Asynchronous Service-Oriented Architecture for Internet-of-Things based Control Systems

Nihal M
Department of Computer Science & Information Systems,
Birla Institute of Technology and Science, Pilani
Email: 2014HS120212P

Tareq Mohd Nazir
Department of Computer Science & Information Systems
Birla Institute of Technology and Science, Pilani
Email: 2018H1120285P

ABSTRACT

Service-oriented architectures (SOA) facilitate in the creation of services which are platform and language independent, composable, reusable, modular and loosely coupled. These services are self-describing and self-advertising. SOA provides communication standards and protocols by which the above properties are fulfilled. The authors in this paper describe various technologies involved in web infrastructure (TCP, HTTP, RESTful services), service invocation (SOAP), service description (WSDL), publishing and discovery (UDDI), service composition (BPEL), all of which constitute the various dimensions of SOA. They further investigate the enterprise-service bus (ESB) which is the mainstay of message-oriented middleware, that acts a common interface between various service technologies for providing platform independence and interoperability between clients and servers, all of which forms a part of the extended SOA. The main contribution in the paper is the proposal of an SOA based 'Internet-of-Things' (IoT) application leveraging a cloud computing infrastructure, using the design principles and various common standards of service-oriented architectures. The various research issues and limitations involved in SOA in terms of various quality attributes are also presented, along with concluding remarks on potential future work in this area.

KEYWORDS

Service Oriented Architecture, Internet-of-Things, Enterprise Service Bus, Simple Object Access Protocol, Universal Description, Discovery and Integration, Web Service Definition Language

INTRODUCTION

Present day consortiums need to react productively and swiftly to daily opportunities in response to the ever-changing international market strategies. For high-end business agility, organizations have to modularize the structure of the business process and expose them with the whole company in an elevated systematized manner. In order to deal with such scenarios, new concepts of making effective use of services was introduced as show in the Figure-1. These concepts facilitate in making the business processes more independent and loosely coupled. The pattern switch from by-product to service-oriented architecture system is currently oscillating the business, both from a financial and technical way of thinking. The current business requirements and the compacted finance for the project in several instances contribute to incompetent advancement of information systems. Rather, accumulating information systems of current software elements is the approach to be taken. Such software elements are defined as services. On a conceptual level, enterprise applications are made of local services and organizational services. In this particular architecture, the software resources are wrapped as "services", and are well established facilitating in independent modules which present conventional enterprise functionality that are self-sufficient.

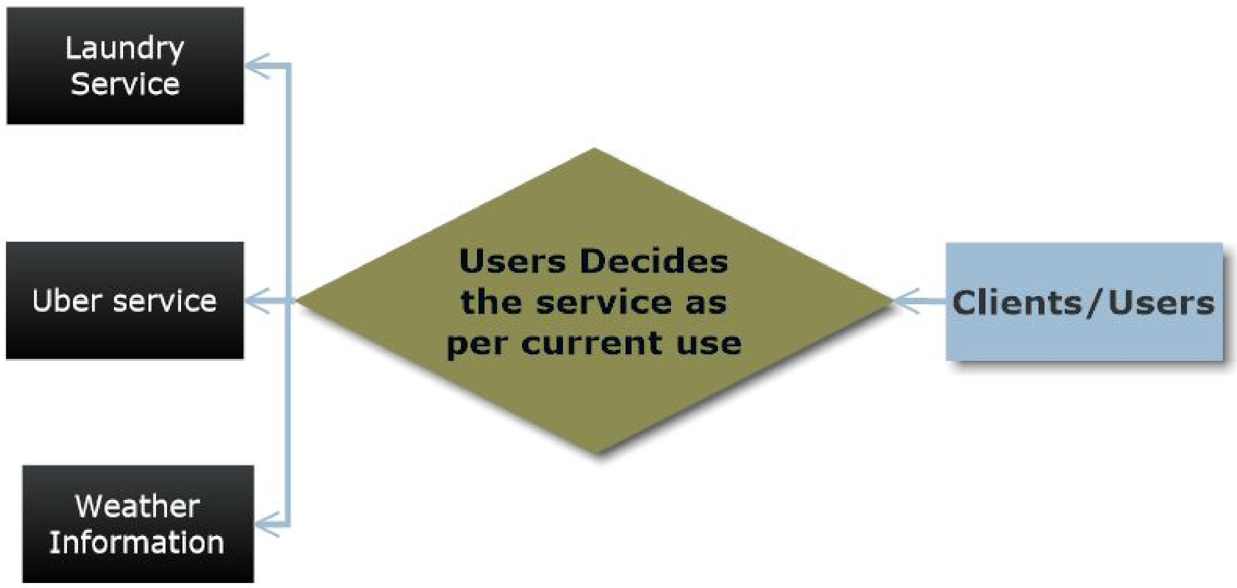


Figure-1 Various Services available for users in current generation

Conventional description language is used to define the services which use a published interface so as to interact with other demanding functions in a manner which allows completion of particular business processes or tasks. The authors in this paper mention the current service

standards that are in use, such as - Web Services Description Language (WSDL), Simple Object Access Protocol (SOAP), and Universal Description, Discovery, and Integration registry (UDDI).
Service Oriented Architecture is built in a way so as to preserve the advantages of distributed enterprise computing, such as application integration, transaction monitoring, security management over various platforms and also giving access to various access devices and legacy systems. The fundamental objective of service-oriented architecture is to eliminate these obstacles so that the applications can integrate easily and progress flawlessly. Service-oriented architecture can provide flexibility and agility which the enterprise users require, describing a modularized service that may be clustered in order for reuse that help make the business processes adhere to changes in the future.

Compared to other software architectures, SOA apprehends an obvious way of building a system by providing the services directly to the end-user applications or to other services which are published and can be discovered in the current network. SOA helps in building a framework that will enable enterprises to subscribe to the external existing services in their application or to another enterprise service which is being built to solve a different business process, thereby saving the time and cost of building the service from the core. SOA surpasses the struggles of software reuse, object-oriented programming and substitutable programming concepts. The Enterprise Service Bus (ESB) serves as a backbone for distributed computing and integration, thus aiding in the building of SOA architectures [1,2,3]. Channabasavaiah et al [18] give elaborate details and reasons to migrate to service oriented architectures.

Another major advantage of applying SOA is that it is language or technology independent. That means the services can be created using any language like JAVA or C#, and can be published for discovery by client users. This is accomplished by developing a common interface, and by using a description language ("Web Service Definition Language" when using web services). This interface contains the behavior and functions through which the services can be accessed to perform a certain business process. A service present in a service-oriented architecture is an accessible part of the functionality containing three main features. The primary feature of SOA is it exists in an independent state. A secondary feature of SOA is that they are not platform dependent. Finally, SOA focuses on providing services that are dynamically located, referenced and interconnected.

In this literature survey the author is trying to explain SOA, its benefits, methodologies and the latest technologies which can be incorporated using SOA standards to deliver the pacing needs ot the transforming technologies. The author is broadening the discussion by introducing the principles, abstract ideas and the implementation in the area of application integration, middleware, and common object brokers, Enterprise Service Bus (ESB) [1,2]. Also, the author has included an insight on applying the SOA in the IoT systems. As the advancement in the IoT systems which is leading towards, systems composing of millions of heterogeneous devices there should be change in way systems are generally designed, deployed and they access services. To overcome these challenges the author is suggesting to use the SOA with the IoT system. For purpose of showing the SOA with IoT system author has proposed an abstract view of IoT Home automation system. SOA is widely used for various other applications including health care [19].

