Chapter

Connection Operations

And

Service Discovery

Content of Lecture

1.1 Introduction

The terms **disconnected operation** and **weakly connected** (or partially connected) have recently use to describe the degree of connectivity (i.e. channel bandwidth and availability) between system components. In general, a mobile is said to be operating in disconnected mode if no communications with its base station are possible. Weakly connected operation implies that limited communication is possible .

In a wireless environment, the assume of always online may not be correct. Due to some bandwidth limitations such as reduced throughput rate and increased data transfer cost, mobile devices might not be able to connect to the servers at all or they only have some limited connections to the servers. To solve the problem, **disconnected operations** and **weakly connected operations** are introduced, which are discussed in detail in this chapter. These two kinds of operations can help improve the availability of files with the use of some techniques like hoarding, emulating, reintegration, etc. Both **disconnected and weakly connected operations** can help improve the availability of files in a wireless and distributed environment. disconnected and weakly connected operations target more on **availability**.

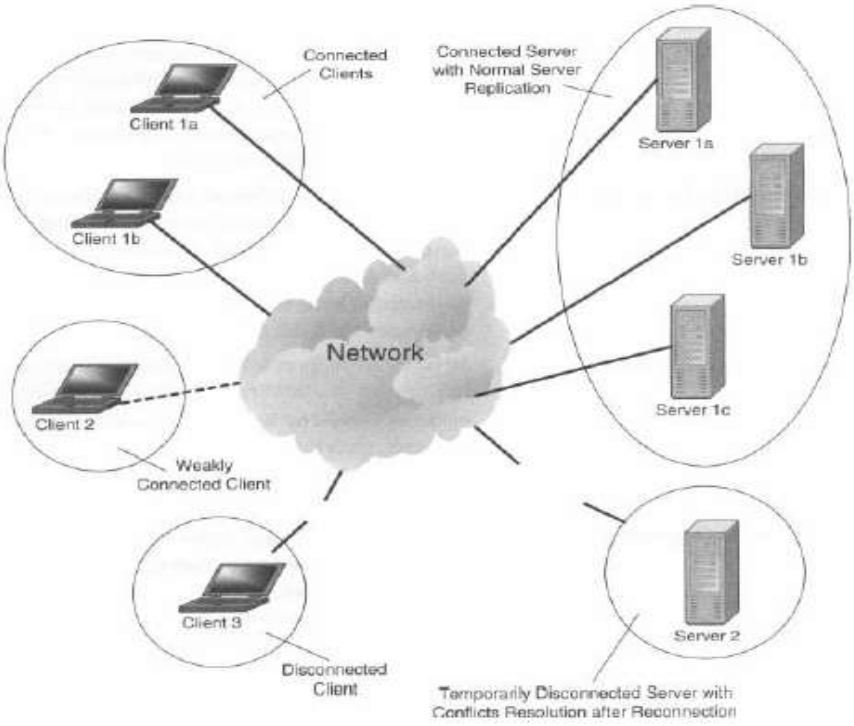


Figure (1.1) disconnection and weak connection operation

1.2 Disconnected Operation

users usually download files that can be fit into their portable devices and it is highly probable that the files will be used in the near future. Because of these reasons, in disconnected operations, the LRU (Least Recently Used) policy is utilized, which is assisted by the prediction of file access of users.

during disconnected operations, the same file system name can still be used. The files which have been downloaded can still be accessed and updated, and possible associated conflicts are automatically resolved after the client reconnects to the network.
Figure 1.2 illustrates the implementation of disconnected operations In order to support disconnected operations, three states are defined in the Venus cache manager according to different situations-hoarding, emulating and reintegrating. In normal situation, Venus is in the hoarding state which performs read/write operations normally However, when disconnection occurs, Venus then transits to the emulating state in which servers are pretended to be present. After reconnection, Venus transits to the integrating state in which all the updates and files are synchronized with the servers. After the synchronization process, Venus returns to the hoarding state.

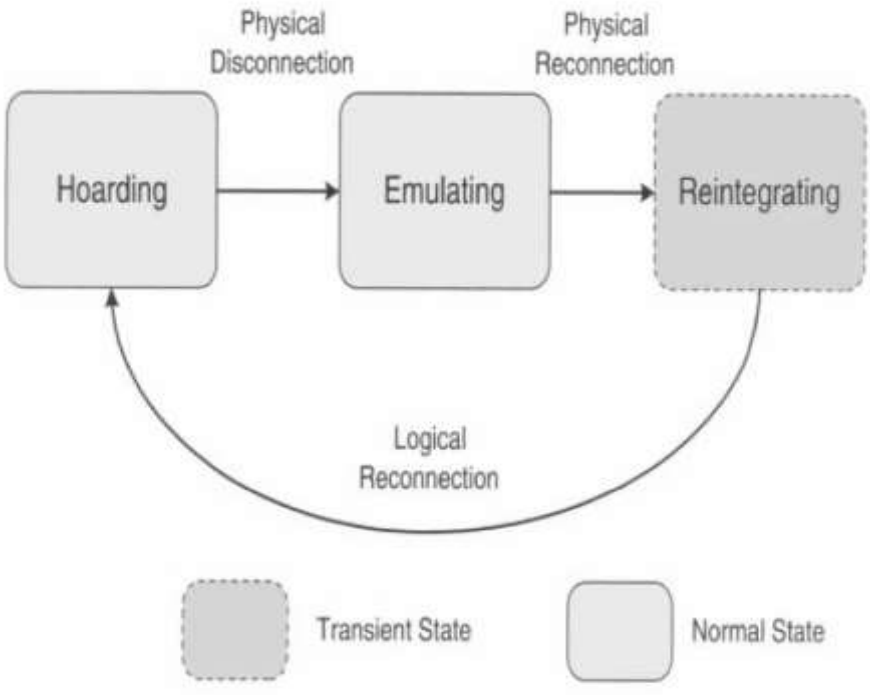


Figure (1.2) implementation of disconnected operations

1.3 Characteristics of different states in disconnected operations.

1.3.1 Hoarding : In this state, the client is always alert of disconnection by caching some files that are important. However, due to the limited storage space in the client device, only certain files can be cached. To choose the appropriate files to be cached, two kinds of information are used-implicit and explicit. Implicit information is based on algorithms like LRU as stated before. On the other hand, explicit information is based on the client hoard database (HDB), which includes the files that are explicitly required to be cached. With these two types of information, Venus periodically checks for the list of caching files to prepare for disconnection.

