it improves the average transmission rate for the required notification messages. Authors in (Duan et al. (2005)) emphasize the fundamental implications of push and pull techniques and suggest the use of receiver-pull model to curb the unwanted Internet traffic. Based on their suggestion, our work uses receiver-pull model for wisely selecting the push-based services. In (Karagiannis et al. (2015)), authors discuss and compare different application layer protocols, which are needed to handle communications by assuming that all the end-devices make their data available to the Internet by sending the information to a web-server or cloud. This raises several privacy implications like *Does the cloud need to know if the SmartKey is* *connected to the SmartVehicle?*, which can be addressed by keeping the private and sensitive data in the edge rather than sending it to the centralized cloud (Kim and Lee (2017)).

So far the problem of automatic query generation in IoT environment has not been addressed in detail, although, a few research papers discuss about the automatic generation in web applications like patent search. For instance, an automatic query generation for patent search has been discussed in (Xue and Croft (2009)) to transfer a query patent to a search query by combining different search features with machine learning techniques. A query taxonomy generation is discussed in (Chuang and Chien (2003)), which organizes the user queries in a hierarchical structure, and it helps the web applications to retrieve the information. Authors in (Singh et al. (2017)) made an attempt on exploring the performance of relevance feedback using individual query expansion term selection approaches, and later it also uses several rank aggregation approach to aggregate the query expansion terms, which improved the performance of the system.

Early works on content delivery in IoT environment using agents is discussed in (O'Hare and O'Grady (2003); Gochhayat and Pallapa (2015)). In few proposals, the context-information has been considered to improve the query retrieval performance. In (Shen and Zhai (2003)), the authors have considered the user query history as the context information to revise the query and to improve the retrieval performance of the current query. The context information, the previous queries, and the URLs clicked by the user, has been used to classify the queries in (Cao et al. (2009)). The location information, which is considered as the essential context parameter is generally used to provide location-based services. The authors in (Shankar et al. (2009)) designed a scheme, which uses the location information to query the location-based services while maintaining the privacy. To select appropriate services and to provide personalized web services the user profile is considered in (Balke and Wagner (2003)). The scheme expands the user's service requests by considering user-specific demands and wishes. User context has been considered to discover suitable services (Doulkeridis et al. (2006); Sheng and Benatallah (2005)) and to provide context-aware web services (Chen et al. (2006); Yang et al. (2008)) in the continuously changing environment. To take the limitations of the above mentioned works into account and to advance the state-of-the-art, the agents in our proposed architecture generate queries automatically. It is done by considering the three important axes of IoT ecosystem, which includes context, profile, and history information. The use of these axes will improve the user quality-of-experience as well as the resource utilization in the resource constrained IoT networks.

## 3. Terminologies

In this section, we discuss a set of primary terminologies that are used in our work. The proposed work is based on three fundamental technologies: (i) IoT services, which provide different types of services to users in a distributed environment; (ii) IoT user model, which represents the user in an abstract manner; and (iii) Agents, which manage and adapt the content based on user model (please refer to Fig. 2). We first discuss about IoT applications and their requirement in 3.1, then the IoT user query in 3.2, and subsequently we discuss IoT user model, web services, and agent technologies in sections 3.3, 3.4, and 3.5, respectively.

### 3.1. IoT application

An IoT application provides comprehensive, coherent, and personalized services to the end user. It is done by collecting user related information from distributed information repository, and by adapting and providing the information—visual, audible—based on user-specific requirement and preferences (Bormann et al. (2012)). To develop IoT applications, one has to consider the following essential requirements.

- Usability: In IoT, most of the applications would push their services towards the user, and these services should be handled properly along with the ability of the user to control them.
- Controllability: The user should have the power to control over when and what services or notifications it receives by defining its own interest before the application content is pushed towards her (Duan et al. (2005)). The user controllability of the IoT devices would keep the user secured in the case of an emergency or life-threatening situation.
- Accountability: The user should quickly verify that the promised services are being provided by the service provider (Zhou et al. (2017)), i.e., the provider should be accountable for the services which it provides in order to avoid malicious behavior.
- Resource-constraints compliant: The user devices in IoT environment will have resource constraint, so they might not be able to do all the operations all the time. Hence, some part of the

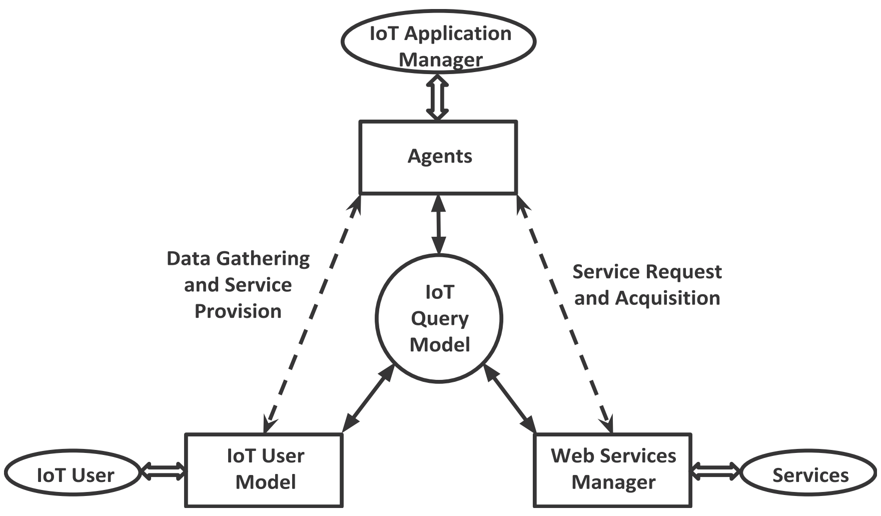

**Fig. 2.** Three important technologies required for IoT service.

operations should be delegated to a trusted resource-rich close-by devices. The tools like agent technology and edge computing provides better solutions in this context (Li et al. (2017); Kim and Lee (2017); Halder et al. (2018)) and the same should be explored further.

- Adaptability: In order to dynamically invoke the web services which can cater the user's need, an automatic query formulation method is needed. It should have the ability to dynamically generate the query on behalf of the user by considering the user's current context, profile information, and the past queries made by the other users.

### 3.2. IoT user query

In IoT environment, the applications should formulate a query to get the services to support the user activities. The IoT user query is a query which is generated by the IoT system on behalf of the user to get the relevant services in a given context. It consists of the most probable query terms at a given location and time, and the context and personal information of the user. While, to avail web services, the existing mechanisms demand the users to specify their requirements, therefore, the IoT user query model should automatically generate a query to reflect the user requirements for the services. For our proposal, the format of IoT user query is given in 5.3, and the IoT query will be formed based on IoT user model.

### 3.3. IoT user model

A user model is a collection of user information. It is essential in IoT environment that the system generates the IoT user query and adapts to the services. The IoT user model represents the user and the environment in an abstract way. The process of building, updating, and modifying a user's model, i.e., user modeling, is very challenging because of the dynamic environment and different users' requirement (Nurmi et al. (2006)).

A user model includes multidimensional user information such as spatial-temporal characteristics (time, location, and other context information), interpretation (understanding level of the user, i.e., the way user perceive), and preferences (user-centric requirements). The IoT user model can have the following information.

