An Evaluation of Ubiquitous Service Discovery and Remote Management in Pervasive Computing Environments

Minu Balakrishnan
Assistant Professor, Department of Information Technology,
Sri Eshwar College of Engineering, Coimbatore, Tamil Nadu, India.
minubalakrishnan@sece.ac.in

Article Info
Journal of Machine and Computing (http://anapub.co.ke/journals/jmc/jmc.html)
Doi: https://doi.org/10.53759/7669/jmc202202019
Received 05 March 2022; Revised form 15 May 2022; Accepted 30 June 2022.
Available online 05 October 2022.
©2022 Published by AnaPub Publications.

Abstract – Due to the rise of ubiquitous computing, issues around the discovery of new services have become a hot topic in the academic community. This is because futuristic ubiquitous computing frameworks as well as unsupervised Ad hoc systems will require service discovery. Automatically discovering network services and their characteristics, as well as dynamically advertising their persistence, are both possible with service discovery. Multiple service discovery technologies, including Service Discovery Protocols (SDP) runners, Universal Plug and Play (UPnP), Service Location Protocol (SLP), and Jini, have been suggested. In this paper, we detail the process of service discovery and control using Ubiquitous Remote Manager (URM), a system for remote and autonomous management of home gateways that allows for more system simplicity and finer grained command of the underlying network in a ubiquitous computing setting. Ubiquitous Service Discovery (USD) is a key component of the infrastructure of the smart homes of the future. USD's goal is to locate the most suitable service, given the user's preferences and needs, in a pervasive and extensive setting.

Keywords – Ubiquitous Service Discovery (USD), Service Discovery Protocols (SDP), Ubiquitous Remote Manager (URM), Protocol Data Units (PDU)

I. INTRODUCTION

Ubiquitous Computing, Quiet Technology, and Smart Things are all synonyms for widespread computing. This form of computing is all about being able to send any kind of data to any kind of gadget through any kind of connection. To sum up, it is all about pervasive computing [1], in which computers are built into everything from appliances to cars to chairs to even humans themselves. Ubiquitous Synonym for "pervasive," signifying ubiquity or widespread presence. Mobile or embedded sensors in anything from automobiles to tools to appliances to apparel and consumer products all communicate with one another using pervasive computing devices that are so small they are nearly invisible.A network's available services may be discovered using a process called "service discovery," which works by automatically querying each available node. When a user makes a service request, the term "service discovery" refers to the process of locating a supplier who can fulfill that need. Once the requested service's location has been obtained, the user may proceed to make use of it. To provide a highly adaptable infrastructure where customers may seek out specific services of relevance and service providers can declare and promote their capacities to the system is, ostensibly, the goal of service discovery technique. In addition to reducing the need for manual intervention, service discovery enables self-healing networks by automatically identifying decommissioned services.

Once services are found, networked devices may be remotely controlled if they all conform to a common protocol.Every protocol has its own "description language" that specifies the terminology and syntax to be used when describing the service and its attributes, making it easier to identify them. Methods for this job range from the key/value to the template-based and the semantically descriptive, depending on the desired level of expressiveness. Key/value analysis employs a system of attribute-value pairs to describe services. Similar to the first method, the template-based approach employs this same method, but also provides a collection of frequently-used properties that have already been created. Semantic description is ontology-based. In comparison to the previous two methods, it is more expressive. Techniques for service discovery aid users in locating networked resources such as software and hardware. This is a great feature for users who often switch networks, as well as those who frequently build spontaneous (Ad hoc) wireless networks and utilize their devices in a ubiquitous computing environment. To possibly locate a service, a user should first specify the sort of service they need (printer, for example) and then, if more than one service meets their criteria, they must use their discretion to choose amongst them. Service discovery strategies not only benefit the user but also provide the unexpected benefit of easing the burden on network administrators, particularly when introducing new services into a huge networking system. The scenarios in the paragraph below shows the applicability of service discovery:

A political conference is now underway, and a news reporter has made a trip to the press room set up at the convention center. The newsroom is equipped with a number of services, including a network point, fax machine, a scanner, and printer, which has a capacity to access the web. Firstly, the reporter must determine what kind of printer is available, and then she must update the driver that enables printing. By eliminating the need for manual configuration after the installation of a service discovery protocol, the user experience is greatly improved. Ubiquitous Service Discovery (USD) [2] represented in **Fig. 1** below is an architecture designed to facilitate widespread service discovery in a pervasive and mobile computing setting. It can function on WANs and LANs. Events, LBS, comms, and online services are all within the scope of what is being managed.