1.3.2 Emulating : Venus enters this state during disconnection. In this state, read/write requests on the Coda file system are redirected to the cache. However, only those requests which involve files in the cache can be satisfied. On the other hand, any changes to the files are required to logged, so that the updates performed during the period of disconnection can be delivered back to the system without loss. This is done by the client modification log (CML). In order to reduce the number of records on the log, certain optimization is performed on the log file. For example, if the user creates a file, saves it and then deletes it, there will be three actions recorded in the log. By performing log optimization, these three records are figured out and removed.

1.3.3 Reintegrating : During reintegration, with the help of the CML, Venus performs update operations as stated in the server replication part previously. The logged updates are propagated to the AVSG and consistency is checked by the servers in the AVSG (Available Volume *Server* Group).

1.4 WEAKLY CONNECTED OPERATIONS

As discussed before, disconnected operations are motivated by the popularity of portable devices. Nowadays, most of the portable devices are integrated with certain wireless technologies. For example, mobile phones are integrated with 2G/3G technologies, laptops and **PDAs** are integrated with IEEE 802.11 a/b/g, etc. On the other hand, since disconnected operations have relatively low quality of files due to the use of caching, researchers have started thinking about the possibility of using temporary wireless connections to improve the quality of the cache. This results in the emergence of weakly connected operations .

Weakly connected operations are very similar to disconnected operations. However, as shown in Figure 24.6, there are some fundamental differences. In the design of weakly connected operations, Venus can enter three states-hoarding, emulating and write disconnected. Different from that of disconnected operations, the transient state, i.e., reintegrating, is replaced by the write disconnected state. Upon recovering from disconnection, Venus first enters the write disconnected state. It transits to the hoarding state only after all the updates are performed and a strong connection is detected.

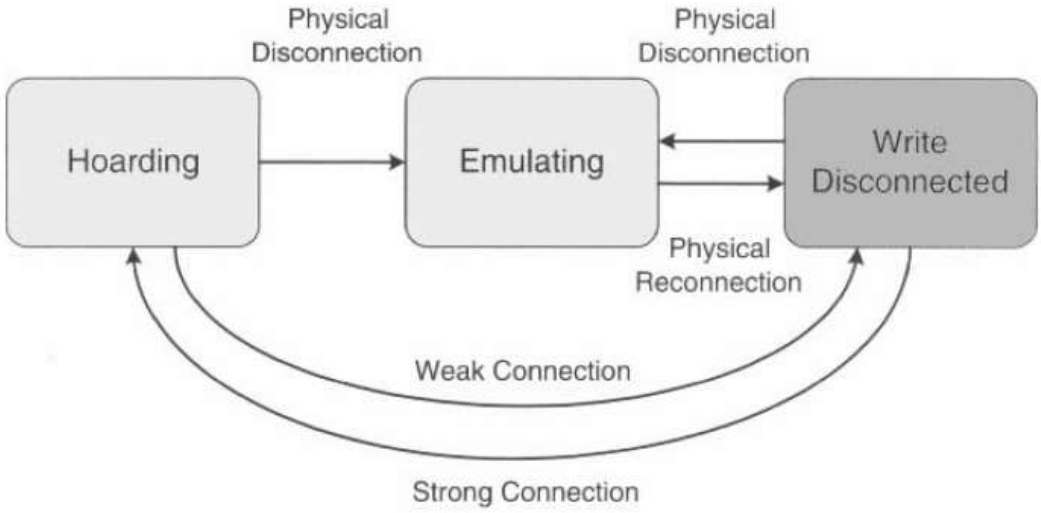


Figure (1.3) Implement of weak connection operation

The write disconnected state is the state in which weakly connected operations differ from disconnected operations. This state is a mixed state between the hoarding, emulating, and reintegrating states. In the write disconnected state, updates are logged similar to that in the emulating state. At the same time, the updates are propagated to the servers using trickle reintegration asynchronously. On the other hand, since the device has limited connection ability in this state, cache misses which cannot be served in the emulating state can now be served. However, if the size of the file requested is too large, user intervention may be required. As mentioned before, CML (client modification log) is the place for logging updates, and it is very useful for keeping it compact. Due to this reason, optimization techniques are usually applied to the CML as explained in the previous section. However, the use of trickle reintegration may result in the case that some updates are propagated to the servers before optimization can be performed on the CML. To mitigate this situation, an aging window is adopted into the log file to prevent premature updates from reducing the efficiency of log optimization. Figure 1.4 shows a typical CML log file which is under trickle reintegration. The updates can only be propagated after a certain time governed by an aging window. Specifically, in the figure, only those updates older than time A can be propagated.