- *Personal information:* The personal information consists of: 1) user interest and preferences (academic, professional or personal), 2) user understanding or knowledge level, 3) user goal (short-term or long-term), 4) access rights to the information (security aspect), and 5) time constraints, i.e., the time in which user has to go through the information (less or more time which can be derived from the calendar or event lists) (Nurmi et al. (2006)). Additionally, a user's personal information can be stationary (e.g., name, gender, and blood group) or dynamic (e.g., knowledge, age, educational qualification, and financial status).
- *Context* information*:* The use of contextual information can improve the system effectiveness and user experience. Context characterizes the current environment of the user which includes physical, system, application, and social aspects. These aspects will help the system to understand the user and provide the desired services in different situations. The physical context includes information such as location, time, temperature, and pressure, while the system context includes device type, and device capacity, and the application context includes application type (e.g., health care, business, tourist, education) and social application includes information about social events, and the status of the user in the group (Melucci et al. (2012); Melucci (2005); Melucci and White (2007)).

### 3.4. Web services

Web services are self-contained, distributed, modular, and dynamic information exchange systems (Kosuga et al. (2002); Hashemian and Mavaddat (2005)). Web services use a standardized XML messaging system to make itself available over the Internet (Staab et al. (2003); Roy and Ramanujan (2001)). Thus, these services improve the interoperability and extensibility to exchange data over various computer networks (Roy and Ramanujan (2001)). Hence, the personalized services that are based on the user model can be provided using the web services by selecting and locating the most relevant service (Lueg (1998)) to reduce user's information burden (Carvalho et al. (2010)). The Web services use XML to describe a standardized way of integrating web-based applications. Web services are built around the collection of open protocols and standards such as TCP/IP, HTTP, Java, HTML, and XML to provide interoperability and for exchanging data between applications or systems. All the standard web services works using the following components: 1) SOAP (Simple Object Access Protocol), it is used to transfer the data; 2) UDDI (Universal Description, Discovery and Integration), it is used for listing the available services; and 3) WSDL (Web Services Description Language), it is used for describing the available services.

### 3.5. Agents

Agents are autonomous programs that execute applications on a device on behalf of the user's IoT environment (Chen (2013)). The essential features of agents are autonomy and adaptability. The mobile agents are those who are not bound to the device where they were created, and they travel from device to device by resuming their execution (Manate et al. (2013)). The agents are used in a distributed computing environment for their inherent capabilities, i.e., asynchronous autonomous interaction, robustness and fault tolerance, and support for heterogeneous systems (Sim (2012); Fortino et al. (2014)). The agent based systems solve the scalability problem faced by the centralized systems. The functionality or the task assigned to an agent, which executes the task in a distributed and autonomous fashion, can be easily monitored and modified by the manager by updating the software code of the agent. Faults can be easily detected using mobile agents by analyzing the devices connected to a network.

The agents run on an “Agent Platform” which offers execution, communication, mobility, tracking, directory, persistence, and security services (Xu et al. (2013)). The execution environment and the mobility service allows agents to run their code and enables an agent to move within different execution environments. The mobile agents are secured, as the mobile agent platform provides encryption, code integrity for transmission and execution, authentication, and trust. JADE and Voyager are popularly used mobile agent platforms, which provide the above-mentioned services and use SSL connection to assure data flow encryption and to detect any attempt to tamper with an agent. Agents play an important role in highly distributed environment like IoT ecosystem (Rho et al. (2013)).

## 4. LISA: proposed IoT service architecture

In this section, we present our proposed IoT service architecture called LISA, and we describe its working methodology in detail. The objective of the LISA is to filter the most relevant services based on user's context and profile information. To achieve the objective following functionalities are provided by the LISA: 1) it generates a query automatically using the past queries made by the users and the IoT user model; 2) it selects the most relevant services from the set of available services; and 3) it adapts the content based on *IoT* user model, whenever required. The overview of our proposed IoT service architecture along with its major components and their interactions with each other is shown in Fig. 3.

We now present the functioning of the service architecture by describing its different component or modules.

- **Query Initiator Module (QIM)**: This module triggers the Automatic Query Generator (AQG) based on the change in the user context information, to form a query on behalf of the user. The context value changes when any of the following events happen: 1) *Location change* - based on the proximity or the closeness of the user and the object of interest; 2) *Time change* - events which happens during a particular time of the day, month, or year such as lunch time, festival time, and season of the year; 3) *Social change* - the role played by the user in a society or group changes as happens in a meeting; and 4) *User requirement change* - the user requirement depends on the dynamic mood or emotional state of the user. Although location change is taken into account while designing system in recent years, but all the other above mention "changes" needs to be considered to initiate the query generator. The change of the context information can be given priorities based on the application, such that for a particular application the system initiates the query generator but for others it does not. For example, the "location change" information will be most useful for applications like tourist guide systems and IoT museum, while it might not be important during IoT health-care system (where the patient's health condition is more important than its location). Section 5.1 explains the functionality of QIM in detail.
- **Automatic Query Generator (AQG)**: It forms the query based on the context information, profile information, and the past queries generated by the users. By comparing the user profile information such as age group, salary, and educational qualification with the past users, the AQG selects the best suitable query terms for the query generation. The selection of query terms and the formulation of the query in LISA are discussed in detail in sections 5.2, and 5.3.
- **Web Service Provider (WSP)**: It specifies different kinds of services provided by various providers. It collects all the information about the web services and sends it to the web service adaptation module, which selects the services based on the context information and provides them to the users. The WSP uses UDDI (Universal Description, Discovery, and Integration) to get the list of relevant services available based on the query generated by the AQG. It uses WSDL (Web Services Description Language) for understanding the description of the relevant services, and it uses SOAP (Simple Object Access Protocol) to gather the information from the relevant services.
- **Web Service Adaptation Module (WSAM)**: It selects the services by checking the gathered data from the service providers with the IoT user model to find out the most suitable web services for the user, and it adapts the content collected from different service providers to present it to the user in the best suitable way. The services were selected based on the query terms, and the device and time constraints. For finding the similarity between the query terms and the services, LISA use Jaccard Index (Nayak and Lee (2007)). Jaccard similarity coefficient or Jaccard index is a way to compare the diversity and similarity of sample sets. Jaccard similarity is used for comparing two binary sets. The Jaccard Index between the query terms set Q and set of terms defining $i_{th}$ service ($S_i$) is defined as,

$$Sim_{Q-S_i} = \frac{T_{Q,S_i}}{T_Q + T_{S_i} - T_{Q,S_i}}, \tag{1}$$

where $T_Q$ and $T_{S_i}$ are the numbers of terms in sets $Q$ and $S_i$ respectively; $T_{Q,S_i}$ is the number of terms common in sets $Q$ and $S_i$. The services are selected based on the user device specification. The available time of the user that can be gathered from the user schedule is compared with the service response time of each service to find a suitable service. The service response time ($RT_{S_i}$) of a service $S_i$ is calculated as follow,

$$RT_{S_i} = ST_{S_i} + DT_{S_i} + WT_{S_i}, \tag{2}$$

where, $ST_{S_i}$, $DT_{S_i}$, and $WT_{S_i}$ are the service time, i.e., the time taken by service provider to execute the service (Gudla and Bose (2016)), the delay time, i.e., the time taken by service provider to receive the request and send the service, and the waiting time, i.e., the time a