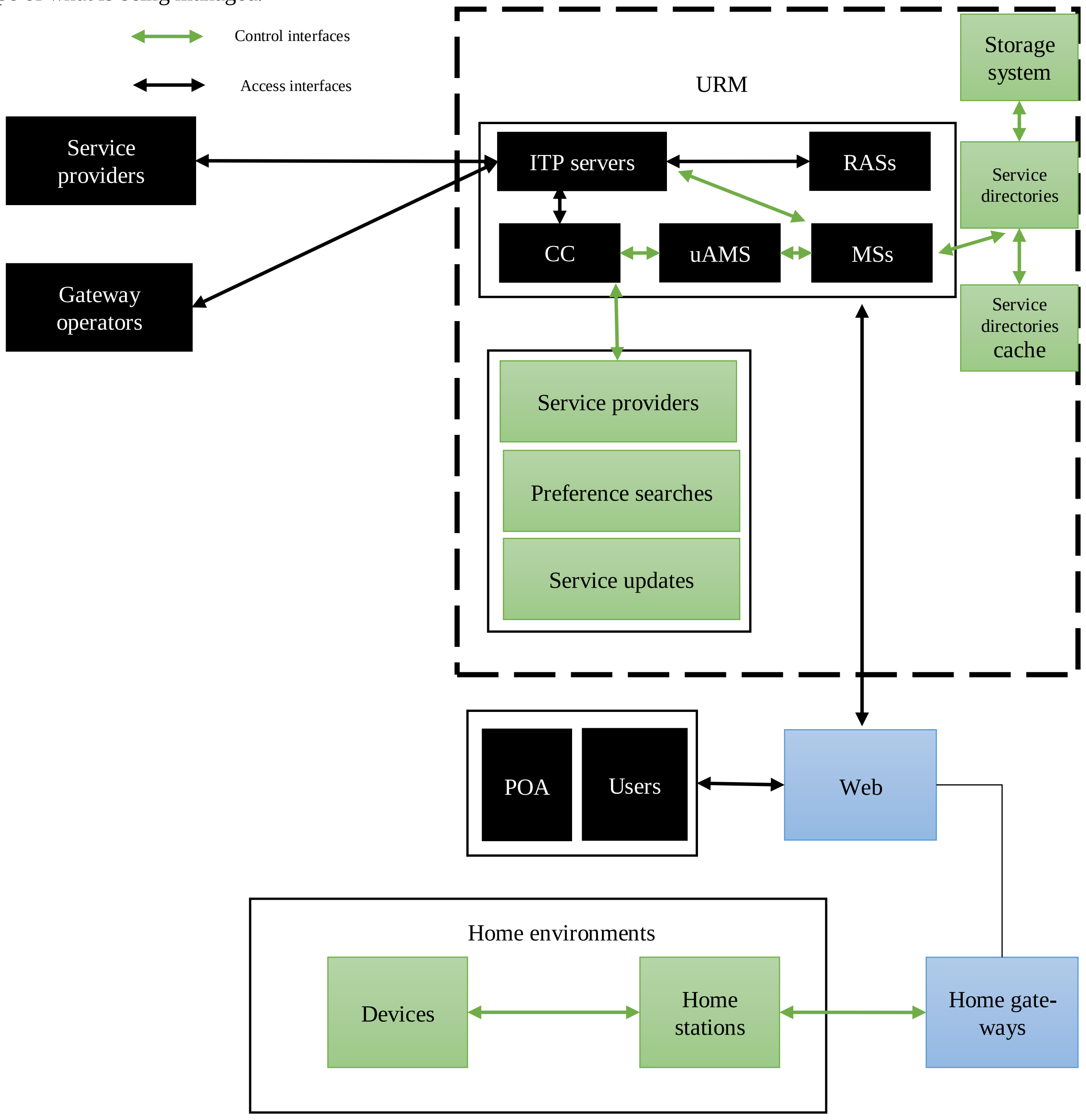

Fig 1. Architecture of USD within home environment

Section I has presented an introduction of ubiquitous computing and ubiquitous service delivery concepts. In this section, a USD architecture within a home environment has been mentioned for widespread service delivery in pervasive computing. The principle aim of this paper is to evaluation USD and URM within pervasive computing settings. The remainder of the paper has been organized as follows: Section II provides a critical review of previous works in ubiquitous computing. Section III introduces ubiquitous computing, and Section IV provides a comparison of a framework for SDPs within pervasive computing settings. Section V reviews pervasive computing technologies. Section VI reflects in ubiquitous management, introducing the concept of remote management and service discovery. Section VII draws final remarks and directions for future research.

II. LITERATURE REVIEW

Most service discovery approaches depend on the "publicize andlisten" in framework that applies multicast (periodically) for service discovery and announcement. Many organizations, groups, and consortia are working to solve this issue on many different fronts. There are several different Service Discovery Protocols (SDP) [3] in the works at the moment. Currently, JINI, SDP, UPnP, and SLP are the most often used.Developed by Sun Microsystems, Jini is a Java add-on. The Jini system expands the JavaTM application platform beyond a single virtual environment and into a distributed cluster. The Java application architecture is well-suited for pervasive computing because it facilitates the transfer of both programs and data across computers. The environment's built-in safety features make it safe to execute code that was downloaded from another computer. Java's strong typing features make it possible to determine an object's class before executing it on a virtual computer, even if the object was not created on that system. The end result is a network in which objects may freely travel throughout the network and access any node to conduct any action. The components of the Jini framework are as shown in **Table 1** below:

Table 1. Components of the Jini framework

Component	Brief description
Lookup service	This service tracks the various Jini applications and provides the essential proxies meant to enhance communications. Moreover, the service is also considered the Jini service.
Jini service	The Jini Service has been added to the lookup system and can be called using the service's public functionality, which is described as a Java remote functionality. Communication between Jini services is enabled by RMI, an underpinning system (Remote Approach Invocation).
Jini client	This is a piece of code that makes the inquiry to the lookup service for the proxy before calling upon the Jini Service.

Mechanisms enabling devices, applications, and clients to connect and disconnect from a network are provided by the Jini technology foundation. When interacting with a Jini system, joining and departing are simple, normal, and often even automated processes. When compared to traditional interconnected groups, where network configuration is handled centrally and manually, Jini systems provide far more flexibility. The Jini architecture's goal is to unite disparate hardware and software pieces into a unified, adaptable network. The resultant federation offers the benefits of a big, centralized system, such as easy accessibility, simplicity of management, and capacity for sharing, while maintaining the adaptability, consistent response, and control of a standalone PC or workstation. A unified Jini system is designed with the workplace in mind. The federation's participants are presumed to have a common understanding of trust, governance, id, and policy. Jini systems may be federated to form more robust networks for enterprises. Throughout the Jini architecture, the idea of a service stands out as the most central. The term "service" refers to any entity that may be used by humans, computers, or other services. Services may be anything from a calculation to storage to a communication route to an application filter to a user to a hardware device. Publishing a manuscript and converting it from one word processor format to another are two instances of services.

Jini federated systems [4] allow users to pool resources by pooling their membership. You can't think about a Jini system in terms of clients, servers, users, or even software and data files. Instead, Jini is built around a set of services that may be assembled as needed to accomplish a job. It is possible for one service to utilize another service, and for a customer of one service to also be another service with its own clients. By its very nature, a Jini system is highly adaptable, allowing for the inclusion or exclusion of services from a confederation at any moment in response to demand, necessity, or the evolving needs of the team making use of the system. Jini systems provide the means through which a distributed system's services may be built, located, communicated, and used. Equipment like printers, monitors, and drives, programs like apps and utilities, data like databases and files, and even the people who utilize the system itself are all examples of services. A Jini system's services talk to one another using a service paradigm that is a collection of Java APIs. These kinds of protocols may be used in any number of ways. A subset of these protocols, outlining crucial service interactions, is defined in the Jini system's foundational architecture.

Service discovery protocol (SDP) allows programs to learn about and examine accessible services and their attributes. The SDP protocol is designed to facilitate service discovery in a Bluetooth setting. It works splendidly with the ever-changing Bluetooth communications environment. Once a service has been found via SDP, there are a variety of different ways to interact with it, some of which are not specified in the SDP specification. Every transaction in SDP is represented by a pair of Protocol Data Units (PDUs): a request PDU and a response PDU. When SDP is integrated with Bluetooth L2CAP procols of discovery, only a single request PDU for each interconnection to a particular SDP server might be evident at any one time. The set of accessible services in a Bluetooth ecosystem is dynamic, depending on the RF vicinity of moving devices; hence, a service discovery mechanism tailored to the Bluetooth ecosystem is required.

The Bluetooth standard defines a service discovery mechanism that is tailored to the Bluetooth ecosystem. In terms of the underlying communication, SDP has few needs because of its simplicity. Over a secure packet transport, it is possible to operate (or even more unreliable, in case the client applies repeat requests and timeouts as needed). Each transaction in

SDP is represented by a pair of Protocol Data Units (PDUs) [5]—a request PDU and a response PDU. Nevertheless, queries may be routed via a queue, and replies may be sent back out of sequence. SDP operates on a request/response basis as shown in **Fig. 2**, with one query PDU and one feedback PDU making up a single transaction. When using SDP with Bluetooth L2CAP protocol, only a single request PDU might be pending in every inter-connection to a particular server at any one time. To rephrase, a client must wait for an answer to each query before sending another across an L2CAP link. By preventing SDP from transmitting more than one rejected request PDU, a basic form of channel control may be achieved.