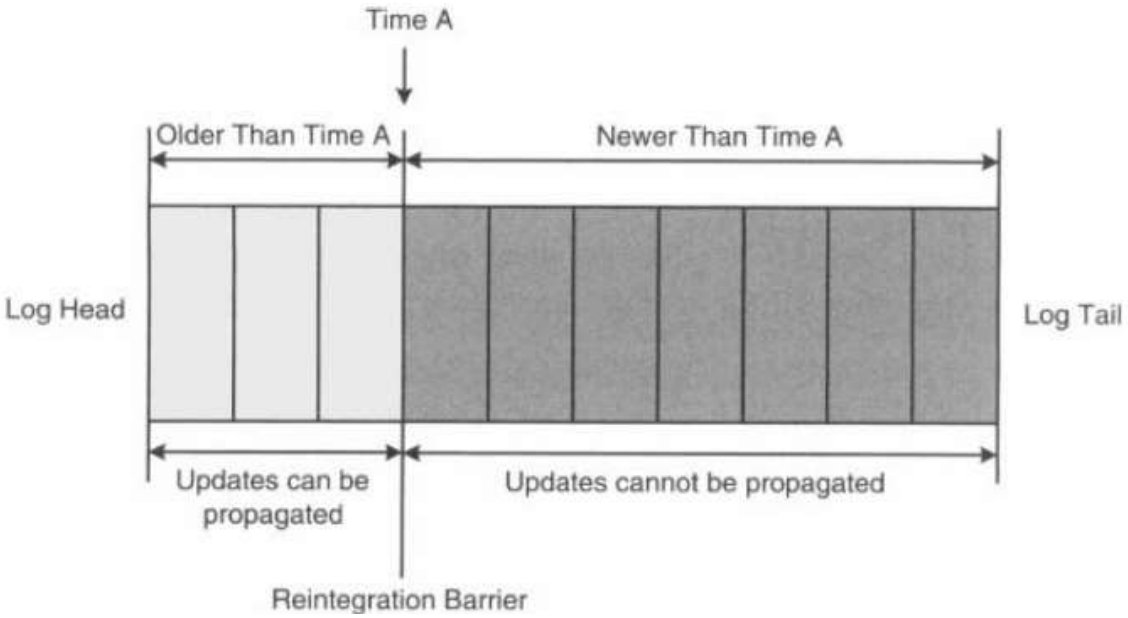


Figure (1.4) typical CML log file which is under trickle reintegration

Additionally, there is one more technique which can be used in trickle reintegration to improve the performance in the write disconnected state-the choosing of a different reintegration chunk size. This size can be chosen according to the connection link quality and bandwidth. As a result, different connection environments and situations can be covered with this technique.

1.5 Service Discovery

In network systems, a service may refer to any software or hardware facility that is available to the user, Service discovery in the computing environment can be defined as network protocols that allow automatic detection of hardware devices and services as well as those generated by activities of humans on a computer network. Once the demanded service is located, the user may invoke and access it if it is a Web service or go to the actual location of its provider to get served.

Each service discovery protocol (SDP) consists of two basic participating entities: **the service provider** and **the service user**. The service provider is the entity that offers the service, whereas the service user is the one that is interested in finding and using a service. In conventional frameworks, the role of an entity is fixed for both the service provider and the user; however, the service entity can be both the provider and the user at the same time. For service discovery, a considerable number of frameworks already exist, but the core mechanism for service discovery remains the same *advertise*, *discover*, and *access*. The advertisement or the process of service publication is usually carried out by deploying a third participating entity known as the *service directory*. This yellow page is responsible for hosting partially or entirely all the service information for service matching in the discovery phase. However, as we will see later, a service directory is not a compulsory entity in a service discovery framework and has its own advantages and disadvantages. Service discovery is the process of finding services, where the mechanism used is strongly related to the service organization scheme in the framework as well as to the underlying network types. As an example, the organization can be centralized or distributed, and the network can be wired or wireless. Finally, service access refers to the method used to invoke and utilize the service if it is a Web service.

1.6 Industry Standard Services

In this section, we would like to briefly introduce existing frameworks adopted by industry to give readers an overview of service discovery in real-world scenarios. The industry standards presented in this chapter include Universal Description, Discovery, and Integration (UDDI) , Jini , Universal Plug and Play (UPnP) , Bluetooth , Service Location Protocol (SLP) , Salutation , Bonjour , and DEAP space