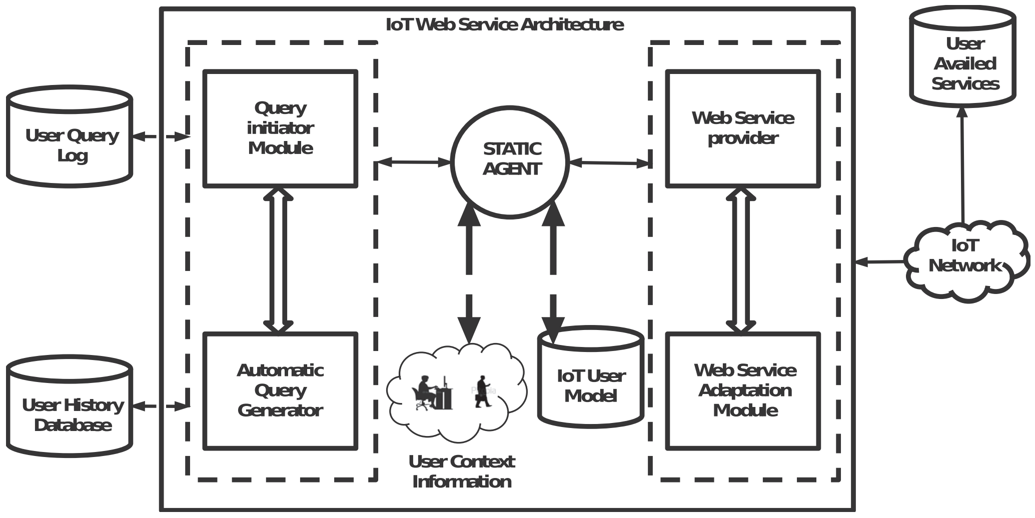

**Fig. 3.** LISA architecture.

user request has to wait before it get service from the service provider (Silver et al. (2003)). The Relative Time Difference ($RTD$) based on the available user time ($UT$) and service provider's $S_i$ time ($RT_{S_i}$) is calculated as follow,

$$RTD_{S_i} = 1 - \frac{|UT - RT_{S_i}|}{max\{UT, RT_{S_i}\}}. \tag{3}$$

The services having $Sim_{Q-S_i} + RTD_{S_i}$ values higher than the threshold value are considered for the service provisioning.

- **Static Agent (SA)**:The Static Agent (SA) administers the overall functioning of the system and makes the system work well in a distributed environment by generating and dispatching the mobile agents (MAs). In order to distribute workload, and to tackle single point failure, we consider a set of SAs, which can work together to distribute the burden among them. The functions of a static agent at the IoT services are:
  - to register the user to collect her personal information and the context information.
  - to activate the AQG module whenever it gets any information from the QIM to start or modify a service.
  - to calculate the user and applications statistical information to provides it to the IoT user model to update the user model.
  - to instruct the web service provider to fetch the web services for the user whenever required and provide this information to the web service adaptation module to adapt the content, and to provide the adapted information to the user.
  - to interact with other SAs to distribute the work.
  - to anticipate the user movement to the next subnetwork.
  - to generate and dispatch a MA to the next subnetwork, where the user is about to migrate.
- **Mobile Agent (MA)**:The MAs are created and dispatched by SAs to act on their behalf to formulate the query and do the web service adaption and provision. The functions of a MA in a subnetwork are:
  - to carry user's information with it, i.e., the IoT user model from SA.
  - to analyze the context information of the user in the subnetwork and formulate the query on behalf of SA.
  - to collect the web services and adapts the information based on the user device and context information.
  - to inform the SA about the user activities in the subnetwork.
  - to predict the user movement and alert the SA about the same, so that the SA can generate another MA (if one is not available) and dispatch it to the next subnetwork where the user might migrate.

## 5. IoT user query generation model

In this section, we discuss the process of the automatic IoT query generation in LISA. The creation of a query has the following three stages: initiation of the query, selection of query terms to form a query, and formation of IoT query.

### 5.1. IoT Query Initiator

The IoT Query Initiator Module (QIM) works in the following way. First, it collects the context information from the environment. Then it checks whether the change in the context information is relevant to initiate the IoT query model. If the quantified change is more than the threshold value, it activates the AQG to generate the query. The AQG form a query based on the current context information like location (the user's closeness to the object), time (social status change - the role played by the user in a group changes), and user requirements. The “location information” is further divided into three types: *Everywhere*, *Mobile*, and *Location specific*. The *Everywhere* indicates that the service is required everywhere, the *Mobile* means the queries need to be modified based on the user movements, and *Location specific* characterizes that the information needs to be specified based on the location. The “Time information” is also divided into three levels: *Always*, *Temporal* and *Emergency*. Here, *Always* represent the service in need to be provided all the time, *Temporal* means service needs to be presented based on the temporal information like lunch time or dinner time, and *Emergency* indicates that the services need to be provided or changed dynamically at the emergency time. The “Social Status” is divided into four tiers: *Individual*, *Group Hierarchy 3*, *Group Hierarchy 2* and *Group Hierarchy 1*. The *Individual* means the user is alone, while the *Group Hierarchy 3*, *Group Hierarchy 2* and *Group Hierarchy 1* represent the user's role in the group from the lowest to the highest order. The “User requirements” is divided into 3 categories: *Regular*, *Application Specific*, and *User Mood*. The *Regular* category represents that the user does not want any specific information, *application specific* represents that the applications demand need to be met to run the service properly, and *User mood* considers the volatile moods to provide the services. Table 1 shows the context, their types, and the rating associated with them.

**Table 1**
Different context types and their corresponding ratings.

| Context Information | Types | Rating |
|---|---|---|
| Location | Everywhere | 1 |
| (L) | Mobile | 2 |
| | Location Specific | 3 |
| Time | Always | 1 |
| (T) | Temporal | 2 |
| | Emergency | 3 |
| Social status | Individual | 1 |
| (S) | Group Hierarchy 3 | 2 |
| | Group Hierarchy 2 | 3 |
| | Group Hierarchy 1 | 4 |
| User Requirements | Regular | 1 |
| (R) | Application Specific | 2 |
| | User Mood | 3 |

The following four type of events take place because of the change of context information: Location_event (Le), Time_event (Te), Social_event (Se), and Requirement_event (Re). The Location_event value is calculated as follows.

$$Le = \begin{cases} 1 & \text{If } \dfrac{|LR_{t_2} - LR_{t_1}|}{Highest\_LR} > L_{th}, \\ 0 & \text{Otherwise}, \end{cases} \tag{4}$$

where $LR_{t_2}$ and $LR_{t_1}$ are the location rating at time $t_2$ and $t_1$, respectively, $Highest\_LR$ and $L_{th}$ (chosen by the system designer) are the highest rating of the location context type and the threshold value above which the change in location information is considered relevant. The QIM informs AQG if any one of these events occurs as it is shown in Equation (5). The algorithm of the IoT Query Initiator is shown in Algorithm 1.

$$Action = \begin{cases} Inform\ AQG & Le \oplus Te \oplus Se \oplus Re = 1 \\ No\ Action & Otherwise. \end{cases} \tag{5}$$

**Algorithm 1** Algorithm For IoT Query Initiator

```
1: Input:Context Information - Location (L), Time (T), Social (S), Requirement (R)
2: Output:Action to inform the Automatic Query generator (AQG) or not
3: Check L, T, S and R values in the table 1 and assign the Ratings (LR, TR, SR and RR)
4: if $\frac{|LR_{t_2} - LR_{t_1}|}{Highest\_LR} > L_{th}$ then
5:     Le = 1
6: else
7:     Le = 0
8: end if
9: Similarly, calculate the value of Te, Se and Re.
10: if Le ⊕ Te ⊕ Se ⊕ Re = 1 then
11:     Inform AQG
12: else
13:     No Action
14: end if
```

## 5.2. Query terms selection

In this section, we discuss the process of Query Term Selection (QTS) for the formulation of the query, which is based on the user history, context, and profile information. It is also based on the queries generated by the users in the past (Shen and Zhai (2003)) that are stored in the query log. Each query generated in the past at a particular location around a specific time shows the importance of the location and time. For example, querying about a restaurant of specific interest during lunch time (Cao et al. (2009)). As different people have different interests and desires, choosing the query best suited to the person's desire is essential. The queries generated by the users in the past are saved in the query log from which the query terms are selected to automatically create the new query. To select the query terms, the frequency of the query terms, and the weight assignment to the terms based on the user's profile information are considered.