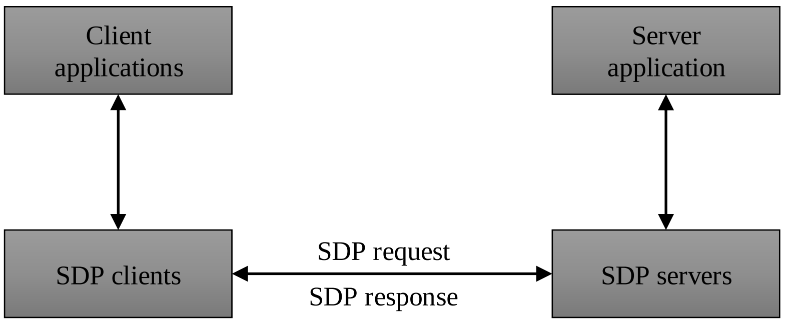

Fig 2. Courtesy of Bluetooth SIG, SDP Specs

Networked devices including PCs, printers, gateways, Wi-Fipoints, and cellular phones are able to automatically find and connect with one another using a set of protocols known as Universal Plug and Play (UPnP). When it comes to networking, UPnP is mainly designed for home networks that don't need enterprise-grade equipment. The UPnP Forum is a group working to standardize and improve the methods by which computers and other electronic devices may communicate with one another and with other devices. At the Forum, over 800 companies participated, covering everything from consumer devices to cloud computing. Starting in 2016, the Open Connectivity Foundation (OCF) has been in charge of all UPnP-related activities [6]. Assuming an IP-based network, UPnP builds on top of IP in order to provide descriptions for devices or services, operations, data transmission, and event notification. By layering HTTP over UDP (port 1900) and using multicast, it is possible to allow device discovery requests and ads (known as HTTPMU). Similarly, search-result responses are delivered through UDP, however they are sent as unicast (known as HTTPU). When it comes to home and small office wireless networks, UPnP is essentially a zero-configuration version of UPnP, a technology that allows devices to be attached to a computer on-the-fly. When a UPnP device is plugged into a network, it will automatically configure itself to operate with any other UPnP devices already on the same network.

The multi-hop foundation helps to make UPnP chatty, spending too many system resources on channels with a huge number of gadgets; streamlined access control systems do not even map well to dynamic situations; and UPnP does not offer a unified setup syntax, e.g., that of CLI environment of Cisco IOS/JUNOS, making it inappropriate for implementation in corporate settings for purposes of analysis, variability, and continuity. Microsoft's UpnP protocol allows plug-and-play functionality to be applied to any device that can be accessed via a TCP/IP network. There is no centralized service registration as in SLP, making it ideal for use in smaller business or home computer networks. The UPnP discovery system permits the gadget to promote its features to other network nodes. Similarly, the UPnP protocol allows control points to effectively scan networks for essential networking devices once it has been added.

The Service Location Protocol (SLP) [7] is a prospective standardized protocol established by IETF (Internet Engineering Task Force) in order to effect easier network resource identification and use. This service discovery protocol is distributed, compact, scalable, and expandable inside a given site. It provides for central administration but is not necessary for it to function. Users, particularly mobile users, won't be able to take use of network services to their full potential if setup continues to be a pain.

Table 2: Parts of SLP

Parts of SLP	Brief description
User Agent (UA)	This part provides support to the query functionality. It obtains service data for different applications. The user agents have the capacity to retrieve service data from directory agents and service agents. Hosts-on-demand users are some of the examples of this part.
Service Agent (SA)	This part advertises the locality and features of services in absence of a service.
Directory Agent (DA)	This paper gathers service data from agents in order to generate repositories of service datasets meant for its centralization for ease of access by different agents involved. There might be just a single DA available in a particular host.

SLP is a proposed standard by the IETF that would allow clients to be automatically configured and informed about new services available on the network. It allows for completely decentralized management and can expand from a small, unmanaged network to a large, enterprise-level one. SLP's primary purpose is service discovery, but it also has the capability of working with IPv6 and extending security to browsing activities. The three parts of SLP were as shown in **Table 2.**

III. UBIQUITOUS COMPUTING

Researchers in [8] have been exploring the possibilities of ubiquitous computing since at least 1993. The term "ubiquitous computing" refers to a technique for maximizing computer utilization in which users have constant, seamless access to various computers located in different parts of the physical environment. Users and their applications are crucial to the concept of ubiquitous computing. Different from the traditional computer paradigm, these environments include a physical place where people and the space may interact and share data using the environment's dedicated hardware and software. The broad availability of low-cost computer equipment and wireless networks has made this kind of thing possible. In a distributed computing environment, users are free to switch between the several available machines rather than being limited to a single workstation. An adaptable layout is essential in an environment where the same space serves many functions at distinct timeframes. Consideration e.g., current group size and the nature of the event should inform the layout.

In ubiquitous computing environments, users and various sensors will soon be sharing the same resources. Because of its key role in the system's authorization process, contextual data, e.g., location of applications and sensors they enable, must be handled securely. However, the broad nature of today's applications and the method in which users interact with a pervasive context provide novel security problems for conventional user-password methodology to computer security. Within a vibrant ubiquitous computing environment, e.g., the prevalence of more heterogeneous gathers and cellular phone users, accessibility to authorized individuals is a fundamental security requirement for ensuring the safety of users, confidentiality of data, as well as its availability and integrity. Though it is still in its infancy stage, study works on the application and security of ubiquitous computing is thriving. Several articles have looked at what kind of security measures are needed for pervasive computing. In [9], we imagine a flexible, multi-functional space that can be reconfigured to meet the needs of different users.

Role-centric access control methods are employed for easier administration to involved users and permission, and they are utilized to establish and apply accessibility control protocols for various key configurations of the operational area. A scenario is used to illustrate how context data may be used to dynamically allocate permissions to roles and the individuals responsible for doing so. An attempt is now being made to build a secure architecture for the GAIA operating system. By freeing the user's viewpoint from the physical limits of a traditionally distributed system, the architecture allows for the dynamic and flexible handling of security challenges in a computer system with hundreds of processing units. Thanks to a method described in [10], users' private information is safe, and their conversations are concealed from prying eyes. This protocol creates a "MIST" to conceal a user's identification from systems as well as other users, allowing unrestricted interaction with the diverse set of cloud-based services and businesses.

Safi, Luo, Pan, Liu and Yan [11] examines a payment prototype's potential in the realm of pervasive computing. In order to decide whether or not to enable clients to pay with an e-purse, a computational model of trust is developed for usage in ubiquitous contexts. The authors have developed a method (and a prototype of it) that reduces the risk of privacy leakage in this way without prohibiting the use of trust for easing payment by allowing users to segment trust based on the environment. When it comes to implementing ubiquitous computing, there are various security concerns that must be resolved. In particular, the criteria illustrated in **Table 3**should be met for effective designing of access control architecture that is both reliable and adaptable.