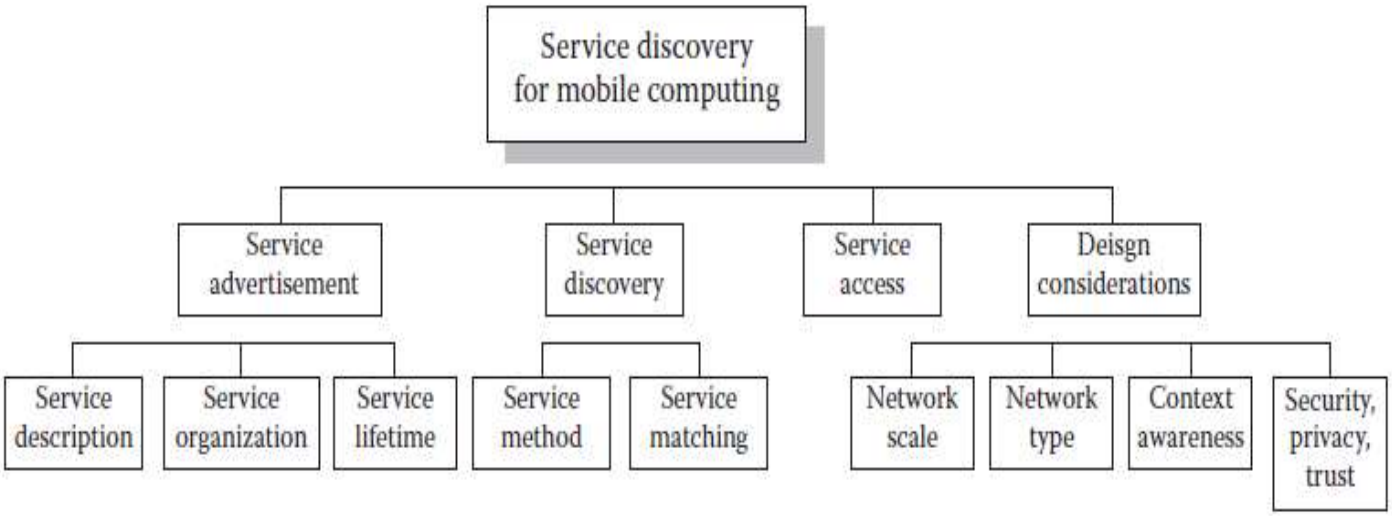


Figure (1.6) Service discovery for mobile computing

1.6.1 Universal Description Discovery and Integration

The UDDI project is an industry initiative to define publishing and discovering Web services and to advance the B2B (business 2 business) integration on the Internet . The most important component in the framework is the Universal Business Registry (UBR), which is a master directory for all publicly available Web services. Four types of information are stored in the UBR, namely, business information, service information, binding information, and information about specifications for services. Moreover, to better understand services such as the format of exchanged data and the protocol used, the technical model (t Model) is linked with each service template. For service registration and discovery, the UDDI provides a set of application programming interfaces (APIs) for service providers and consumers to interact with the registry. For example, service providers can use *save_xx()* and *delete_xx()* to update and delete their entities from the registry. Similarly, service consumers can use *find_xx()* and *get_xx()* to find and get detailed information about the entities in which they are interested.

Figure (1.7) ***Universal Description Discovery and Integration***

1.6.2 Jini

The Jini framework developed by Sun Microsystems and now transferred to the Apache River project specifies a way for clients and services to find each other on the network and to work together to accomplish a task. The whole architecture is built on top of the Java Virtual Machine (JVM), and thus is platform independent. It organizes services into service groups called djinn, and each group has a DNS-style name associated with it. Within djinn, there is a central registry called the Jini Lookup Server (JLS) that maintains a flat collection of service items. Each service item represents an instance of a service, and is described by a universally unique identifier (UUID) as well as a collection of attributes. When looking for a service, clients must specify a service Template including a UUID (if known), service type, service attributes, and so on. A match is made between the service description (i.e., service Item) and service Template by using a set of rules. Note Jini also specifies how to discover one or more JLSs by using multicast and unicast network communication protocols.

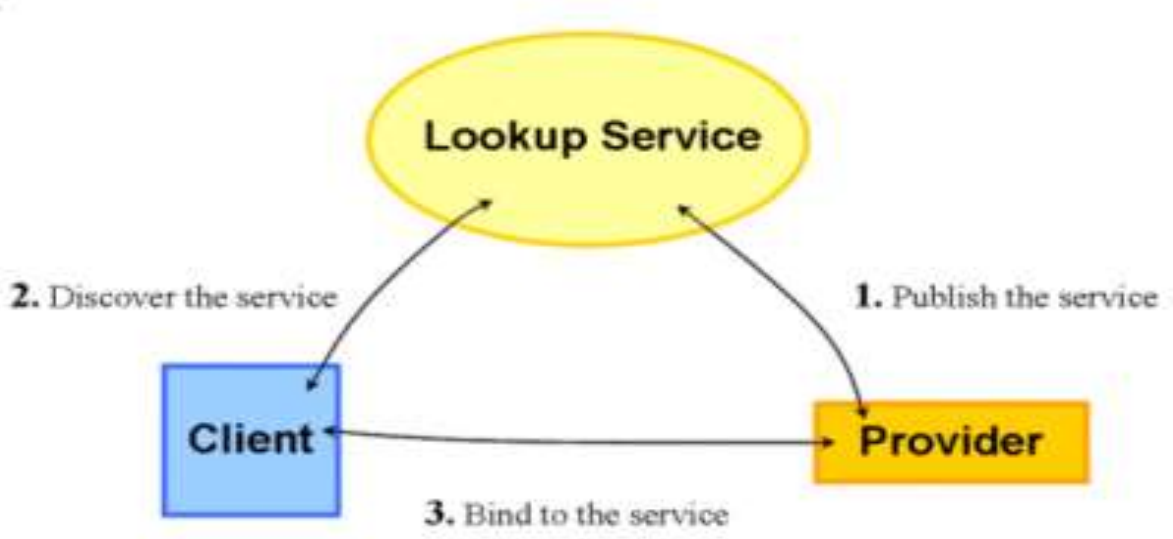


Figure (1.8) Jini Service discovery

1.6.3 Universal Plug and Play

The goals of the UPnP protocol are to allow devices to connect to networks with zero configuration and to simplify the process of using devices from a wide range of vendors. There are three basic components in the protocol: ***devices*, *services*, and *control points*.** A UPnP device is a container of services and nested devices; a service is the smallest unit of control; and a control point is a controller capable of discovering and controlling other devices. Whenever a service wants to join the network, it needs to send out a message to notify others about its presence. The advertisement is done using a multicast protocol. However, before the device advertises its services, it needs to obtain an Internet protocol (IP) address for device identification. UPnP supports the Auto IP protocol; that is, the device can get its IP address even without a dynamic host configuration protocol server. Once the device advertises its service information including the service URL, other devices in the network can observe the advertisement and control points can record it. When a client wants to discover a service, the simple SDP over IP networks is adopted. The client can do a multicast of its query to other devices in the network. The query can be answered either by the control point, which has the requested service information, or by the service directly.