Internet-Of-Things (IoT) has attracted a lot of attention during recent times, due to the vast amount of its estimated potential applications. People all over the world are now connected by the internet, and if the objects around us could also be connected to internet, we could leverage many more benefits from such a global communication. An example of the potential benefits of IoT is design of smart homes via smart home automation system. The authors propose the benefits of such an application, and how SOA could further enhance the existing advantages due to IoT. All the devices, computers, gadgets, objects, sensors around us could be interfaced with the internet to communicate, share and use data, and various smart applications can be built on top of this. The garage door can prepare to open when the car is nearby the house. Burglar alarm messages can be sent to the owner of the house in case of burglary. Anything connected to the IoT appliance can be automated in this manner, and suitably customized to meet the end demands of the user.

The Home Automation system that author proposed is not working modal but an abstract view of how IoT system should perform when the SOA framework is applied on it. The system helps in performing all the functionalities of the house such as coffee brewing, bedroom light control, bedroom lamp control and garage door control. The rest of the paper is organized as follows: Section II talks about the related work done with respect to SOA. Section III presents SOA current standards and introductory concept of web services and Section IV presents the Application of Service SOA on an IoT based system. In Section V the author presents the state-of-the-art Home Automation System. Section VI contains limitations and roadblocks in remodeling the IoT system with SOA based approach. In section Section VIII the author concludes by stating the Future work.

RELATED WORK

In the 1980's, the phrase "middleware technologies" gained popularity. The developments of the middleware technologies gave rise to idea of connecting two devices for inter-communication without requiring any dependencies. The enhanced version of these middleware technologies gave rise to a type of architecture that could be applied in various distributed software systems , and was termed as service-oriented architecture in 1998. The first step taken towards introducing process to process communication residing on two different systems was Remote Procedure Calling (RPC). A.D. Birrell and B. J. Nelso [20] explain the idea of delivering communication over network between two processes programmed in same high-level language. Figure-2 displays an abstract view of a simple remote procedure call where system1 sends a request or message to system 2 via a client stub, and system 2 replies to system 1 via a server stub. Client stub is capable of marshalling the parameter passed from system 1 so as to send the parameter to system 2 in message format. The main drawback of RPC was that it was not platform and language independent.

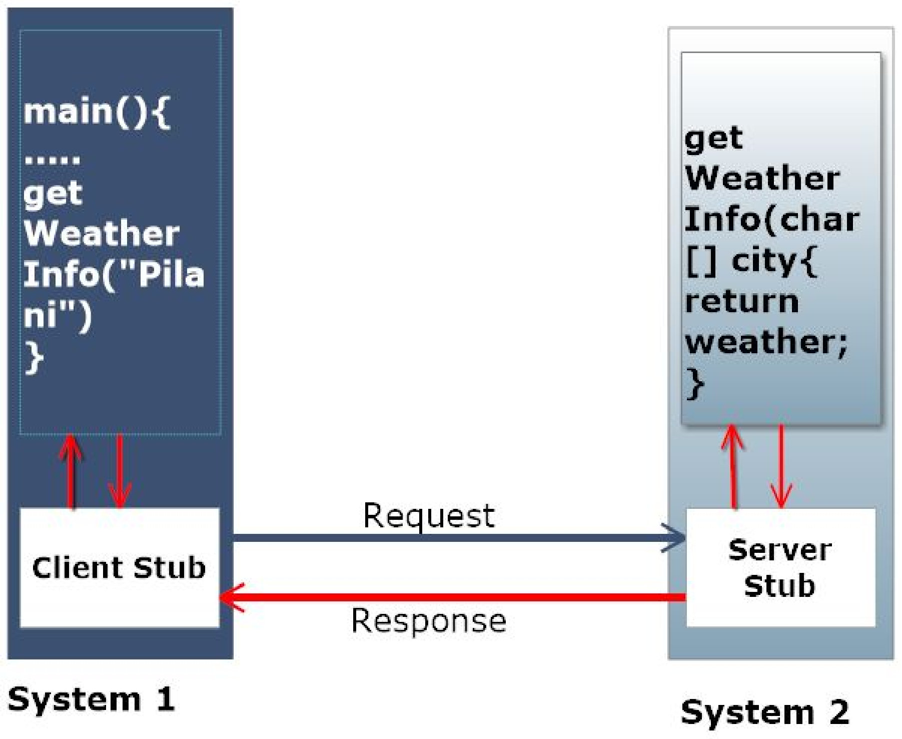


Figure-2 Remote Procedure Calls

Next architecture which gained a lot of success was the "Common Object Request Broker Architecture" (CORBA) which resolved the drawbacks of RPC. In 1989, the "Object Management Group" was formed to accomplish the task of developing the standard way of communication over the network without the limitation of platform and language dependency of distributed networking [16]. CORBA (shown in Figure-3) works on the same principle of RPC, and the similarity between them is visible in Figures - 2 and 3. Two more components are added in CORBA, namely, Interface Description Language and secondly the Object Request Broker. Interface Description Language allows communication between two software that are written in different languages. As per Steve Vinoski [16] IDL resolves the issue of interoperability which was a limitation in RPC. Object Request Broker is another part of CORBA which allows function calls to be made between two systems connected in a distributed networking environment. Thus, the ORB resolves the issues of portability which was a drawback of RPC.

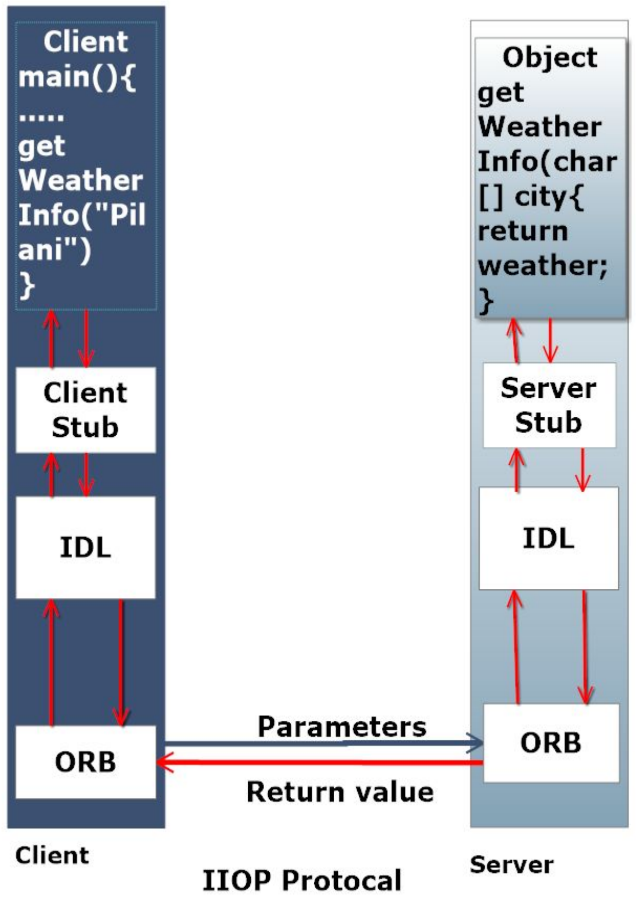


Figure-3 Common Object Request Broker Architecture

Though CORBA resolves these issues, it has its own drawbacks. One of the drawbacks is that it requires that all n systems be connected to each other, and thereby making the network cumbersome as shown in the Figure-4. It also has issues regarding resource scalability. This implies that an addition of one system to the network implies its connection with n number of services in the network. This gave rise to the next new concept called the Enterprise Service Bus.