Table 3. Criteria to be met in designing access control architecture

Criteria	Brief description
Access based on contextual information	Authorized user or topic accesses to resources or objects are often created in the context of conventional disseminated as well as multi-user networks in current access approaches. Unlike in ubiquitous setting, where objects and users are free to come and go as they want, traditional systems have fixed users and things. Access to the IEEE digital library, for example, requires a membership number and password, but in pervasive contexts, such safeguards may not be in place. There does not seem to be a means of conveying any extra context-based factors, which ought to be considered when permission is granted.
Access control in a collaborative environment	Collaboration services, in which several people work together to accomplish a single goal, are a common use of ubiquitous computing. For collaborative tasks involving several, potentially unreliable users, an access control model in a pervasive environment must provide a safe framework for permissions to be distributed and managed. Tony, the project manager, may choose to delegate some or all of his authority to Christine, a team member.
Decentralized administration	When it comes to ubiquitous computing, a physical space is often a part of a much wider system, and the policy for regulating who has access to what may be established by the managers of both the framework and the environment.
Access for different users and devices	Ubiquitous computing is not limited to purely technical uses. Customers have, for instance, been subjected to safety checks by airport security systems without their knowledge of the nature of such systems or how they function. Users in a ubiquitous setting may not need extensive training and instead prioritize gadgets that are intuitive to use.

The last criterion deals with the interaction between humans and computers, a topic that is beyond the scope of this work. We zero down on those last three necessities. This article introduces authorization models that use control to regulate who may enter a certain area, as well as secure infrastructures for carrying them out. To determine whether or not a certain subject is permitted to use a set of resources, the standard approach to access control analyzes the authorization decisions that have been made on their behalf. All subjects have certain responsibilities that must be met before they are granted access to certain resources. Conditions are criteria that must be met regardless of the relationship between the subject and the object. The duties and requirements of novel hosts are decision considerations for access control in contexts with ubiquitous computing since consumers and items are both very dynamic.

Decisions about granting access are often made at the moment a request is received, without taking into account any restrictions for limiting or revoking access over the long term. The gadgets and other things in the ubiquitous computing environment are utilized by many people. There are complicated connections among consumers, items, and authorization among different objects and consumers. Objects can be devices, or documents or files, and so forth. Users must comply with responsibilities, meet requirements, and submit to continual control with varying security policies due to the pervasive and complicated nature of the environment. It is widely agreed that use control is the most promising new approach to efficient security management. Obligation, requirement, as well as authorization are employed in designing of a more secure infrastructure. The prevailing control allows for a vibrant access verification for more contextual dataset.

IV. COMPARISON ANALYSIS FOR SDP FOR PERVASIVE COMPUTING SETTING

Various research difficulties arise in the area of service discovery networks to the very dynamic nature of ubiquitous computing contexts. Some of the study problems have been solved, while others have gotten little or no attention. The purpose of this subsection is to provide a comparative paradigm for service discovery algorithms used in ubiquitous computing settings. **Fig. 3** depicts the overall design of the framework. Our hope is that this paradigm will help researchers focus on the problems that need to be solved by discovery protocols before they can be effectively implemented in ubiquitous computing settings. In the sections that follow, we'll break down each of the framework's identified problems.

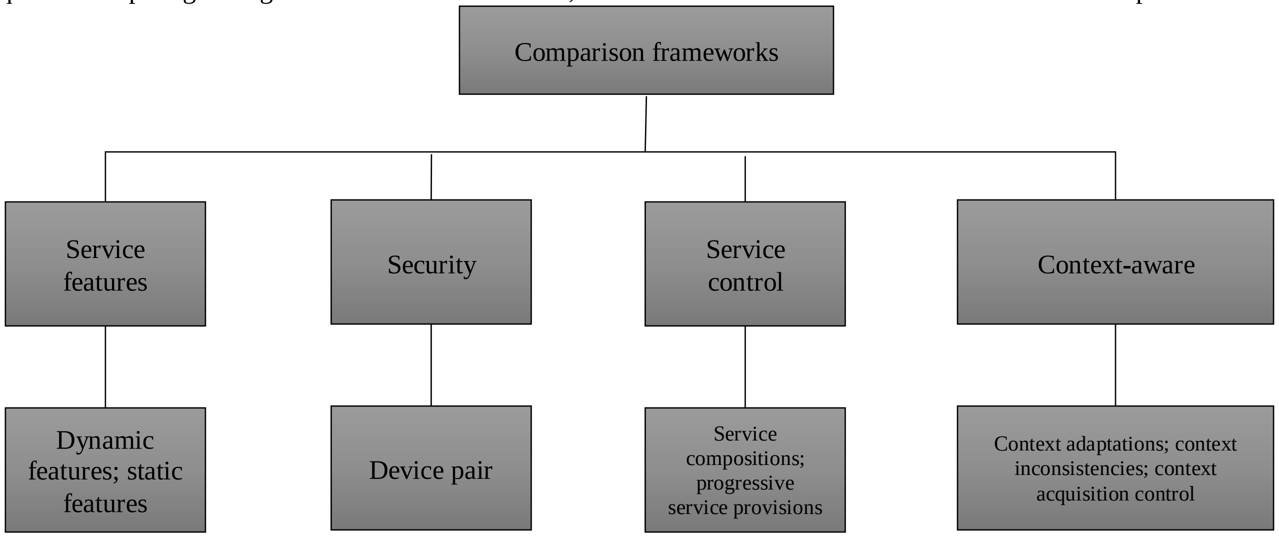
Comparison frameworks
Service features
Security
Service control
Context-aware
Dynamic features; static features
Device pair
Service compositions; progressive service provisions
Context adaptations; context inconsistencies; context acquisition control

Fig 3. Frameworks of comparison for discovery protocols within a pervasive computing ecosystem

Context-Awareness

Mark Weiser, in his landmark article, outlined a vision for ubiquitous computing that relies heavily on contextual awareness. Several descriptions of contexts, and based on extensions, context-awareness could be identified in previous research. However, the most often cited definition is provided by authors in [12], who provide a definition of data, which could be used in illustrating situations of an object. When discussing the relationship between a consumer and a program, it is important to distinguish between the user and the program as separate entities. Moreover, context-awareness is defined as "a feature of a system that makes use of contexts to provide essential services as well as data to respective users, whereby relevance depends on the works of users" Contextual-aware resource discovery (also referred to as contextual restructuring) and contextual adaptations are two context-aware forms that have been recognized and studied by scholars. These are examples of context-aware capabilities that allow applications to modify their actions based on their current environment. Realizing Mark Weiser's vision of omnipresent computing depends critically on these two types of adaptability. So far, the protocols that have been described do not allow for contextual modification.