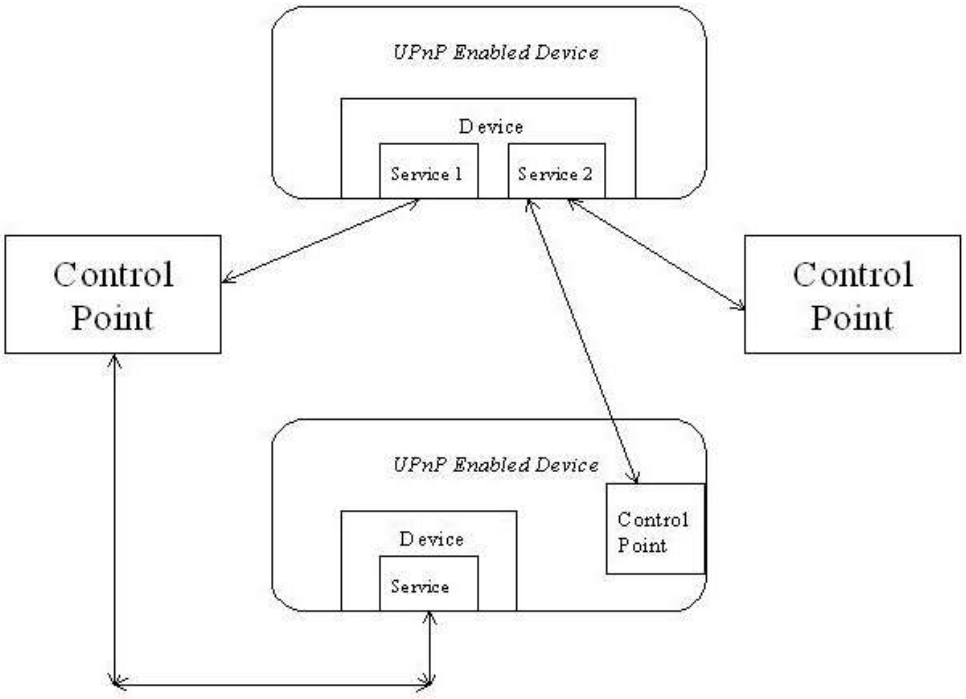


Figure (1.9) Universal Plug and Play Service discovery

1.6.4 Bluetooth

Bluetooth , as a recent industrial standard, is a cable-replacement technology designed to wirelessly connect peripherals, which is unlike UPnP's wired constraint. The technology allows a Bluetooth-enabled device to communicate via short-ranged radio links with low power and low cost. For service discovery, the SDP is defined at the application layer. Services in the SDP are represented by a *service record* form in the same way as Jini, and each service is an instance of a Service class that provides an abstraction for attributes associated with each service record; however, users may create their own attributes for the service provided. Similar to UPnP, the Bluetooth protocol is designed for personal area network and with low energy cost in mind; therefore, it only supports short-range communication, for example, 1–100 m.