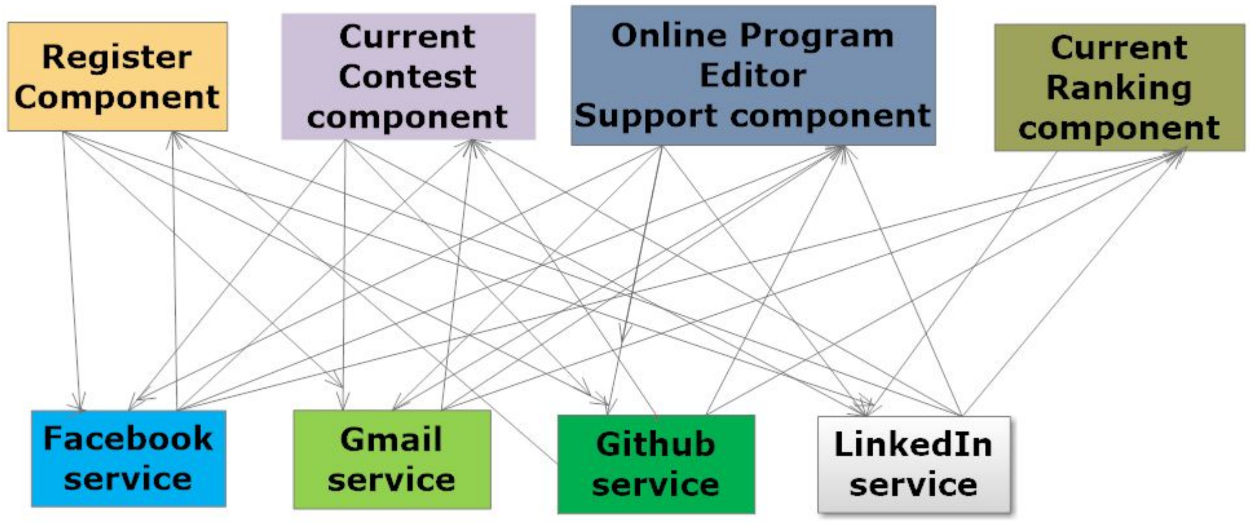
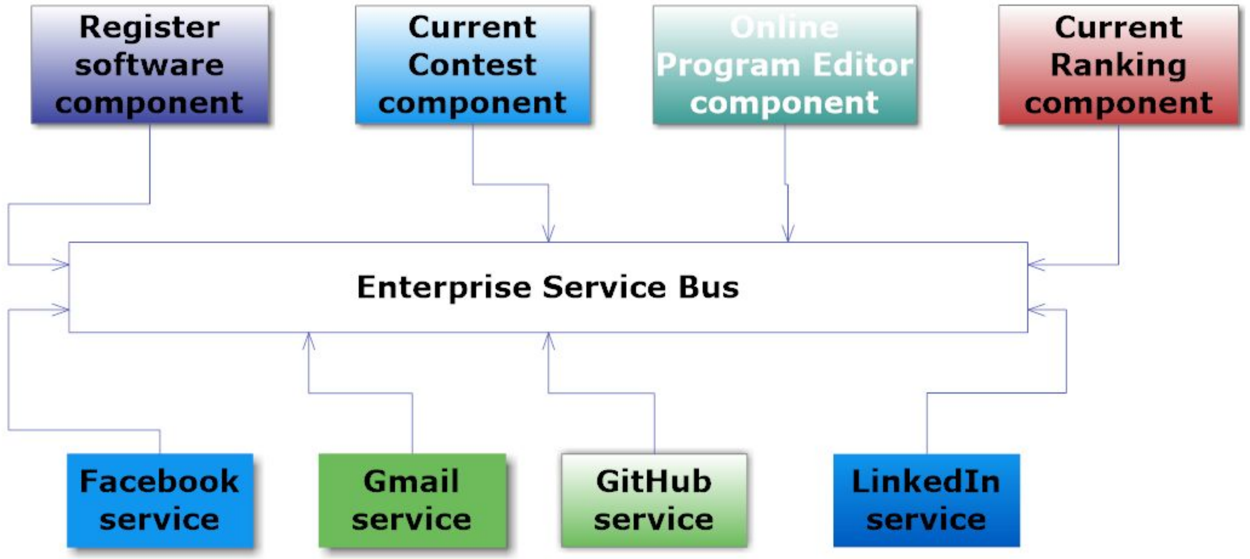

Register Component
Current Contest component
Online Program Editor Support component
Current Ranking component
Facebook service
Gmail service
Github service
LinkedIn service

Figure-4 Different domain interfaces connected to connected to each and every service using Common Object Broker Request Architecture

As per the authors Colombe H´erault, Ga¨el Thomas and Philippe Lalanda [18], Enterprise service bus is a standard way of communication between the various components and the services to which they subscribe. ESBs act as a bridge between the Web services and business applications and it tries to expose the former to the latter via an asynchronous or event driven mediator architecture, and also provides a way to exchange messages between them by various translation and routing mechanisms [1,4,3]. Events are clearly different and distinct from services in their contextual usage and applications, and these are described by Bloomberg et al [12] in their paper "Events vs Services". ESB is built on top of various other standards such as XML, SOAP, WSDL, Connector Architecture, WS-Security, WS-policy and WS-ReliableMessaging to support the heterogeneity of various software applications and services. ESB provides various functionalities such as service orchestration which is based on Business process Execution Language (BPEL), and asynchronous communication of services and therefore helps in discovering the best service for the user [18].



Register software component
Current Contest component
Online Program Editor component
Current Ranking component
Enterprise Service Bus
Facebook service
Gmail service
GitHub service
LinkedIn service

Figure-5 Different domain interfaces connected to connected to each and every service using Enterprise service bus

A more advanced way of using the SOA is to use it with RESTful services instead of SOAP as it is are easier to work with and is restricted to use secure protocols such as TCP and HTTP/HTTPS [10].

The Internet-Of-Things recently created a paradigm shift in the way people think of internet. It comprises of two terms - internet and things. Internet refers to a group of computers connected to each other for the purpose of sharing data in standard formats of communication like HTTP, TCP/IP protocols and XML formats of data [11]. But the idea of IoT extends this concept further. Now a days, almost any object can be interfaced with the internet, and objects along with web pages can interact with users exchanging temporal data of all kinds.

The 'things' in the term IoT can literally refer to any living or non-living thing [11]. The general notion of a 'thing' indicates to gadgets, devices or sensors like phone, laptop, car, radio, music system, air conditioner, automobiles, temperature sensor, IR sensors etc. But IoT also includes objects which do not ordinarily exchange data like washing machines, TV, cycle, box, pens, notebooks etc. IoT is so broad that it includes living entities like plants, trees, humans, animals and literally any object or living being we see around us [11]. This broad notion of IoT makes it very relevant to current world scenario where nearly everyone uses and has access to internet and other gadgets that are linked to the internet.