1) Context Management, Acquisition and Inconsistencies

The method by which contextual data is extracted from the surroundings and used is a difficulty for context-aware systems. The first step in context-aware service discovery is acquiring relevant context. Contextual data may be gathered in a few different ways: via the use of sensors, through prediction, or by analysis of a user's profile and past interactions with a service. Thus, in ubiquitous computing, discovery protocols must include support for context collection and processing. Nevertheless, owing to factors like faulty sensor technologies or inadequate sensor technologies (for example, RFID scanners only detect roughly 60%-70%), Context might become cluttered and inadequate and consequently inconsistent. An irregular environment may result in the identification of unrequired services or make systems to fail to identify specific services irrespective of the fact that the needed one is there, both of which are problems with context-aware discovery. For applications to run smoothly, context inconsistency must be resolved through discovery mechanisms.

Contextual Adaptation

Adapting a service's behavior based on its surroundings is what we mean when we talk about context-aware adaptation. SDP should offer adaptation support to help meet user needs in a way that causes as little disruption as possible, given the fluidity of pervasive computing contexts (such as variable network capacity and heterogeneous nature of pervasive gadgets of machine parameters (such as low battery power and small screen size). The found service that provides video material, for instance, may be modified to deliver a textual rendition of the information in response to change in bandwidth. For example, a basic house lighting setup may have the lighting service set to a user-preferred setting according to individual's present operations (e.g., watching television, reading or snoozing), while a more complex setup could have the TV volume automatically lowered if a user is using a cellular phone. Numerous proposals for network communication protocols (context-aware) designs have been made, all of which make use of context to locate the most relevant services but fail to provide mechanisms for adapting to changing circumstances.

Service Management

Mobility, node failures, and poor wireless connection are just a few of the many factors that might disrupt service delivery. There are two main categories of mobility: (1) mobility of devices, and (2) mobility of service codes. As devices move throughout the network, they disconnect from services, and as service codes move around the network, they no longer have any ties to the clients who originally used them. When a wireless connection is poor, either the service provider or the consumer might be cut off. To further complicate matters, a user's goals may need the provision of a composite service, which is a mix of many network services (service composition). Service composition is an integral part of our service management framework. So, the two main activities that make up service management are creating new services and ensuring that existing ones keep running well. To facilitate service composition and ensure ongoing service supply in the face of the dynamic nature of pervasive computing settings, protocols of service discovery ought to incorporate methodologies and infrastructures for effective management of services.

Service Compositions

In order to provide a high-level service, which potentially accomplishes the users' requirements, it is necessary to first identify and acquire the constituent services. Sometimes, the discovery framework needs to find and combine the right services to make up a unique package that meets the needs of the user and allows them to complete their task without any hitches. The client-requested service cannot be formulated without the proper mechanism for service composition, which can only be provided by service control support.

Continued Service Provisioning

One of the main goals of ubiquitous computing is to provide the user the tools she needs to go about her day without much interference from her technology. In order to realize this objective, it is crucial that service delivery is managed effectively and that service availability is assured at all times. Moving devices, changing service codes, losing a lease, experiencing hardware issues, or having software crash are just some of the various disruptions that might occur in the course of delivering continuous service. The delivery of continuous services is severely hampered by the frequent occurrence of invalid bands. The bands are viewed as expired whenever pointers to the interlinked services are no longer resolvable. It is possible for bindings to become invalid for a variety of reasons, such as when a service that is part of a binding is moved across the networks for load balancing or whenever it is shifted towards a firm, which employs that band to enhance service delivery and preserve its bandwidth. In case a service is moved to a new location, all previous references to that service become obsolete. Service management support, in order to perfectly maintain a reliable and invisible service giving process, must use methods to (1) mitigate the problems of ineffective binding, and (2) executing more proactive hunts for substitutes whenever services are considered unreachable because of devices' mobility.

Security

Security on the SDP is complicated by the transient clients' nature as well as service in the realm of ubiquitous computing. Data exchange between clients and services in a foreign environment requires extra precautions. Access control, encryption, and cryptographic technologies are some of the current methods used to guarantee security. In order for two devices to exchange data with one another, they must first establish a connection utilizing some kind of wireless technology (e.g., Bluetooth or WI-FI). Pairing, often called association, is the process of connecting two nearby devices. A personal digital assistant (PDA) and a smart screen are a good example of a gadget coupling. Given that wireless is used for communication, several attacks, such as denial-of-service (DoS), man-in-the-middle (MiTM), and others, may be launched against the pairing process. Since there is a massive device in the marketplace in terms of their wireless communication capabilities, computational capacity, and sensor technologies, it is challenging to offer a universal remedy for safer pairing. Prevailing discovery methodologies guarantee some security, but this topic requires further research, and robust security support should be offered.

Dynamic Service Characteristics

Services can be described by their features. Several service attributes may be used to characterize a printing service, such as the printer's model, its current health, its physical location, etc. The service registrations or promotes itself by highlighting its unique set of features that may be used to locate and learn more about it. Due to their transient nature, environmental services often undergo a redescription. To be found in accordance with its new location, music player services rendered by PDA with “hotel” property will need to have its location attribute updated to reflect its new setting. In particular instances, it might be essential to transform service characteristics other than location. To provide just one example, the printer service has an element called "print jobs" that specifies how many printouts have been requested. A printer's queue size might fluctuate depending on how many print jobs have been sent to it and how many have been completed. To do so, you must update the printer service's service property. Providing up-to-date information for choosing the best service sometimes necessitates a dynamic change in the service definition via its qualities. Only when service characteristics can be defined as dynamic attributes can these adjustments be made. Consequently, SDP must offer support for vibrant characteristics to enable the dynamic updating of service attributes, as well as a means to monitor the state of the service (for instance, the number of printouts at the printing press) and other service characteristics (for instance, the placement of the mp3 player delivery).

V. PERVASIVE COMPUTING TECHNOLOGIES

Usage Control

The concept of usage control is concerned as the framework of the future. The eight parts of the use control model are as follows: subjects; subject characteristics; objects; objects' characteristics; rights; approvals; duties; and circumstances. This article uses the terms "subject" and "item" in their conventional meaning, as they are commonplace in conventional access control. Reading and writing are examples of rights since they reflect the ability of a person to interact with an object. When the subject makes an effort to use the resource, we may verify whether or not we have the authority to do so. Subject characteristics, object attributes, authorizations, responsibilities, and circumstances are taken into account by the use decision functions shown in **Fig. 4** to reach this conclusion. Attributes of subjects and objects may be taken into account while making access determinations. Identity, group name, role, membership, credit, etc. are all examples of subject characteristics. labels, shareholding, subclasses, access control lists, etc. are all examples of object attributes. A book like Harry Potter may have different prices for different use rights in an online store, such as $20 for a read right and $1,000 for a resale right.