Let $Q = \{q_1, q_2, \ldots, q_n\}$ be a set of queries generated in the past by the users at a particular instant of time. Here query $q_i$ is a set of terms, i.e., $q_i = \{t_1, t_2, \ldots\}$. Let $QTS$ be the set of all the terms in the query log that are collected at a particular instant of time, i.e., $QTS = \cup_{i=1}\{q_i\} = \{t_1, t_2, \ldots, t_m\}$. Let $tv_i$ be the weight associated with the term $t_i$ of $QTS$, and it is calculated by finding the number of queries that have the term by total number of terms,

$$tv_i(t_i) = \frac{\left|\left\{q_j : t_i \in q_j \;\&\; t_i \in Q\right\}\right|}{|QTS|}. \tag{6}$$

Let $a_u$ and $s_u$ be the age and salary information of the user $u$. The age and salary of a user can fall into different levels, and these are considered to calculate the percentage of the users belonging to a particular level who have queried in the past. Let $A$ and $S$ be the total number of levels in age and salary categories.

Let

$$D_{age\_per}(t_1) = \{dap_1, dap_2, \ldots, dap_A\}$$

$$D_{salary\_per}(t_1) = \{dsp_1, dsp_2, \ldots, dsp_S\}$$

be the sets whose elements represent the percentage of the total users belonging to a particular level queried the term $t_i$, in age and salary categories respectively. Let

$$D_{age\_wei}(t_1) = \{daw_1, daw_2, \ldots, daw_A\}$$

$$D_{salary\_wei}(t_1) = \{dsw_1, dsw_2, \ldots, dsw_S\}$$

be the sets of weight assigned to the different levels of age and salary based of $u$'s age and salary information. We use the following weighing method: 1) assign priority value 1 to the level to which the user belongs, 2) multiply $\alpha$ (where $0 < \alpha < 1$) to the previous assigned value and assign it the priority value to the immediate levels (i.e., below and above) of the user level, 3) continue the same for other levels till all the levels have been assigned with the priority value, and 4) normalize the values to get the assigned weight. For example, if a user belongs to level 3 of 5 level of user division, the priority value assign to the level 3 is 1 (see Table 2), the priority value assigned to levels 2 and 4 (the immediate levels) is $\alpha$, and the priority value of levels 1 and 5 is $\alpha^2$. Then the weight assigned for level 1 to 5 are $\alpha^2/(total\,priority)$, $\alpha/(total\,priority)$, 1, $\alpha/(total\,priority)$, $\alpha^2/(total\,priority)$, respectively (where $total\,priority = 2\alpha^2 + 2\alpha + 1$).

We assign a weight to the term $t_1$ for the $u$ based on the age and salary. The $weight_{age}(t_1) = dap_1{*}daw_1 + dap_2{*}daw_2 + \ldots + dap_A{*}daw_A$ and $weight_{salary}(t_1) = dsp_1{*}dsw_1 + dsp_2{*}dsw_2 + \ldots + dsp_S{*}dsw_S$. The query terms selected are based on $weight_{age}$, $weight_{salary}$ and the frequency of the term $t_i$. We generate Selected Query Terms (SQT) set by considering the terms whose total weight ($total_{weight}(t_i)$) (i.e., the summation of all weight values) is more than the threshold. The algorithm of the QTS is shown in Algorithm 2.

**Algorithm 2** Algorithm of Query Terms Selection

```
1: Input:$a_u$ - Age of the user $u$,
   $s_u$ - Salary of the user $u$,
   $QTS = \{t_1, t_2, ..., t_m\}$: Query Term Set, from the query log
   $D_{age\_per}(t_1)$ and $D_{salary\_per}(t_1)$ - Percentage of the users belonging to Age and Salary categories, respectively, queried the query term $t_1$
2: Output: SQT (Selected Query Terms)
3: Find the user's level based on the Age and salary
4: Assign the weights to the query term based on the user's age and salary level based on table 2:
   $D_{age\_wei}(t_i) = \{daw_1, daw_2, ..., daw_A\}$
   $D_{salary\_wei}(t_i) = \{dsw_1, dsw_2, ..., dsw_S\}$
5: Calculate term weight based on the user age and salary:
   $weight_{age}(t_i) = \sum_{j \in A} dap_j(t_i) * daw_j(t_i)$
   $weight_{salary}(t_i) = \sum_{j \in S} dsp_j(t_i) * dsw_j(t_i)$
6: Find the total weight for the term frequency term $tv_i$
   $total_{weight}(t_i) = weight_{age}(t_i) + weight_{salary}(t_i) + tv_i$
7: Select the query terms which are higher than the threshold value, i.e.,
   $SQT = \{t_i : total_{weight}(t_i) > query_{threshold}\}$
```

## 5.3. Formation of IoT query

The generic IoT user query (Q) is represented as follow.

$$UQ := \{ <T>, <UA>, <UD>, <UUH>, <UUM> \} \tag{7}$$

where $\langle T\rangle$, $<UA>$, $\langle UD\rangle$, $\langle UUH\rangle$, and $<UUM>$ are query term set, IoT application specification, IoT device specification, IoT history parameters, and IoT user model, respectively.

- Query Terms set (QTS): The QTS is represented as:

**Table 2**
Different User level and his priority at different levels.

| User Level/Weight | Level 1 | Level 2 | Level 3 | Level 4 | Level 5 | $\alpha_{total}$ |
|---|---|---|---|---|---|---|
| Level 1 | $\frac{1}{\alpha_{total}}$ | $\frac{\alpha}{\alpha_{total}}$ | $\frac{\alpha^2}{\alpha_{total}}$ | $\frac{\alpha^3}{\alpha_{total}}$ | $\frac{\alpha^4}{\alpha_{total}}$ | $1+\alpha+\alpha^2+\alpha^3+\alpha^4$ |
| Level 2 | $\frac{\alpha}{\alpha_{total}}$ | $\frac{1}{\alpha_{total}}$ | $\frac{\alpha}{\alpha_{total}}$ | $\frac{\alpha^2}{\alpha_{total}}$ | $\frac{\alpha^3}{\alpha_{total}}$ | $1+2\alpha+\alpha^2+\alpha^3$ |
| Level 3 | $\frac{\alpha^2}{\alpha_{total}}$ | $\frac{\alpha}{\alpha_{total}}$ | $\frac{1}{\alpha_{total}}$ | $\frac{\alpha}{\alpha_{total}}$ | $\frac{\alpha^2}{\alpha_{total}}$ | $1+2\alpha+2\alpha^2$ |
| Level 4 | $\frac{\alpha^3}{\alpha_{total}}$ | $\frac{\alpha^2}{\alpha_{total}}$ | $\frac{\alpha}{\alpha_{total}}$ | $\frac{1}{\alpha_{total}}$ | $\frac{\alpha}{\alpha_{total}}$ | $1+2\alpha+\alpha^2+\alpha^3$ |
| Level 5 | $\frac{\alpha^4}{\alpha_{total}}$ | $\frac{\alpha^3}{\alpha_{total}}$ | $\frac{\alpha^2}{\alpha_{total}}$ | $\frac{\alpha}{\alpha_{total}}$ | $\frac{1}{\alpha_{total}}$ | $1+\alpha+\alpha^2+\alpha^3+\alpha^4$ |

$$<T> := \{(t_1, tv_1),\ (t_2, tv_2), \ldots,\ (t_m, tv_m)\}, \tag{8}$$

where $t_1$ is the term attribute $t_1 \in SQT$, and $tv_1$ is the associated term values, i.e., the frequency of the term.