The authors specifically choose home automation as the IoT application because of its wide usage and popularity in present scenario. Pavithra et al [12] demonstrate the usefulness of IoT in home automation by building a prototype that monitors human activity and controls various sensors in a user customized way using Raspberry Pi module. They also propose a user-friendly GUI which interacts with the world-wide web for communication. Various other low power modules like zigbee and WiFi are also used to aid communication between devices and Raspberry Pi server. The user moves with the IoT mobile application, and various sensors like fans, lights and door locks are remotely controlled via the internet. The mobile app alerts the user in case of smoke via the usage of suitable smoke detection sensors interfaced with the server. User can control the various home appliances remotely, and in case of power failure or server failure, the various appliances are domestically controlled by the system leading to improved fault tolerance. Fault tolerance is a very important aspect of home automation as can be seen in the paper by Zaiter et al [21]. Nancy et al [14] in their paper also conduct a systematic survey on fault tolerance under certain assumed conditions, focusing specifically on elderly people who are averse intrusive sensors which give false alarms due to high false positive rate.

Ravi et al [15] in their paper also propose a similar home automation system called the smart security that alarms the user when someone trespasses their property. The recurring theme of IoT being applied for various home automation appliances indicates the potential scope for innovation. The authors therefore rely on SOA based approaches to improve on the non-functional requirements of the various IoT based appliances. Mohamed et al [16] in their paper discuss various approaches to adopt SOA in IoT based systems. Lan et al [17] in their paper build on these concepts further to show the utility of SOA based architectures in IoT. Various architectural concepts can be borrowed from SOA to enhance quality attributes of IoT based systems. The authors take home automation as a use case scenario to demonstrate the SOA paradigm in IoT, but the same concept can be generalized to other user based IoT systems. The effectiveness of such an approach is shown in further sections of the paper.

In-spite of the huge success of IoT in the above-mentioned appliances, there are still many research challenges as outlined by Daniel et al in [18]. Identifying each object in various scenarios itself poses several problems. There are other challenges related to sensing and actuation, data confidentiality, privacy and security that are outlined in [18]. Some of these problems are solved via the concept of distributed systems technology and distributed intelligence, which are beyond the scope of this paper. But the authors mention them here as a reference for the interested readers.

SOA CURRENT STANDARDS AND INTRODUCTORY CONCEPT OF WEB SERVICES

Figure-6 shows a simple web service architecture and network stack that declare the functions needed to be performed by different instances present in the web service.

1) Web service functions

Web services play three major functions in a web service architecture as described below.

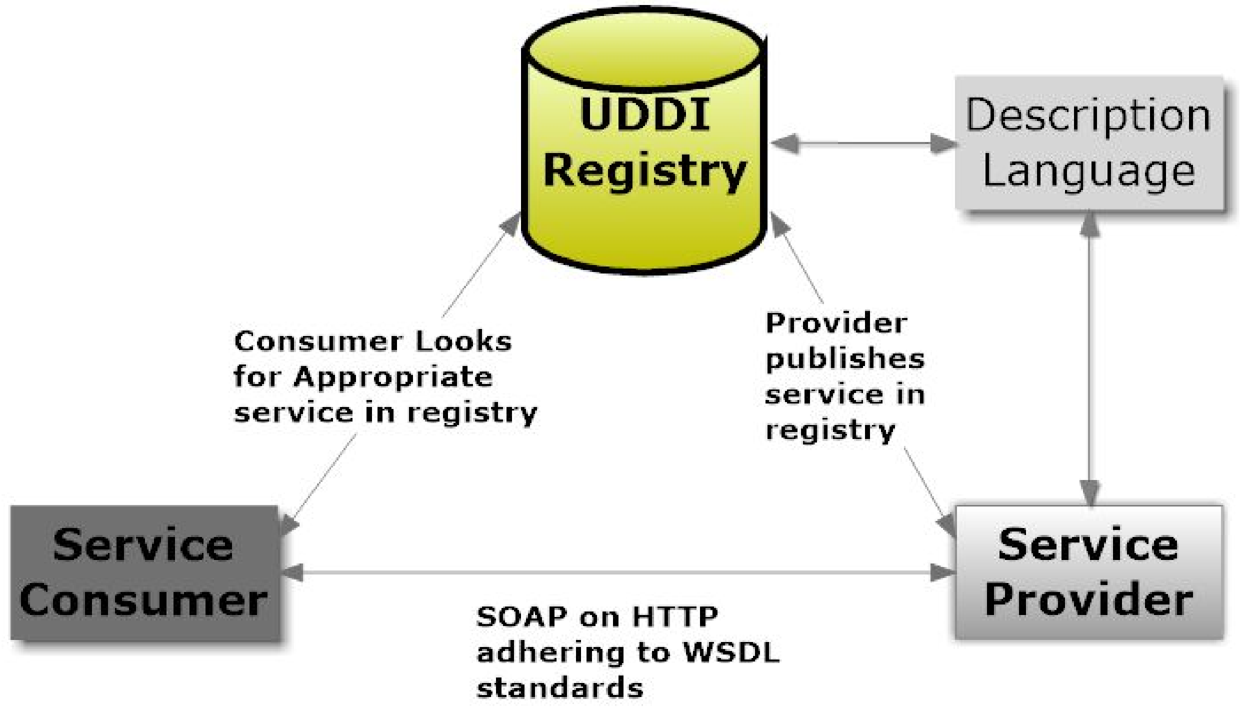

UDDI Registry
Description Language
Consumer Looks for Appropriate service in registry
Provider publishes service in registry
Service Consumer
Service Provider
SOAP on HTTP adhering to WSDL standards

Figure-6. Simple Web Service Architecture.

UDDI Registry - This is a transitive repository arrangement of services which are made easily accessible. It is also a central space where service providers can easily publish services and also discover or subscribe to any readily available services [5,6,21].

Service Provider - This component maintains the service and the system that offers single or multiple services for customers who use the service. For the thorough promotion of services, the service provider publishes the service into the UDDI registry concurrently including the service contract which includes the definitions of kind of service, its usage and traffic of the service.

Service Consumer - This component does a thorough search for the available service in the UDDI repository and selects the appropriate service to be referenced with the service of the service provider. It also requests for proceeding with the task whenever a service call is invoked.

2) Web Service Network Stack

The network stack describes the flow of web services that are included in network communications with the help of a standard protocol like HyperText Transfer Protocol ("HTTP") or HyperText Transfer Protocol Secure ("HTTPS”), Simple Object Access Protocol message protocol, XML language, UDDI web service registry repository and "Web Service Definition Language" (WSDL) to create link between the processes.

“Simple Object Access Protocol”

SOAP is an XML language-based messaging protocol which is liable to request and receive messages over a shared network enabling some of the key features of SOA that aid in making the communication between the component’s platform and language independent. For transferring messages, SOAP uses "HTTP" protocol and the messages are in an XML format. SOAP allows instance or object sharing between the applications giving them access to call these objects and functions that are existing on shared servers. This protocol consists of a template called an envelope that is used for describing the type of message being sent and how the message is going to be transferred and the rules being used for representing the remote procedure calls or functions.