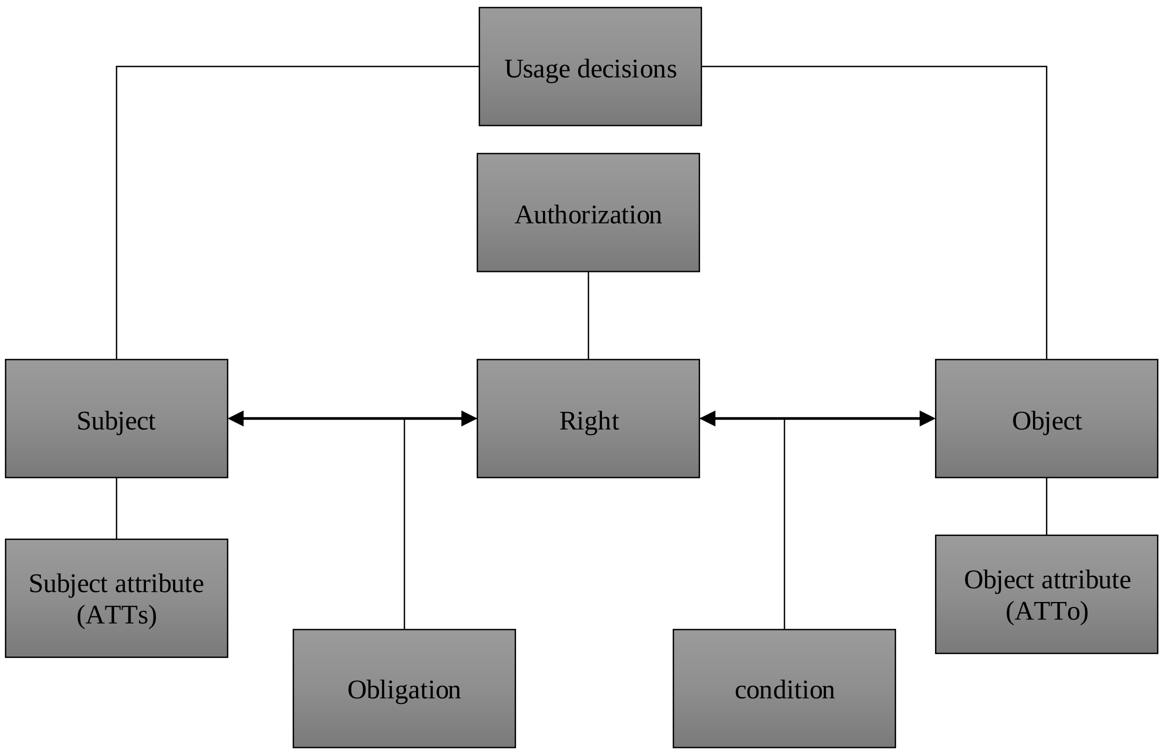

Fig 4. Model components

Decision functions consider elements like authorizations, duties, and conditions while deciding whether or not to provide a subject access to an object. Authorizations are determined by the subject, object, and right combination. While authorization is typically needed prior to access, it is also possible to need it while access is in progress; for example, a certificate revocation list (CRL) could be checked at regular intervals. When a certificate is found on the CRL, it immediately nullifies the associated access. Subject and object properties may need to be changed if an authorization is to take effect. **Fig 5.** Depicts three types of updates, labeled as "prior," "during," and "post," all of which have continuity properties. Prepaid mobile phones, for instance, necessitate updating the users' (subject') clearance before entry (prior-properties) and also require more periodic updating of users' (subject') remaining credit during utility ('during' properties), with possible cancelation in cases of over-usage. Reports to the usable capacity on a fixed phone billing system are required after each use. Subjects have obligations that must be met either before they are granted access or at any point during their use of the system. An instance of pre-obligations is the need for users to register with their name and email address before gaining access to the IEEE Digital Library. One example of a persistent duty is the precondition that a user keeps particular advertising curtains open while using a given service. Attributes of the subject and the object can be used to determine the nature of the responsibilities necessary to grant access.

Environmental and system-oriented constraints are taken into account as conditions. For instance, full publications in the IEEE online library are available to IEEE members. System security circumstances such as reduced rate, standard, high alert, etc. can also be conditions. As we have seen in **Fig 5**, continuity is an additional consideration that must be made when making a choice. The standard model of access control presumes that users have been properly authorized before granting them access (prior). Due to the ever-shifting nature of context in pervasive computing settings, it makes sense to extend this for constant policing through the evaluation of usage prerequisites at each touchpoint (during).

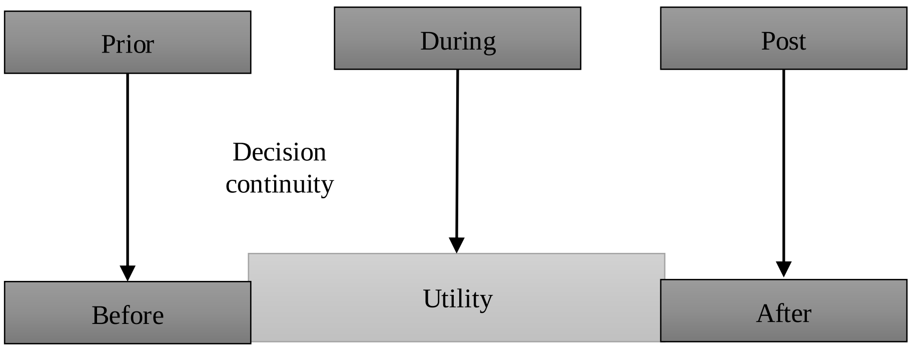

Fig. 5: Continuity features

Ubiquitous Computing Model

Use control relies on the notion of an object's access right (or "usage") as its foundational idea. When users enter a designated area outfitted with pervasive computing infrastructure, they are automatically given a set of privileges that determine how they may interact with the space's many services. Items in ubiquitous settings might be anything from services to devices to dynamically formed objects. For instance, the set of actions that users may apply to a device or service is defined by the access privileges to that device or service. In this paper, we present the idea of a space object and the space rights that accompany it. Simplifying the job of the space administrator, access control rules are specified by space objects and rights. Ultimately, the goal of ubiquitous computing is to build a system that is focused on the needs of the user and the tasks they need to do.

Because of this, we focus on people and software in these ubiquitous settings. Accounts for users wishing to access the space are established by system administrators, who then allocate use of access space objects to those users in accordance with their obligations. Whenever an individual with a "use" incorporates a "space" they are immediately given a space object that has just the rights, which make sense within a certain space. In **Fig. 6**, we can see the process by which individuals are allotted certain space utilizations. A space manager, for instance, can designate a certain area of the building as the "interview room" and restrict access to that area to those who work in the HR department. An HR officer may have access to the whole system in general, but when it comes to using the interview room, only those permissions are granted that are directly linked to the work at hand. To put it another way, the rights granted to an item in a space are typically a subset of the rights, which are given to systems as a whole.The application's usages are utilized to define its security rules. The space manager converts these application metrics into space metrics. A job interview application, for instance, may be used by both the interview committee and the applicants, as illustrated in **Fig. 7**. These two purposes are determined by the roles of various application participants. Candidates must have access to a presenting device, and members of the committee must be able to view the slides. In order to use the application, a candidate needs access to the various devices in the room. Access controls within spaces are solely implemented in relation to these space usages, where applications and individuals involved are assigned by the space allocator to suitable objects and their accessibility privileges. Our model's security zones are defined by several authorisation methods. When a space changes modes, it may be possible for new objects to be produced on the fly and given permissions automatically depending on the space's context.