- IoT Application Specification set: The IoT Application Specification set is represented as follow.

$$<UA> := \{(ua_1, uav_1), (ua_2, uav_2), \ldots, (ua_n, uav_n)\} \tag{9}$$

where $ua_1$ is the IoT application attribute and $uav_1$ are the associated attribute values. The different attributes required for the IoT application specification are: 1) language of the content (*Application _ Language*), 2) available multimedia resources - text, audio, video (*Application_Content_Type*), 3) application type, - informative, Guide system, route information, cost, quality (*Application_Type*), and 4) application constraints or toleration limit (*Application_Constraints*) such as time and area constraints. The time constraint consists of time to visit the museum or tourist spot, average duration of visit, opening and closing time of tourist spot, restaurant, or hotel. The area constraints such as area of the museum and the transportation means available in the areas are also considered.

- IoT Device Specification set: The IoT Device Specification set is represented as follow.

$$<UD> := \{(ud_1, udv_1), (ud_2, udv_2), \ldots, (ud_p, udv_p)\} \tag{10}$$

where $ud_1$ is the IoT device attribute and $udv_1$ are the associated attribute values. The different device attributes considered are: 1) device type (*Device_Type*) such as Smart phone, PC, and laptop, 2) device screen resolution, 3) network interfaces (*Device_Network*) that the devices has, 4) device interface (*Device_Interface*), i.e., touch screen or keypad, and 5) device sensors (location, orientation, speed).

- IoT User History set: The IoT User History set is represented as follow.

$$<IUH> := \{(uh_1, uhv_1), (uh_2, uhv_2), \ldots, (uh_q, uhv_q)\} \tag{11}$$

where $uh_1$ is the IoT history attribute and $uhv_1$ are the associated attribute values. The user history which is stored in a database can be used to find different attributes such as: 1) the time spend at an exhibition/museum (*User_time_history*), 2) the type of content - audio, video, text, usually preferred or seen by the user (*Content_Type_seen*), 3) the means of travel - walking, cycling, driving, or traveling by bus, metro, train, or flight (*Means_of_Travel*), and 4) user mobility history.

- IoT User Model set: The IoT User Model set is represented as:

$$<UUM> := \{(um_1, umv_1), (um_2, umv_2), \ldots, (um_r, umv_r)\} \tag{12}$$

where $um_1$ is the IoT user model attribute and $umv_1$ are the associated attribute values. The User model consists of attribute such as: 1) the demographic data (*Demographic_data*) of the user - age, education level, interest, disabilities (e.g., blind, deaf); 2) Types of monument preferred by the user - church, historical or geographical spot; 3) Travel Preference (*Travel_Preference*) - by walk, bus, train, car; and 4) the user location (*User_Location*).

Two examples of the IoT user queries are given below:

- Query 1: History of the Old buildings in City X

$$\{T\} := \{\text{"}Hisotry\text{"}, \text{"}old\text{"}, \text{"}building\text{"}, \text{"}City\ X\text{"}\}$$

$$\{UA\} := \{Type\_of\_Info = \text{"}Monument\ info\text{"}, Language\_Info = \text{"}English\text{"}, Opening\_Time = \text{"}10am\text{"}, Closing\_Time = \text{"}5pm\text{"}, \}$$

$$\{UD\} := \{Device\_Type = \text{"}Smart\ Phone\text{"}, Device\_Network = \text{"}4G\text{"}, Device\_Interface = \text{"}Touch\ Screen\text{"}\}$$

$$\{UUH\} := \{User\_Time\_History = \text{"}1\ hour\text{"}, Content\_Type\_Seen = \text{"}text,\ audio\text{"}, Means\_Of\_Travel = \text{"}rental\ car\text{"}\}$$

$$\{UUM\} := \{Location = \text{"}Home\text{"}, Location\_Coordinate = \text{"}x, y\text{"}, Demographic\_age = \text{"}34\text{"}, Demographic\_edu = \text{"}Graduate\ in\ History\text{"}, Prefer\_Type\_of\_building = \text{"}Historical\text{"}\}$$

- Query 2: Climate of City Y

$$\{T\} := \{\text{"}Climate\text{"}, \text{"}City\ Y\text{"}\}$$

$$\{UA\} := \{Type\_of\_Info = \text{“Weather info”}, Language\_Info = \text{“Chineese”}, Opening\_Time = \text{“5am”}, Closing\_Time = \text{“1pm − 2pm, 8pm”}, \}$$

$$\{UD\} := \{Device\_Type = \text{“Mobile Phone”}, Device\_Network = \text{“3G”}, Device\_Resolution = \text{“480∗800 pixel”}\}$$

$$\{UUH\} := \{User\_Time\_History = \text{“30 minutes”}, Content\_Type\_Seen = \text{“text, image”}, Means\_Of\_Travel = \text{“walk”}\}$$

$$\{UUM\} := \{Location = \text{“Home”}, Location\_Coordinate = \text{“x, y”}, Demographic\_age = \text{“52”}, Demographic\_edu = \text{“no”}, \}$$

## 6. Simulation and results

In this section, we describe the simulation environment which has been used to test our proposed architecture. We considered an IoT tourist guide system as a use case scenario and evaluated the obtained results. Fig. 4 shows the target simulation environment.

### 6.1. Evaluation setup

To provide the relevant services efficiently to the end users, the most important metrics are precision and recall, thus we focus our evaluation process around these two metrics. The precision used in our measurements is defined as the relevant services among the retrieved services, while recall is the fraction of relevant services that have been retrieved from the existing relevant services. We also consider the processing time to generate the query and to fetch the information as the evaluation metric. We carried out the simulations on Windows Operating System of version 1709 (OS Build 16299.19) on Intel(R) Core(TM)i7-7700K CPU @ 4.20 GHz with 32 GB RAM. The evaluation programs was written in Matlab 2017b. In this simulation, we have considered 12 different types for queries, which includes history of the location, climate, transportation means (bus, train, air), tourist spots, and educational institutes. We have used five types of IoT services that includes IoT Health-care, transportation, learning, museum, and tourist services, and we considered 15000 service providers for providing information about different kinds of services. Finally, for each evaluation, confidence interval level (i.e., 98%) is determined but not included in this paper due to their lower significance values.

An IoT tourist application, which automatically generates the queries on behalf of the user and fetches information from service providers has been activated. For query formation and service provision along with context information like time and location, we have also considered three profile information (age, education level, and economical status). The users are classified into four types based on their age and salary. Similarly, the users are classified based on their educational qualification as well, i.e., basic education, school children, college students, and professionals. As the users move in different areas, their context information is collected and stored in a database. The context and profile information of the user and query logs are used to generate the current query automatically.