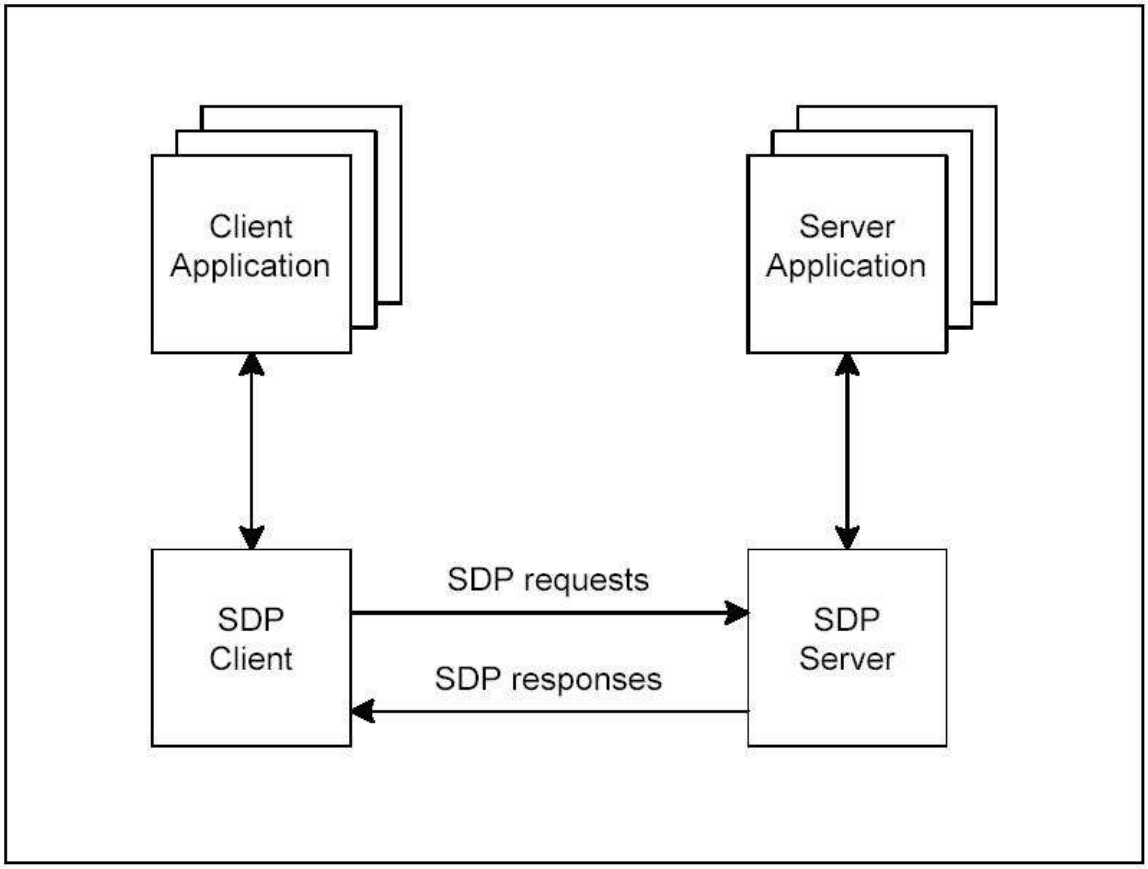


Figure (1.10) Bluetooth Service discovery

1.6.5 Service Location Protocol

The SLP is an Internet Engineering Task Force (IETF) standard that supports automatic resource discovery. Similar to UPnP, this protocol is only for IP-based networks, though it scales better than UPnP. The SLP defines three types of agents in the service discovery framework: *user agent* (UA), *service agent* (SA), and *directory agent* (DA). The UAs acquire service handles for end-user applications that request the services. The SAs are responsible for advertising service handles to the DAs. The DA maintains a list of the advertised services in the network. Services are described using unique URLs and a set of attribute–value pairs. Like the Jini framework, the SLP uses a multicast protocol for service registration. Once SAs join the network they multicast a registration message. Also, if there is a new DA available, it will send an unsolicited advertisement infrequently to inform the UAs and SAs of its presence. Service discovery can be done either by multicasting a service request message so as to be directly answered by existing SAs or by unicasting the request to a known DA to get relevant information.

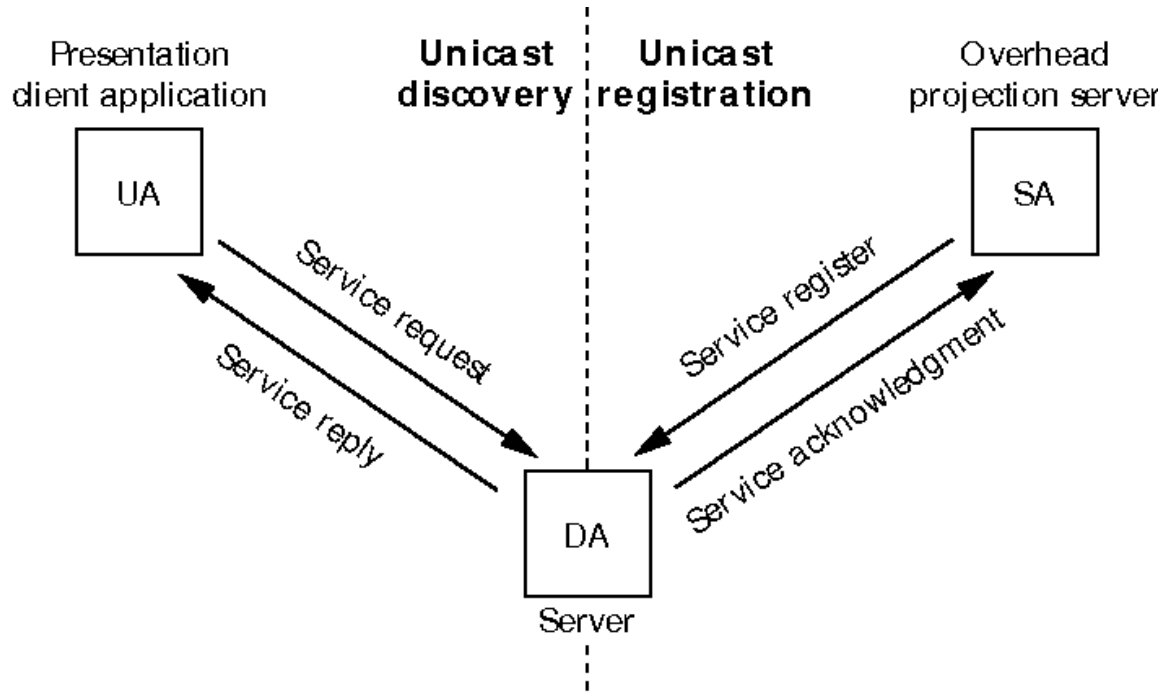


Figure (1.11) Service Location Protocol

1.6.6 Salutation

The Salutation protocol is an open standard service discovery and session Management protocol, although currently dissolved. The architecture defines an entity called the Salutation Lookup Manager (SLM), which, similar to JLS in Jini, acts as a service broker for services in the network. Services are described by using their different functions called Function Units that represent essential features of services and are kept by the local SLM. Service discovery can be performed across multiple SLMs either by unicasting to a central directory server (e.g., DA in SLP) or by broadcasting. The matching is done by sending the required service type from the local SLM to the remote SLM through the Remote Procedure Call (RPC) for comparisons. The Salutation architecture defines a Transport Manager (TM) component that implements the detailed transport functionality to operate on different network technologies such as Infrared and Bluetooth. An SLM may have multiple TMs for the underlying communication methods. The Salutation protocol also has a light version, called Salutation-Lite, which is a smaller footprint version and can be run on the Windows CE operating system for the mobile communications market. Therefore, Salutation makes the least assumption of the underlying protocol stack and computing resources, and it can be easily ported to low-power handheld devices.