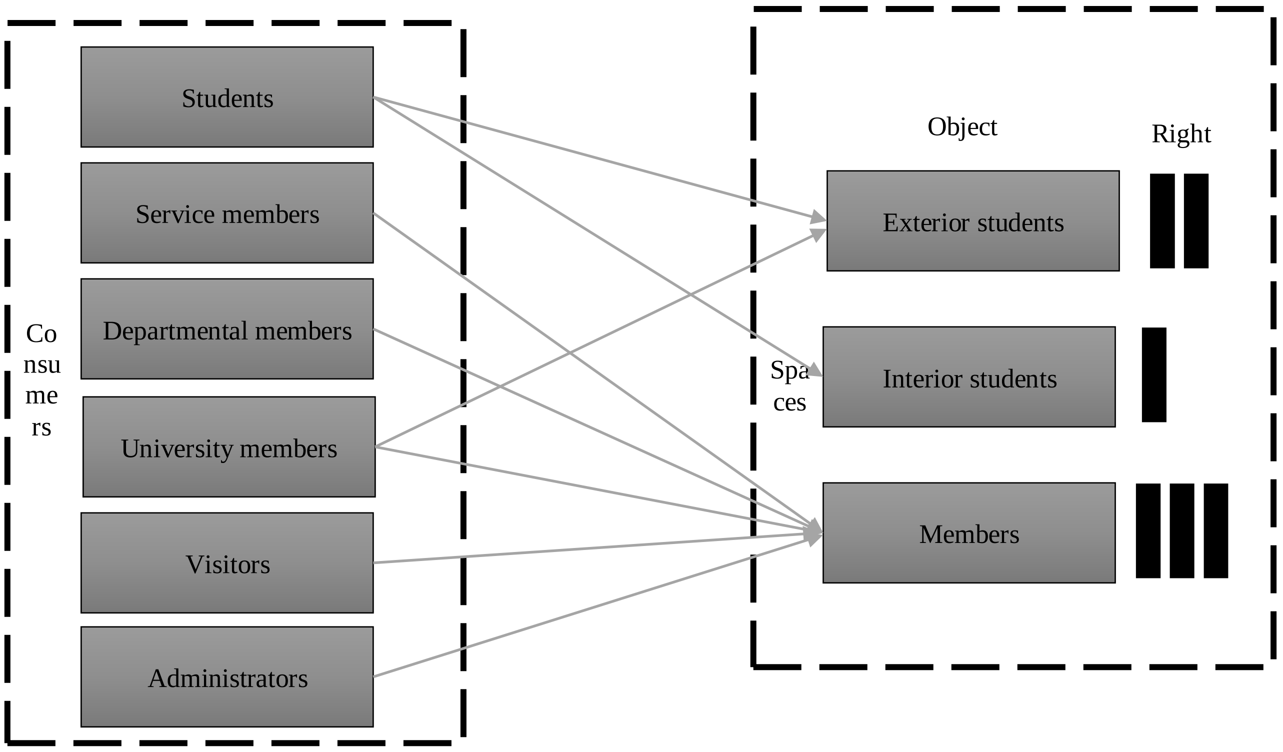

Fig 6. Users-spaces usage assignment

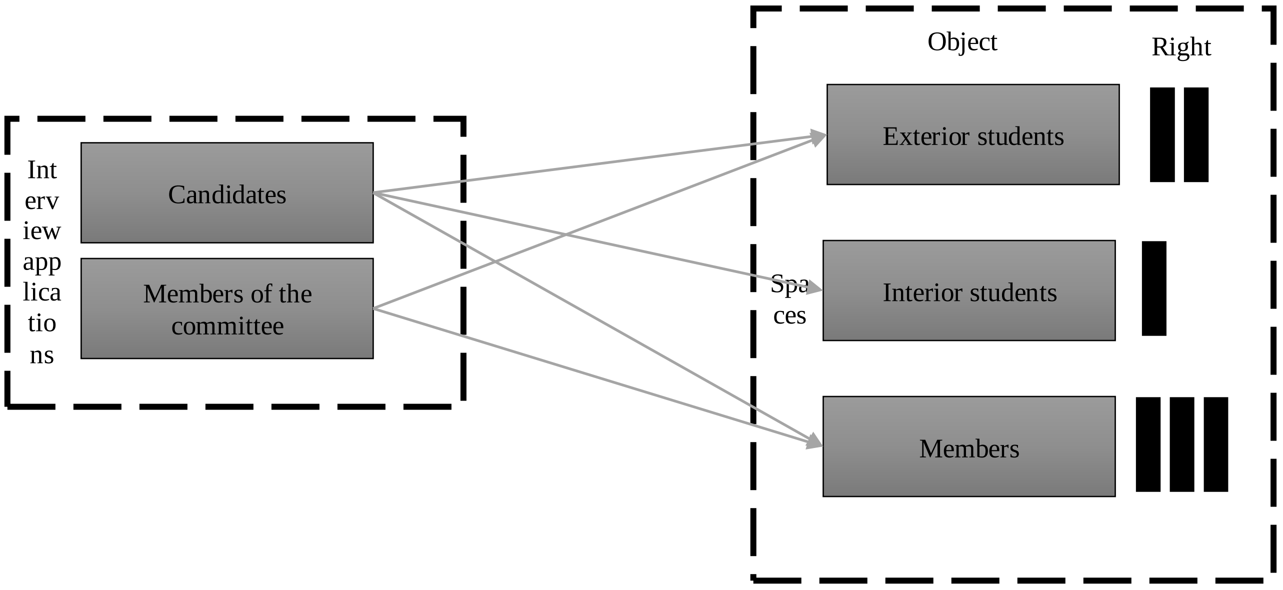

Fig 7. Applications-spaces utility assignment

VI. UBIQUITOUS REMOTE MANAGEMENT AND SERVICE DISCOVERY

Remote Management

The USD has been established for URM (Ubiquitous Remote Manager). By remotely and autonomously managing the home gateways, URM reduces system complexity and increases network control. More useful features, such as the U-Bid Services, U-Service Advertisers, U-Service Compositions, U-Device Advertisers, etc., have been included into URM. It will be impossible to effectively and efficiently handle all of these services on your own. To facilitate this, an autonomous self-management function has been implemented, and crucial components for URM are integrated as illustrated in **Table 4**. Because of this enhancement, the system may run autonomously, meaning little intervention from humans is required.