"Web Service Definition Language"

WSDL is nothing but an interface maintained in XML language and is utilized to define the web service, remote procedure calls and how these procedures are going to be called in a peer to peer and shared environment. There are different XML elements such as - types, messages, port type, binding and service, which are some of the tags that can be used in a WSDL document which defines the web service. The type element describes the data-type being used by the web service and the message tags define the input and output arguments of the web service functions. Each message also contains a part element. More than a single part tag can be used in the message tag. The part tag represents the arguments of a procedure call which should have a unique type and name defined. The port type tag defines the bunch of procedures to be executed by web service and it also describes the messages which are involved. The binding tag present in the WSDL document defines specifications of all procedures, message formats and their rules. At last the service tag defines the endpoint which defines a certain type of binding.

“Universal Description, Discovery and Integration”

This component is self-contained, integrated, shared and XML dependent custom repository utilized for advertising and discovering web services. The UDDI repository does not have the complete features of the services hosted, but it can redirect to different sources that have concrete service implementation. Information present in the UDDI is in the form of an XML language where the service providers can advertise their services in three types of pages called “White pages, Yellow pages and Green pages” [5,6,21].

3) Potential Discovery Mechanisms

Currently, a vast portion of web services on the internet is considered to be running. So finding the appropriate and valuable service is a valuable aspect in order to provide a better user experience and performance to the customers. Currently, there are various state-of-the-art approaches for web service discovery, and a few of them are mentioned below.

1. Automatic and dynamic discovery-based approach (Example: OWL)
2. Granularity based discovery-based approach (Example: WordNet)
3. Keyword-based discovery-based approach (Example: UDDI)
4. Comprehensive discovery-based approach (Example: Any play-store)

APPLICATION OF SERVICE-ORIENTED ARCHITECTURE IN IoT SYSTEMS :

The main principle of IoT systems is to create a network of daily usage components using the internet as a medium of communication. All the components that are interconnected on a network can communicate with each other to solve and ease the daily human activities. It saves time and energy and hence reduces the cost in long term. The IoT systems demand that all the components should be loosely coupled, independent, up and running, and internet enabled to provide the service that humans need. To fulfil these demands in an abstract way, we use service-oriented architectures to make the components independent of each other, and to expose each component to every other component using internet as the common interface. The IoT system based on SOA consists of 4 layers as shown in the figure-7.

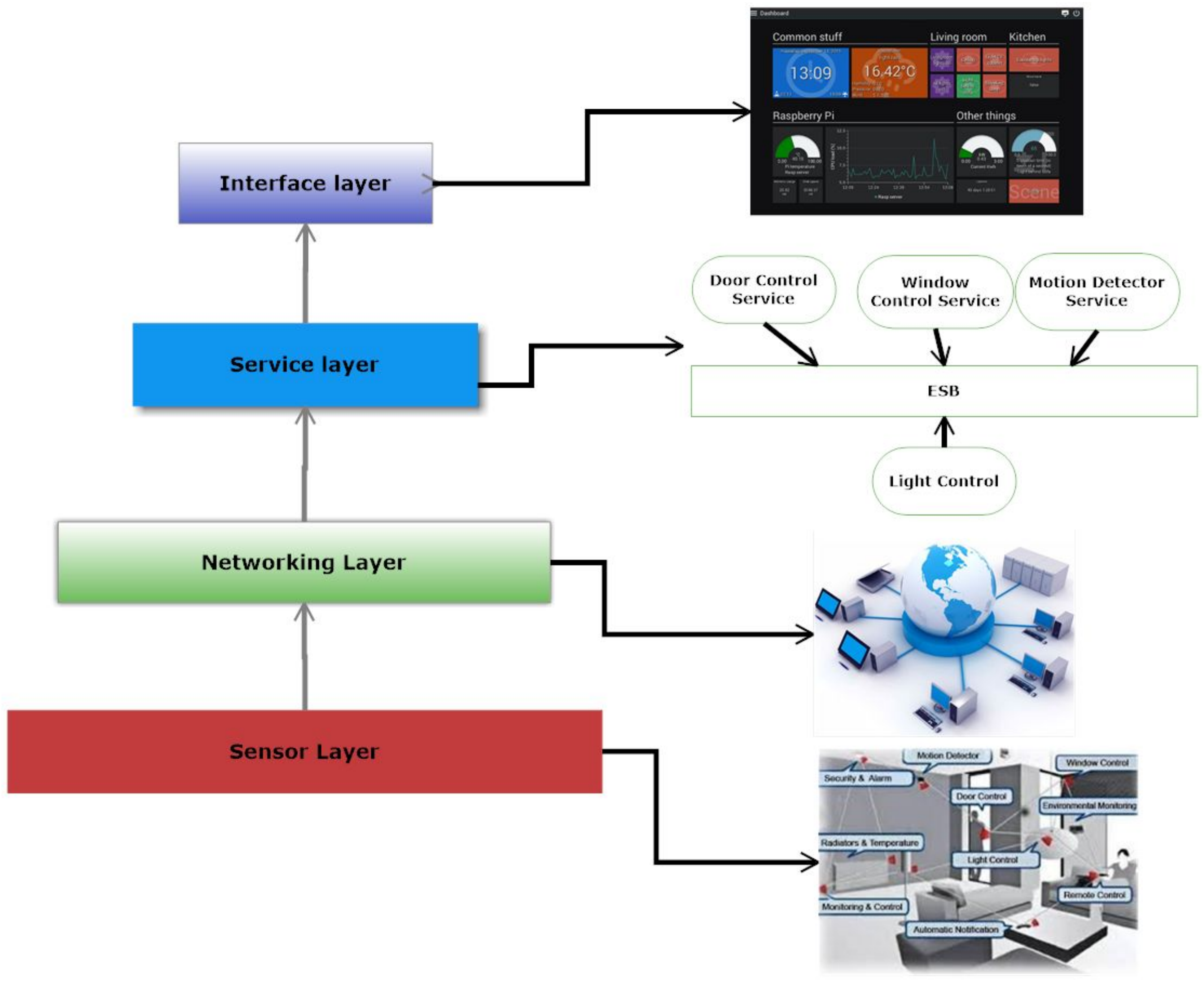


Figure-7. Layers of IoT system based on SOA

1) Sensor layer

The sensor layer consists of hardware components which are connected to each other through the internet. These sensors are responsible for generating the signals when they are exposed to the physical medium from which the signal can be generated. In this layer perspicacious systems through exposed sensors can automatically detect the physical medium and can perform the data interchange between the devices. Each sensor has its own unique digital identification number (for example a universal identification number (UIN) present in each hardware device). This ensures that sensor can be traced back thus achieving the goal of IoT system expectation to be physically available on network and being remotely accessible all over the world.

2) Networking Layer

The networking layer is a foundation which connects the wired components, wireless components or any other device in the internet-of-things architecture. This allows them to detect the environment, communicate to various devices in the internet, share data between things, enable asynchronous calls and its management, and smart IoT processing. This layer authorizes the organized communication between the devices in the IoT environment and also manages the messages between the sensors and systems. If IoT system is enabled with an SOA based approach, the previously subscribed services will be available on the networking layer. This layer is crucial in an SOA based IoT system approach, and takes into consideration the quality of service functionalities, security, data exchange between the devices and control flow among many other quality attributes.