As there was no dataset which could give us the query logs required for our simulation work, we have synthesized the query logs for the automatic query generation and web service selection for the users. The synthesized data includes a number of different types of queries (for history and climate) generated for particular area at different time instances.

Fig. 5 shows the different type of queries generated by the users in the past in a particular area at different time instances. The *x* axis shows the time (in second), and the *y* axis shows the number of type of queries generated, respectively. Fig. 6 shows the number of queries generated by the users in the past about the history and climate at a particular area at different time instances. As it can be seen from the figures that different number and types of queries are being generated at each time instance. This data will be used as synthesized data for our experimentation purposes.

### 6.2. Results analysis

In our use case, we considered the user as a college student with moderate income, who starts moving in the area and requires the tourist information.

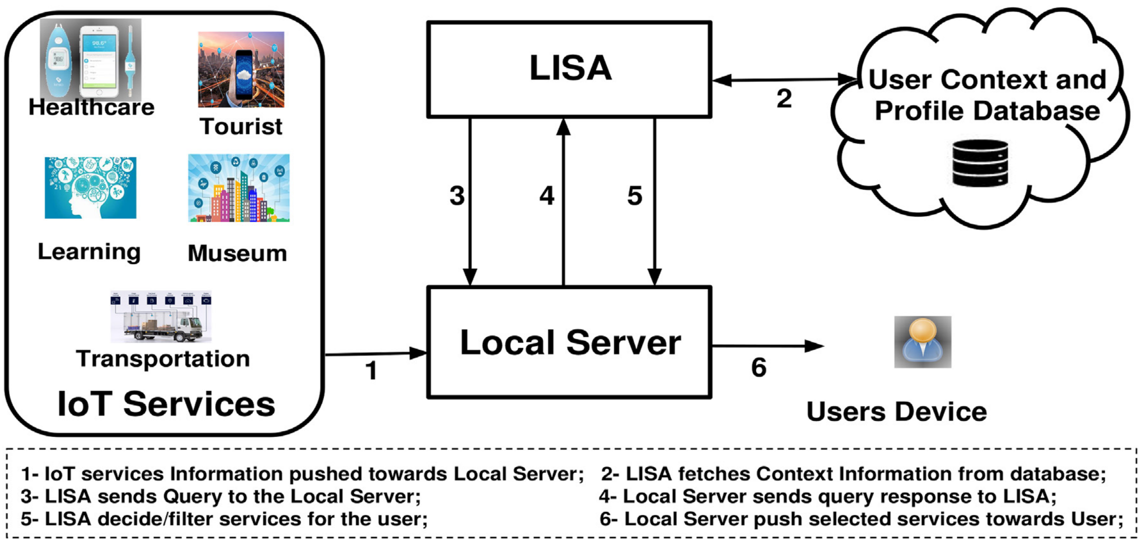
Healthcare
Tourist
Learning
Museum
Transportation
IoT Services
LISA
User Context and Profile Database
Local Server
Users Device
1
2
3
4
5
6
1- IoT services Information pushed towards Local Server; 2- LISA fetches Context Information from database;
3- LISA sends Query to the Local Server; 4- Local Server sends query response to LISA;
5- LISA decide/filter services for the user; 6- Local Server push selected services towards User;

**Fig. 4.** The simulation environment.

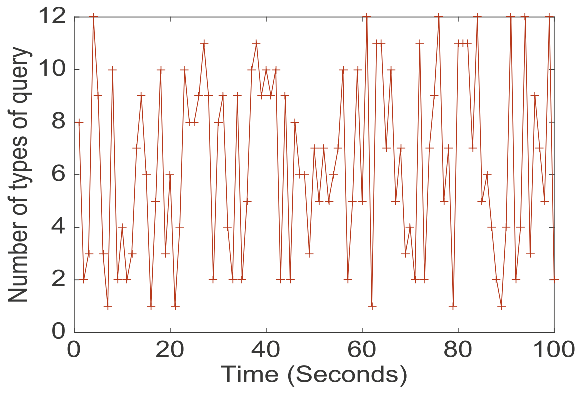

**Fig. 5.** Number of types of queries generated by past users.

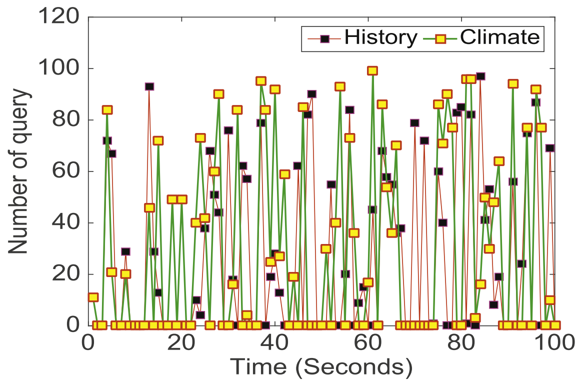

**Fig. 6.** Number of historical and climate type queries generated by past users.

Based on the synthesized data shown in 5 and 6, automatic queries are generated. As the selected queries depend on the threshold value set by the designer. We first show how the selected number of query value depends on the threshold value (more about it can be found in section 5.2). Fig. 7 shows the number of query terms selected by the IoT query user model at a particular area at different time instances. In our work, the aim is not to select only a specific number of queries, but the most important query terms, therefore, we set the threshold value as a percentage of the highest total weight assigned to a term as it is given in the following equation.

$$query_{threshold} = p{*}max\left\{total_{weight}(1)\ total_{weight}(2),\ldots\right\}where\ p = 50 - 90\% \tag{13}$$

The values in Fig. 7 shows that the number of selected query terms increases with increase in threshold value, however, the query terms are lower with lower threshold value, i.e., only the most important query terms are selected when the threshold is high. The use of lower threshold results in lower number of services being offered to the user, thus, it might possible that the users are not offered some the required services as well. while the higher threshold value could irritate the users with a lot of unnecessary services. In case of LISA, we observed that by using the contextual information the query terms can be significantly decreased even in when the threshold value is selected higher.

In order to check the importance of the services selected, we checked the precision and the recall values. As the IoT environments will be filled with a lots of devices providing different services, we decided that the number of services provided will be in the order of thousand, thus, we selected around 15000 services. We first observe the precision and recall while considering less number of services, however, later to observe long-term behavior of the selection process, we considered more number of services. Figs. 8 and 9 show the precision and recall for 500 and 1500 services.

The precision and the recall values very between 0.3 and 0.7. As the number of services increases, the values become constant. It is because our LISA architecture will only allow a specific set of services to be pushed towards the users, which will be based on users current contextual information. Hence, the increase in the different types of services pushed towards user will become constant after certain increase in the number of available services. To check the convergence as well as the scalabilty of our architecture, we also simulated by considering 15000 services. The result is shown in Fig. 10. The precision value becomes stable around 0.25 and the recall around 0.7, which is nearly the same with the results obtained in Figs. 8 and 9. Thus, it shows that LISA achieves desirable recall and precision even with very higher number of services in the IoT ecosystem. Finally, the processing time of the system versus the number of services generated in the system is shown in Fig. 11. It can be seen from Fig. 11 that as predicated the processing time increases with the number of services. From Fig. 11, we can safely conclude that in the existing scenario, around 7000 services would be better for the search and selection of the services without overwhelming the users.