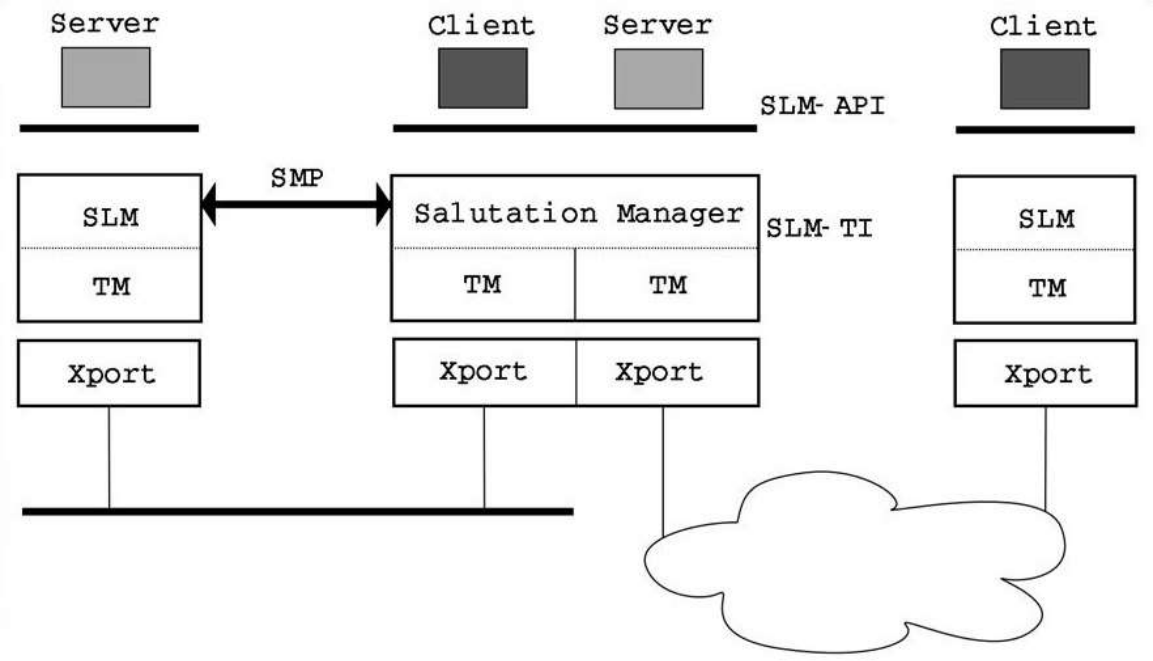


Figure (1.12) Salutation Service discovery

1.6.7 Bonjour

Bonjour is an SDP developed by Apple for its implementation of zero-configuration networking. It locates devices such as printers, as well as the services provided by them in a local network (e.g., iTunes music), on the basis of standard IPs. The core component of service discovery in Bonjour is the Multicast Domain Name System (MDNS) that is used for service name-to-address translation. More specifically, service discovery in Bonjour is accomplished by *browsing*. The DNS-format queries are sent over the local network using IP multicast for a given service type and domain, and any matching services reply with their names. The result is a list of available services to choose from. Because these DNS queries are sent to a multicast address, no single DNS server with global knowledge is required to answer the queries.

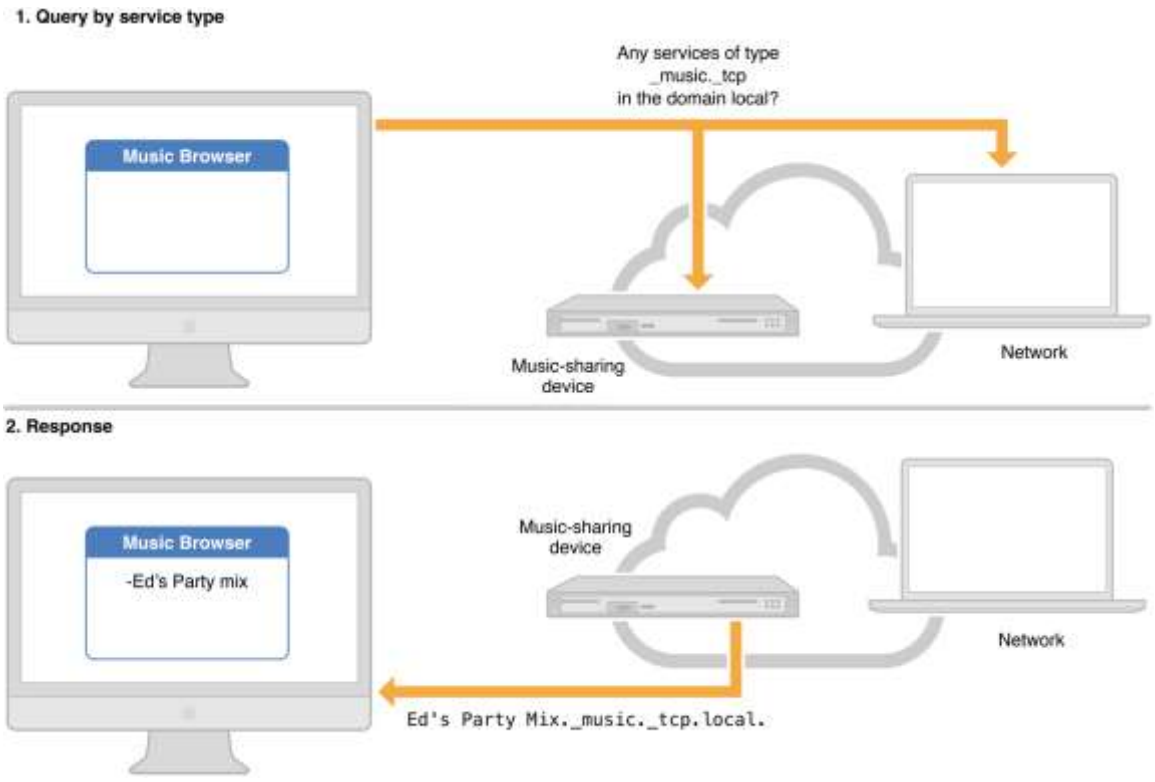


Figure (1.13) Bonjour Service discovery

1.6.8 DEAPspace

The DEAPspace project started by IBM Research Zurich Laboratory, tries to address P2P networking of pervasive devices instead of client–server networking. All service information is stored on service providers. The DEAPspace algorithm uses a push-based approach, in which all devices hold a list of all known services called the *world view*. The world view of each device is further broadcast to its neighbors in short range to make all devices in the network know services on other devices. To indicate service expiry time, the time-to-live is broadcast along with the service information. DEAPspace can be viewed as a representative for P2P-based service discovery in a wireless medium. Each device can be either the service provider or the client or both at the same time.