3) Service Layer

In this layer services that are desired by the user and software are produced and managed. It is mostly based on middleware technologies, which is a fundamental concept for utilizing the services and also the implementation of IoT applications, where together hardware and software can reuse. Middleware acts as a major component in the development of upgraded applications and services.

4) Interface Layer

The last layer is the Interface layer and it is accountable for providing the application to various IoT users. Interface layer plays a major role in delivering services or applications that merge or analyze the information gained from previous three layers. Most of the production of IoT systems are in the areas of agriculture, smart cities, health complex home automation, water management among many others.

PROPOSED SYSTEM DESIGN

Users are experiencing increasingly flexible systems today, that can be extended to be compatible with android and inter-OS platforms, and the design the authors are proposing does not reduce this flexibility. The authors propose a system which accommodates more comfort, flexibility, modifiability, extensibility, safety, adaptability, reusability and modularity. To support these quality attributes author suggests to use an asynchronous or event driven Service Oriented Architecture with the IoT applications.

The authors intend to design and propose an effective home automation system that can be open sourced in a cost-effective manner. The various components and modules involved in the home automation system can interact wirelessly with servers kept at home, and by doing so can add greatly to the flexibility of the proposed architecture. Various communication standards can be used, along with a local area network to facilitate easy inter-network communication between the various hardware and software components.

The proposed system architecture is shown in figure-8. The authors use various sensors like infrared (IR) sensor, light emitting diode, motion detector sensor, pneumatic controlled actuators. The exact part details and component specifications are shown in table-1. From the table it can be seen that the authors use Arduino Mega from the Atmega 2560 series to interface the various kinds of sensors. Arduino is a part of the middleware used for interacting with the various kinds of sensors used. The ESP8266 Arduino Mega has 32 MB memory for processing purposes.

Authors use PIR sensor, which is a pyroelectric infrared module in order to detect persons entering/leaving the room. This can be suitably coupled with Arduino for detecting people entering a room or a building, and can be used for garage door control or bed room light control as shown in figure-8. EE-2PCSIRSENSOR IR proximity sensors are also used for the same purpose.

A relay module shields capable of handling 10 amperes and 5 volt potential difference is used for switching purposes, and the connections between the various components and Arduino are made using male-male or male-female jumper wires, as and when necessary, depending on the kind of component being interfaced with the Arduino.

The LDR (photosensitive optical module) is specifically used for bed lamp control and bedroom light control which comprise a part of the home automation IoT system. When the voltage level goes below a threshold, it denotes the dimness of light in the room, indicating night time and thereby signals the Arduino to turn off the lights. This basic criteria for detecting night time can be augmented with user-based custom criteria's to make the overall module more user-friendly, making it more comfortable and easy for the user to use.

Other sensors and actuators like servo motors, stepper motors, batteries and their corresponding chargers are useful, especially for applications like garage door control, which requires actuators that are interfaced with sensors and Arduino to open/close the garage doors. Authors use the TowerPro MG995 Metal gear Servo motor which facilitate 180 degree rotation.

Type	**Specifications**	**Brand**	**Weight**	**Manufacturer**	**Color**

		Name		**Series/Part Number**	
Arduino Mega	Atmega 2560 (with WiFi R3)	Atmega	150 grams	NodeMCU ESP8266 (32Mb Memory USB-TTL CH340G)	Black
PIR sensor	Pyroelectric Infrared Module	Generic	10.0 grams	KG001	Green board with blue and black components
Relay Module Shield	5V 10A 2 Channel Relay (Arduino ARM PIC AVR DSP Electronic)	Generic	30.0 grams	5V 10A 2 Channel Relay Module Shield for Arduino	Black board with blue components
Jumper Wires	Male-male, male-female	Generic	100.0 grams	DPNT	Red, green, blue, white
IR proximity sensor	For line following and obstacle avoidance robots	Easy electronic s	100.0 grams	EE-2PCSIRSENSOR	Blue board with black components
Optical photosensitive LDR light sensor module	Arduino Shield DC 3 5V	INVENT O	50.0 grams	INVNT_10 Lm393	blue

Table-1: Component Specifications and Details

Wifi routers can be configured with the Arduino in order to provide greater flexibility to users of the home automation system. Users can suitably configure the wifi router with relevant IP addresses and thereby facilitating them to communicate directly through the wifi unit.