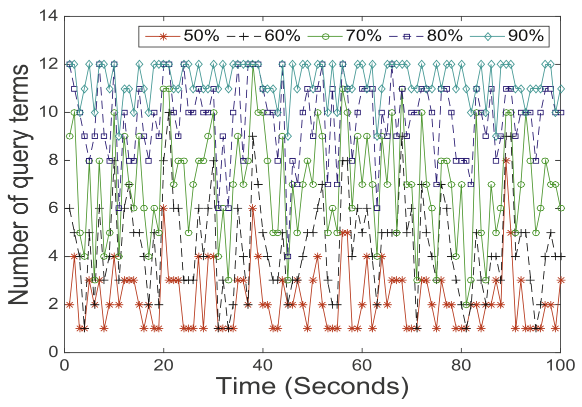

**Fig. 7.** Number of selected query terms for different threshold values.

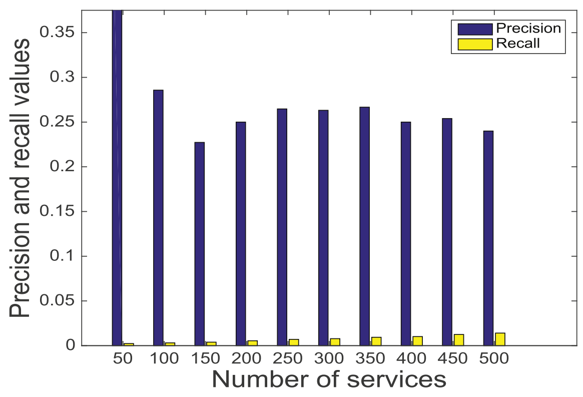

**Fig. 8.** The precision and recall for 500 Services.

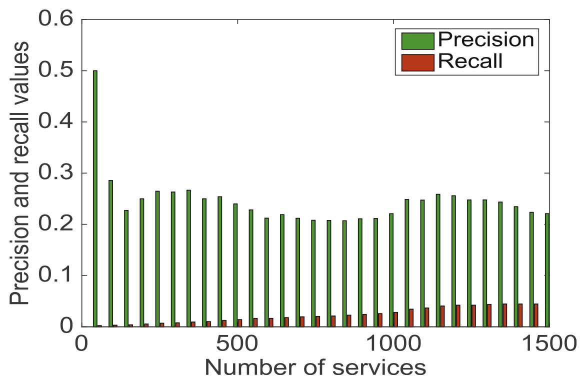

**Fig. 9.** The precision and recall for 1500 Services.

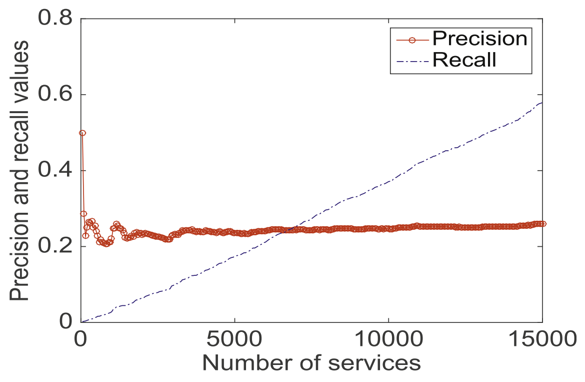

**Fig. 10.** Convergence of the precision and recall value with respect to the number of services.

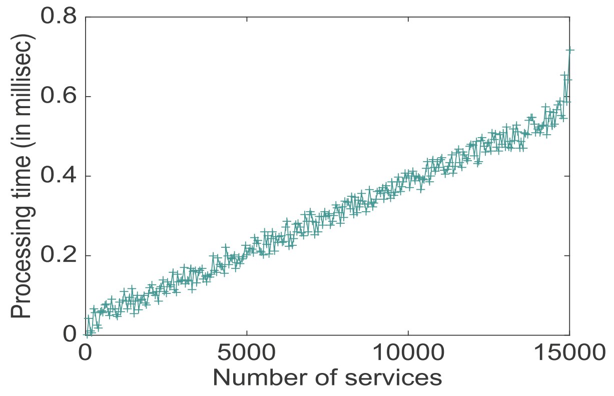

**Fig. 11.** The processing time of the system.

## 7. Conclusion and future works

In this paper, we proposed an architecture called LISA for automatic query generation for service selection for IoT push applications, which helps the IoT computing applications to provide the user desired services without overwhelming the user with unnecessary advertisements. To automatically generate the query on behalf of the user, our model uses contextual information concerning user history, context, and profile information along with the past queries generated at that location. The web service technology is used by our system to help the user to get a particular service without much user intervention. We showed the performance of the proposed architecture by simulating the system to generate the queries for the user and fetching the services automatically.

For future work, we are working to use several scoring methods to first score pool terms, and then rank them using new ranking model. Furthermore, envision the use of fuzzy logic, which can provide adequate weights for each term. We believe that it can be useful to select those terms that have high weight for query expansion and reformulate the query to improve the user context and provide better results to the users. Finally, we will also explore distributed trust based push protocol with collusion resistance by committing the IoT application broadcasters.

## Acknowledgement

Mauro Conti is supported by a Marie Curie Fellowship funded by the European Commission (agreement PCIG11-GA-2012-321980). This work is also partially supported by the EU TagItSmart! Project (agreement H2020-ICT30-2015-688061), and the EU-India REACH Project (agreement ICI+/2014/342-896).