Table 4: Crucial components for URM

Control center	When setting up a system, the Control Center (CC) component is crucial. It's a group of URM interconnection. When it comes to making changes to the system's databases, host machines, etc., only the command center has access. The control unit is always the starting point for setting up a new system. The CC is no longer required after initial setup is complete. Nevertheless, the Control Center (CC) is not the sole means by which adjustments can be made to the system's settings.
Management server	A management server coordinates tasks such as sending messages to and receiving responses from service gateways, tracking the status of their setup, and launching and tracking the progress of management activities. The management server stores the configuration data for all gateways under its command in the centralized database, along with the management tasks that are queued up and waiting to be performed. Every management server must keep the database and the current configuration of service gateways in sync. As well as allowing OSGi frameworks to be installed on devices without requiring a large amount of memory, the server also provides support for retrieving as well as storing data from gateways on backend service (flash RAM, disk).
Remote access server	URM may be accessed remotely by both the service operator and the service user. Service providers use RAS to publish and manage their services. The URM's features and resources are available to anybody who wants to use them.
Ubiquitous autonomic management server	Services in JAR format will be hosted on the Ubiquitous Autonomic Management Server (UAMS) element of the system. All other programs and services in the network have full access to these files and may download them as needed.
HTTP server	The Ubiquitous Remote Manager relies on the HTTP Servers to maintain connections both outside and within the system. URM is connected to by Gateway Operators and Service Providers through HTTP Server.

Ubiquitous Service Discovery

The core tenet of Ubiquitous Service Discovery (USD) is that "service" refers to anything that may be consumed by a user, a computer program, or any other service [13]. In computing, a service could be a storage, connection to a different user, calculation, application filter, framework element, or even a different user. Services include the ability to print a document and convert between several word processor file formats. USD is a set of protocols and tools for finding and using services across a network. Printers, monitors, and disks are hardware; programs like word processors or spreadsheet editors are software; database and documents are information resources; and network operators are forms of services. An integral part of the USD framework is the Service pointer, which is responsible for identifying and exposing networked services to end users, as well as the Service Update Service, which is in charge of keeping the USD framework's service directories and cache fresh. **Table 5** represents a more definition of the integral parts of the USD framework.

Table 5. Integral parts of the USD framework

Component	Brief description
Service pointer	Services are identified and managed using this component. This component acts as the primary interface between the program and its end users. When a user makes a request, the service pointer retrieves it from the network's databases and any local caches that may have it.
Service update service	This service catalog and cache are maintained by the Management System, and are updated whenever a novel service is added by a Provider through URM. Services Advertise also allows for the addition of such offerings.
Preference search service	Using the Preference Search Service, users may make more nuanced service selections based on features they can really understand. To refine your search, you may use filters e.g., type of service and provider.

VII. CONCLUSION AND FUTURE RESEARCH

From this paper, we understand that technology that seamlessly fits into our daily routines is known as pervasive computing. As a result of its integration, the user no longer notices it. The term "pervasive computing" refers to the integration of mobile computing, wireless networking, and the World Wide Web. For ubiquitous computing operations to automatically and contextually provide users with relevant content, service discovery is a must. This contribution provides a comparison aspect for service discovery techniques and use it to assess many Service Discovery Protocols (SDP). While designing a discovery system for ubiquitous computing environments, we have identified various gaps that need to be filled by future research. Contextual inconsistencies and adaptability, service administration, variable service feature support, and device pairing security are all examples of these problems and gaps. In this paper, we provide Ubiquitous Service Discovery (USD), a scalable architecture for discovering services that may be used in both broad and local networks. Automatic and complex queries may be managed with the use of the service pointers and the Preference Search Service. Queries may be run either by hand or by an automated system.

References

[1]. "Support for context-aware pervasive computing environments", IEEE Pervasive Computing, vol. 10, no. 3, pp. 52-53, 2011. Doi : 10.1109/mprv.2011.57.
[2]. A. Giridharan, "Validation of Context Based Service Discovery Protocol for Ubiquitous Applications", International Journal of UbiComp, vol. 3, no. 4, pp. 19-34, 2012. Doi : 10.5121/iju.2012.3402.
[3]. A. Mian, R. Baldoni and R. Beraldi, "A Survey of Service Discovery Protocols in Multihop Mobile Ad Hoc Networks", IEEE Pervasive Computing, vol. 8, no. 1, pp. 66-74, 2009. Doi : 10.1109/mprv.2009.2.
[4]. C. Zhang and H. Pung, "The design and implementation of a Jini/Java-based A/V stream control and management", Multimedia Systems, vol. 9, no. 4, pp. 315-326, 2003. Doi : 10.1007/s00530-003-0058-7.
[5]. A. Hamed, M. El-Kharashi, A. Salem and M. Safar, "Two-Layer Bus-Independent Instruction Set Architecture for Securing Long Protocol Data Units in Automotive Open System Architecture-Based Automotive Electronic Control Units", Electronics, vol. 11, no. 6, p. 952, 2022. Doi : 10.3390/electronics11060952.
[6]. B. Miller, T. Nixon, C. Tai and M. Wood, "Home networking with Universal Plug and Play", IEEE Communications Magazine, vol. 39, no. 12, pp. 104-109, 2001. Doi : 10.1109/35.968819.
[7]. S. Kaushik and R. Poonia, "Evolutionary Study of Service Location Protocol", SSRN Electronic Journal, 2018. Doi : 10.2139/ssrn.3166505.
[8]. K. Swan, M. van ‘T Hooft, A. Kratcoski and J. Schenker, "Ubiquitous Computing and Changing Pedagogical Possibilities: Representations, Conceptualizations and Uses of Knowledge", Journal of Educational Computing Research, vol. 36, no. 4, pp. 481-515, 2007. Doi : 10.2190/b577-7162-2x11-17n5.
[9]. T. Smolinski, C. Soto-Treviño, P. Rabbah, F. Nadim and A. Prinz, "Systematic computational exploration of the parameter space of the multi-compartment model of the lobster pyloric pacemaker kernel suggests that the kernel can achieve functional activity under various parameter configurations", BMC Neuroscience, vol. 8, no. 2, 2007. Doi : 10.1186/1471-2202-8-s2-p164.
[10]. C. Swanson and D. Stinson, "Extended results on privacy against coalitions of users in user-private information retrieval protocols", Cryptography and Communications, vol. 7, no. 4, pp. 415-437, 2015. Doi : 10.1007/s12095-015-0125-x.
[11]. Q. Safi, S. Luo, L. Pan, W. Liu and G. Yan, "Secure authentication framework for cloud-based toll payment message dissemination over ubiquitous VANETs", Pervasive and Mobile Computing, vol. 48, pp. 43-58, 2018. Doi : 10.1016/j.pmcj.2018.05.004.
[12]. "Context Awareness in Data Mining Applications", International Journal of Science and Research (IJSR), vol. 5, no. 1, pp. 253-255, 2016. Doi : 10.21275/v5i1.nov152619.
[13]. F. Palmieri, "Scalable service discovery in ubiquitous and pervasive computing architectures: A percolation-driven approach", Future Generation Computer Systems, vol. 29, no. 3, pp. 693-703, 2013. Doi : 10.1016/j.future.2012.08.004.