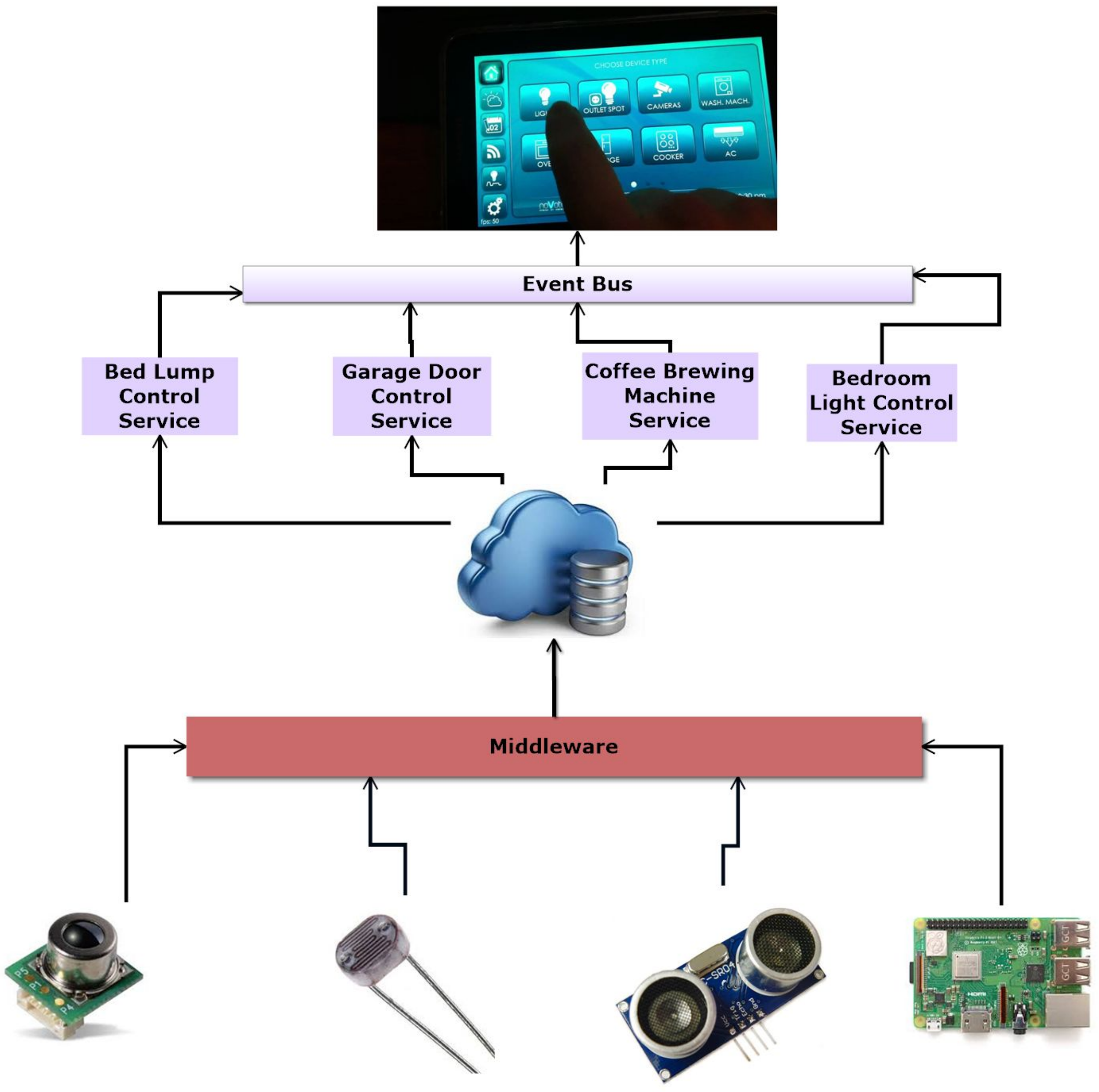


Figure-8. Proposed SOA based IoT Architecture and its different Layers

ABSTRACT DEMONSTRATION OF EVENTS IN REAL TIME HOME AUTOMATION SYSTEM

Figure-9 illustrates a few of the events which are functioning in the home automation system. The figure shows an interaction diagram to illustrate a morning routine as a service. As the Arduino controller detects the arrival of wakeup time which is set by the user, it enters into a loop which increases the bed light gradually to wake up the person. The change is gradual and not abrupt so that the user sleep isn't disturbed by the sudden increase in luminosity. Arduino controller uses PID (proportional integral derivative control) in order to use digital output pins to produce analog output voltage which can be used to control the intensity of light. The luminosity of the light bulb is proportional to its input voltage.

In a similar way the arduino controller increases the alarm volume to awaken the user, and the loop breaks once the max volume is reached. The Arduino then turns on the juice vending machine to prepare a cup of morning juice for the user after he wakes up.

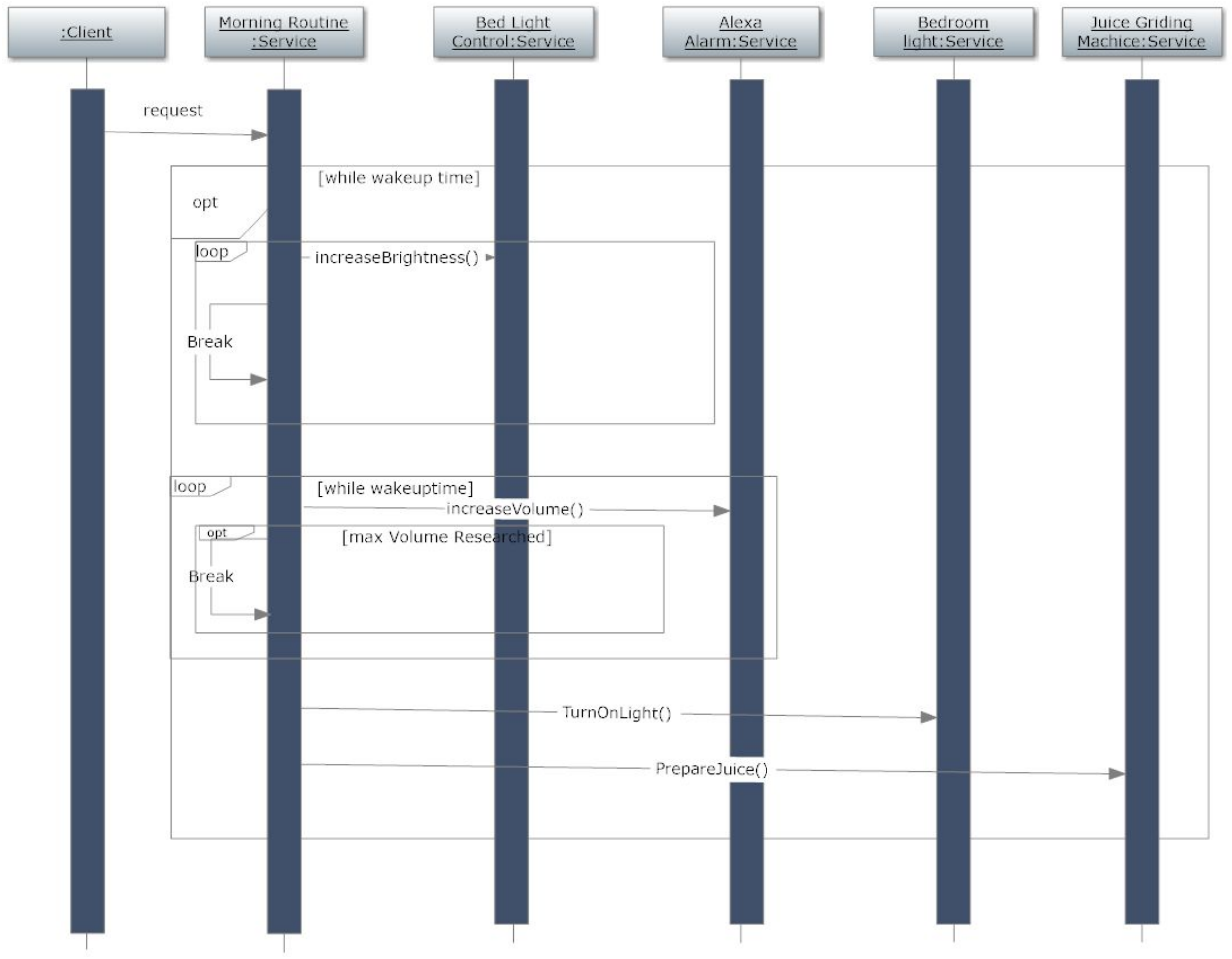


Figure-9. Events in Real Time Home automation system

LIMITATIONS AND ROADBLOCKS IN REMODELLING THE IoT SYSTEM WITH SOA BASED APPROACH

There are multiple limitations and roadblocks when it comes to remodeling IoT systems using SOA based approaches. These limitations can occur in any area of management, testing and more specifically in the area of security. The IoT automation systems which are remodeled according to SOA produce a lot of asynchronous messages depending on the events invocation across the peer to peer components in various directions. Keeping a record of this messages and replying to each of the message is a substantial limitation. Considering a different scenario where the IoT system uses several mediator applications with numerous numbers of message types would lead to increased complexity of tasks related to managing these various message types. Another limitation while reshaping the IoT system with SOA based approach is related to security. It might get cumbersome to provide security at each and every layer of the system. SOA helps in the making the IoT systems interoperable but it's inflexible use increases the issue of scalability. Although IoT system tend to scale with age applying SOA on such system make it highly inflexible to any type of scalability. Below are some of the major issues found to applicable to the IoT system based on SOA.

Nonexistence of specific format for describing the services: There are many defined specifications for web services for example "WSDL"," SOAP" etc., but still there is no particular defined specification for description services available for this type of services. So, this becomes a great limitation while doing the integration from various services. Recent buzz is to apply the OWL of semantic web technologies in context awareness, but there is still no new standardized procedure for describing the IoT services is used till date.

Deficiency of context awareness dependent services: In the recent trend context awareness is being utilized with SOA based services satisfactorily. Today's IoT systems are affected by minimal context awareness due to random distribution of ontologies and unclear semantics for services.