## References

O'Hare, G.M., O'Grady, M.J., 2003. Gulliver's genie: a multi-agent system for ubiquitous and intelligent content delivery. Comput. Commun. 26 (11), 1177–1187.
Aksu, H., Babun, L., Conti, M., Tolomei, G., Uluagac, A.S., 2018. Advertising in the iot era: vision and challenges. IEEE Commun. Mag. arXiv:1802.04102.
Balke, W.-T., Wagner, M., 2003. Towards Personalized Selection of Web Services, pp. 20–24.
Bormann, C., Castellani, A.P., Shelby, Z., March 2012. Coap: an application protocol for billions of tiny internet nodes. IEEE Internet Comput. 16 (2), 62–67. ISSN 1089–7801.
Cao, H., Hu, D.H., Shen, D., Jiang, D., Sun, J.-T., Chen, E., Yang, Q., 2009. Context-aware Query Classification, pp. 3–10.
Carvalho, A., Cunha, C.R., Morais, E.P., 2010. A Framework to Support the Tourist's Information-needs Based on a Ubiquitous Approach.
Chen, M., 2013. Towards smart city: M2m communications with software agent intelligence. Multimed. Tool. Appl. 67 (1), 167–178.
Chen, I.Y., Yang, S.J., Zhang, J., 2006. Ubiquitous Provision of Context Aware Web Services, pp. 60–68.
Chiu, D.K., Leung, H.-f, 2005. Towards Ubiquitous Tourist Service Coordination and Integration: a Multi-agent and Semantic Web Approach, pp. 574–581.
Cho, C., Kim, J., Joo, Y., Shin, J., Oct 2016. An approach for coap based notification service in iot environment. In: 2016 International Conference on Information and Communication Technology Convergence), pp. 440–445.
Chuang, S.-L., Chien, L.-F., 2003. Automatic query taxonomy generation for information retrieval applications. Online Inf. Rev. 27 (4), 243–255.
Conti, M., Kaliyar, P., Remi, C. Lal, 2017. A reliable and secure multicast routing protocol for iot networks. In: Proceedings of the 12th International Conference on Availability, Reliability and Security, ARES '17. ACM, pp. 84:1–84:8. ISBN 978-1-4503-5257-4.
Doulkeridis, C., Loutas, N., Vazirgiannis, M., 2006. A system architecture for context-aware service discovery. Electron. Notes Theor. Comput. Sci. 146 (1), 101–116.
Douzis, K., Sotiriadis, S., Petrakis, E.G., Amza, C., 2018. Modular and generic iot management on the cloud. Future Generat. Comput. Syst. 78, 369–378. ISSN 0167–739X.
Duan, Z., Gopalan, K., Dong, Y., 2005. Push vs. pull: implications of protocol design on controlling unwanted traffic. In: *Proceedings of the Steps to Reducing Unwanted Traffic on the Internet*, SRUTI'05. USENIX Association, Berkeley, CA, USA, 4–4.
Fortino, G., Guerrieri, A., Russo, W., Savaglio, C., May 2014. Integration of agent-based and cloud computing for the smart objects-oriented iot. In: Proceedings of the 2014 IEEE 18th International Conference on Computer Supported Cooperative Work in Design, pp. 493–498.
Germanakos, P., Mourlas, C., Samaras, G., 2005. A Mobile Agent Approach for Ubiquitous and Personalized Ehealth Information Systems, pp. 67–70.
Gochhayat, S.P., Pallapa, V., Dec 2015. An efficient qos support for ubiquitous networks. IEEE Trans. Emerg. Top. Comput. 3 (4), 524–533. https://doi.org/10.1109/TETC.2015.2449669. ISSN 2168–6750.
Gudla, S.K., Bose, J., June 2016. Intelligent web push architecture with push flow control and push continuity. In: 2016 IEEE International Conference on Web Services (ICWS), pp. 658–661.
Gudla, S.K., Panchamukhi, S.K., Bose, J., Saride, G., Maheshwari, A., Jan 2016. Seamless push service with flow control for embedded devices. In: 2016 13th IEEE Annual Consumer Communications Networking Conference (CCNC), pp. 303–304.
Halder, S., Ghosal, A., Conti, M., May 2018. Limca: an Optimal Clustering Algorithm for Lifetime Maximization of Internet of Things. Wireless Networks.
Hashemian, S.V., Mavaddat, F., 2005. A Graph-based Approach to Web Services

Composition, pp. 183–189.
Karagiannis, V., Chatzimisios, P., Vazquez-Gallego, F., Alonso-Zarate, J., 2015. A survey on application layer protocols for the internet of things. Trans. IoT Cloud Comput. 3 (1), 11–17.
Kenteris, M., Gavalas, D., Economou, D., 2009. An innovative mobile electronic tourist guide application. Personal Ubiquitous Comput. 13 (2), 103–118.
Kim, H., Lee, E.A., 2017. Authentication and authorization for the internet of things. IT Professional 19 (5), 27–33. ISSN 1520–9202.
Kim, H.K., Kim, J.K., Ryu, Y.U., 2009. Personalized recommendation over a customer network for ubiquitous shopping. Servi. Comput. IEEE Trans. 2 (2), 140–151.
Kosuga, M., Kirimoto, N., Yamazaki, T., Nakanishi, T., Masuzaki, M., Hasuike, K., 2002. A multimedia service composition scheme for ubiquitous networks. J. Netw. Comput. Appl. 25 (4), 279–293.
Li, Q., Gochhayat, S.P., Conti, M., Energiot, F. Liu, 2017. A solution to improve network lifetime of iot devices. Pervasive Mob. Comput. 42, 124–133.
Lueg, C., 1998. In: Considering Collaborative Filtering as Groupware: Experiences and Lessons Learned, vol. 98, p. 16.
Manate, B., Munteanu, V.I., Fortis, T.F., Oct 2013. Towards a scalable multi-agent architecture for managing iot data. In: 2013 Eighth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, pp. 270–275.
Melucci, M., 2005. Context modeling and discovery using vector space bases. In: Proceedings of the 14th ACM International Conference on Information and Knowledge Management. ACM, pp. 808–815.
Melucci, M., White, R.W., 2007. Discovering hidden contextual factors for implicit feedback. In: Held in Conjunction with the 6 th International and Interdisciplinary Conference on Modeling and Using Context, p. 69.
Melucci, M., et al., 2012. Contextual search: a computational framework. Foundations and Trends® in Information Retrieval, pp. 257–405, 6(4–5).
Nayak, R., Lee, B., Nov 2007. Web Service Discovery with Additional Semantics and Clustering, pp. 555–558.
Nurmi, P., Salden, A., Lau, S.L., Suomela, J., Sutterer, M., Millerat, J., Martin, M., Lagerspetz, E., Poortinga, R., 2006. A System for Context-dependent User Modeling, pp. 1894–1903.
Recker, M.M., Walker, A., Lawless, K., 2003. What do you recommend? implementation and analyses of collaborative information filtering of web resources for education. Instr. Sci. 31 (4–5), 299–316.
Rho, S., Chilamkurti, N., Defrawy, K.E., 2013. Agent societies and social networks for ubiquitous computing. Personal Ubiquitous Comput. 17 (8), 1667–1669.
Roy, J., Ramanujan, A., 2001. Understanding web services. IT Prof. 3 (6), 69–73.
Shankar, P., Ganapathy, V., Iftode, L., 2009. Privately Querying Location-based Services with Sybilquery, pp. 31–40.
Shen, X., Zhai, C.X., 2003. Exploiting Query History for Document Ranking in Interactive Information Retrieval, pp. 377–378.
Sheng, Q.Z., Benatallah Contextuml, B., 2005. A Uml-based Modeling Language for Model-driven Development of Context-aware Web Services, pp. 206–212.
Silver, G., Maduko, A., Jafri, R., Miller, J.A., Sheth, A.P., 2003. Modeling and Simulation of Quality of Service for Composite Web Services.
Sim, K.M., 2012. Agent-based cloud computing. Serv. Comput. IEEE Trans. 5 (4), 564–577.
Singh, J., Prasad, M., Daraghmi, Y.A., Tiwari, P., Yadav, P., Bharill, N., Pratama, M., Saxena, A., 2017. Fuzzy logic hybrid model with semantic filtering approach for pseudo relevance feedback-based query expansion. In: Computational Intelligence (SSCI), 2017 IEEE Symposium Series on. IEEE, pp. 1–7.
Staab, S., Van der Aalst, W., Benjamins, V.R., Sheth, A., Miller, J.A., Bussler, C., Maedche, A., Fensel, D., Gannon, D., 2003. Web services: been there, done that? intelligent systems. IEEE 18 (1), 72–85.
Xu, X., Bessis, N., Cao, J., 2013. An autonomic agent trust model for iot systems. Procedia Comput. Sci. 21, 107–113. ISSN 1877–0509.
Xue, X., Croft, W.B., 2009. Automatic Query Generation for Patent Search, pp. 2037–2040.
Yang, S.J., Zhang, J., Chen, I.Y., 2008. A jess-enabled context elicitation system for providing context-aware web services. Expert Syst. Appl. 34 (4), 2254–2266.
Yu, C.-M., Gochhayat, S.P., Conti, M., Lu, C.-S., 2018. Privacy aware data deduplication for side channel in cloud storage. IEEE Trans. Cloud Comput. (1) 1–1.
Zhou, J., Cao, Z., Dong, X., Vasilakos, A.V., 2017. Security and privacy for cloud-based iot: challenges. IEEE Commun. Mag. 55 (1), 26–33.