Complicated data visualization and investigation: Systems which are IoT based systems produce a huge stack of data that require big data management techniques. The current techniques do not provide IoT based big data management efficiently. In use standards do not provide high scope of data storage and visualization. This limited visualization and analysis does not provide any valuable insight of the system behavior. To address the above-mentioned issues few challenges, need to be overcome. Following are the challenges that need to be focused on.

Scalability: The pliability of the services being used in the IoT systems gives rise to development of advanced kind of services. This produces the IoT services whose service discovery is a complex task. The part executed by sensors as providers is distinctly defined. The integration of already available IoT services will become a limitation.

Heterogeneity: Latest IoT services should be interoperable with the older systems and support diverse service integration. This is practical because of accessibility of various service discovery rules and service data models which can be utilized by subscribing. Semantic diversity, is a hard challenge as different data models get adjusted to various data portrayal techniques.

Mobility: IoT services predominantly use the wireless sensors. So, to come up with new ideas of integrating the IoT services in various domains, services should hold up to the mobility. Producing standards for these services with mobility is essentially a tough task.

Security: Any prominent technology is prone to security issues, which can become a bottleneck when it comes to adaptability in new areas. So, ensuring a high security in the IoT based services which are dependent on the sensors is a highly unreal task. Although the security cannot be compromised at any cost.

Fault Tolerant Environment: IoT services appeared to serve in discrete hazardous environment. Usually in these cases error management is a laborious task. To take an example it is very difficult to enhance life cycle of service by forcing various new techniques such as sensor awake and sleep. To perform these operations in real time environment is challenging.

CONCLUSION

The authors demonstrate how the various aspects of SOA can be successfully integrated into an IoT based home automation system. There are various other research challenges which need to be tackled. Data confidentiality, privacy and security related issues come up while dealing with larger IoT systems shared across networks and accessed by multiple concurrent users at the same time. Solutions to some of these problems have been mentioned in the related works section of the paper, and the authors assure these improvements as part of their future work.

REFERENCES

1. Papazoglou, Mike P., and Willem-Jan Van Den Heuvel. "Service oriented architectures: approaches, technologies and research issues." *The VLDB journal* 16.3 (2007): 389-415.
2. Papazoglou, Michael P., et al. "Service-oriented computing: State of the art and research challenges." Computer 40.11 (2007): 38-45.
3. Robinson, Rick. "Understand enterprise service bus scenarios and solutions in service-oriented architecture." IBM DeveloperWorks, June (2004).
4. Channabasavaiah, Kishore, Kerrie Holley, and Edward Tuggle. "Migrating to a service-oriented architecture." IBM DeveloperWorks 16 (2003): 727-728
5. Avila, Karen, et al. "Applications Based on Service-Oriented Architecture (SOA) in the Field of Home Healthcare." Sensors 17.8 (2017): 1703.
6. Birrell, Andrew D., and Bruce Jay Nelson. "Implementing remote procedure calls." ACM Transactions on Computer Systems (TOCS) 2.1 (1984): 39-59.

7. Felber, Pascal, Rachid Guerraoui, and André Schiper. "The implementation of a CORBA object group service." Theory and Practice of object Systems 4.2 (1998): 93-105

8. Hérault, Colombe, Gaël Thomas, and Philippe Lalanda. "Mediation and enterprise service bus: A position paper." Proceedings of the first international workshop on mediation in semantic web services (MEDIATE 2005). 2005..

9. Bloomberg, J.: Events vs. services. Available at: http://www. zapthink.com, ZapThink white paper, October 2004

10. Rodriguez, Alex. "Restful web services: The basics." IBM developerWorks 33 (2008): 18.

11. Madakam, Somayya, R. Ramaswamy, and Siddharth Tripathi. "Internet of Things (IoT): A literature review." Journal of Computer and Communications 3.05 (2015): 164.

12. Pavithra, D., and Ranjith Balakrishnan. "IoT based monitoring and control system for home automation." 2015 global conference on communication technologies (GCCT). IEEE, 2015.

13. Zaiter, Meriem, and Salima Hacini. "Towards Exploring Context to Insure Fault Tolerance in Home Automation Medical System." 2018 IEEE 5th International Congress on Information Science and Technology (CiSt). IEEE, 2018.

14. ElHady, Nancy, and Julien Provost. "A systematic survey on sensor failure detection and fault-tolerance in ambient assisted living." Sensors 18.7 (2018): 1991.

15. Kodali, Ravi Kishore, et al. "IoT based smart security and home automation system." 2016 international conference on computing, communication and automation (ICCCA). IEEE, 2016.

16. Mohamed, Amin Salih, and Chiai Al-Atroshi. "Adaptability of SOA in IoT Services–An Empirical Survey." International Journal of Computer Applications 975: 8887.

17. Lan, Lina, et al. "An event-driven service-oriented architecture for internet of things service execution." International Journal of Online Engineering (iJOE) 11.2 (2015): 4-8.

18. Miorandi, Daniele, et al. "Internet of things: Vision, applications and research challenges." Ad hoc networks 10.7 (2012): 1497-1516.

19. Reddy, A. Anji, and S. Sowmya Kamath. "Research on potential semantic web service discovery mechanisms." arXiv preprint arXiv:1304.1676 (2013).

20. Pakari, Soodeh, Esmaeel Kheirkhah, and Mehrdad Jalali. "Web service

discovery methods and techniques: A review." International Journal of Computer Science, Engineering and Information Technology 4.1 (2014).

21. Stal, Michael. "Web services: beyond component-based computing." Communications of the ACM 45.10 (2002): 71-76.

KEY TERMS AND DEFINITIONS

Middleware: Provides an interface between two softwares to communicate over the network which are developed in different languages and residing on different platforms (Windows, UNIX, Linux etc.,).

BPEL: BPEL is a service composition technique used to compose services. A service comprising of several different services is known as a composed service and BPEL can be used to coordinate several services. Lower level services can be composed to form higher level services just like in an object oriented paradigm.

OWL: Web Ontology language is classification knowledge representation languages writing ontologies. It is developed to be used by applications that need to execute process for information instead of just presenting information to humans.

WordNet: It is a lexical database used by programmer to discovering and upgrading web services.

TCP: The entire web technology relies on HTTP which in turn relies on transmission control protocol (TCP) which is a part of the internet suite. TCP is a network communication standard that helps in a group of computers communicating with each other via data packets.

RESTful: RESTful services (also known as Representational State Transfer services) relies mostly on HTTP web infrastructure, although it is designed to work with any other networking protocol. RESTful services are stateless, provide a uniform interface, and are used for caching.

INDEX

Service-oriented Architecture (SOA) 1, 2 ,3, 4, 6, 7,8,9,10,11,12,15,16,17	Middleware 1,3,4,12,13
Internet-of-Things (IoT) 1, 3, 4, 7, 8, 11, 13, 15, 16, 17	Enterprise Service Bus (ESB) 1,3,6
Simple Object Access Protocol (SOAP) 1,2,6,7,9,10,17	

Universal Description, Discovery, and Integration (UDDI) 1,2,9,10	
Web Service Definition Language (WSDL) 1,2,6,9,10,17	
Business Process Execution Language (BPEL) 1,6	