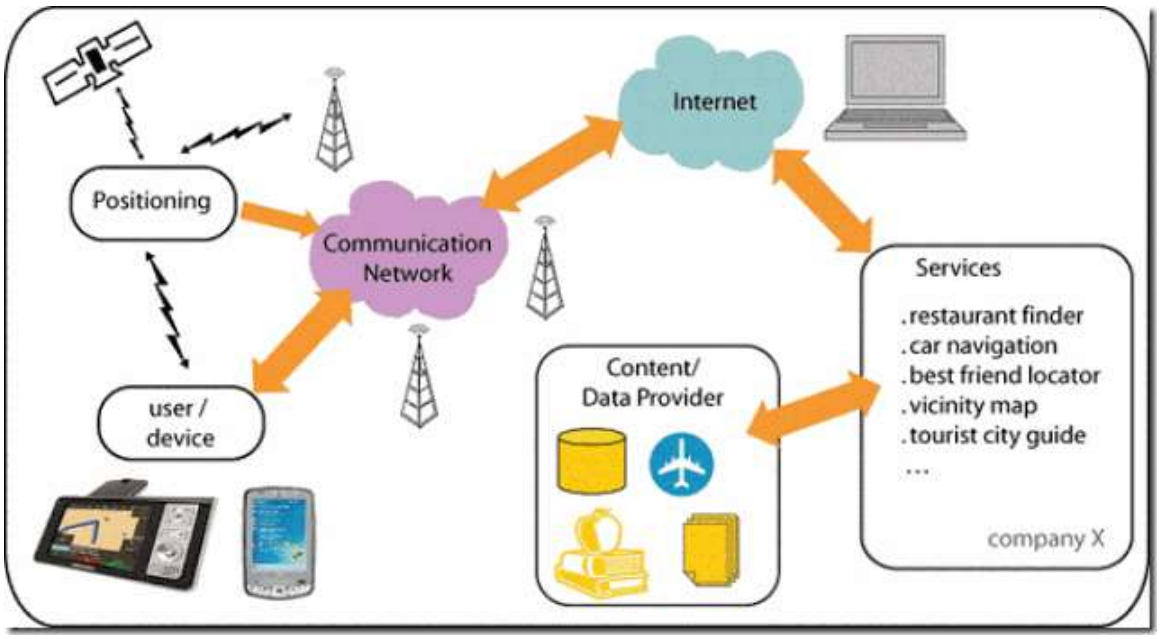


Figure (1.14) DEAPspace Service discovery

1.7 Classifications of Service Discovery Frameworks

1.7.1 Service Advertisement

Components for service advertisement include the service description mechanism, the service organization scheme, and the service lifetime maintenance method.

1.7.1.1 Service Description

Services must be described before they can be discovered the quality of service (QoS) matching is strongly correlated to the richness of service description. The simplest description format is the attribute– value pair structure that is deployed in Jini/Bluetooth; they describe all the service information using a list of attributes in the service template/record. To further reflect the hierarchical relationships among attributes and to allow flexible design,. Note that service attributes can be represented in a richer format rather than in text format only. As a case, in Jini, Java objects are allowed to be used as attributes that associate extra descriptive information with a service. Besides, for Web services, a specific markup language is used to describe their operations and messages involved, and we have standard for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information in XML format.

1.7.1.2 Service Organization

The services are organized according to some specific context elements. The most commonly used element is the service location, and the discovery framework is location based. Most such frameworks consider the physical location of services and cluster them on the basis of physical proximity. This approach is effective when the user is searching services nearby, and the response time can be reduced owing to the fact that routing of queries is restricted to a certain area/network.

1.7.1.3 Service Lifetime

Services can be added or removed from the framework at any time. Thus, service availability or service lifetime maintenance is necessary. The maintenance is especially important for mobile computing, as the mobility of service providers may greatly affect service availability. Generally, there are two main strategies to maintain service lifetime. One is a simple approach that requires service providers to explicitly register or deregister their services to the framework. Service information in the framework (i.e., cache) is assumed to be persistent until the respective provider explicitly requests removal of them. This approach is implemented in the UDDI; however, this approach does not guarantee the quality of registered services. Some optimization techniques may be applied, such as periodically sending ping messages to check the service availability. The other maintenance approach is based on leasing; that is, rather than granting services or resources until that grant has been explicitly canceled by the service provider, a leased resource or service grant is time based. The provider may renew or cancel the lease before the lease time expires, but in the case of no action, the lease simply expires and the corresponding service (i.e., cache information) is removed from the framework. Typical examples are Jini and SLP frameworks.

1.7.2 Service Discovery

Service discovery is the most important element in the whole framework. It consists of two main components , **discovery method** and **matching strategy**. The discovery process deals with the routing of service queries to the target service provider or the node where the requested service information is stored. The search method is strongly related to the organization scheme used for service management as well as the network type.

1.7.2.1 Discovery Method

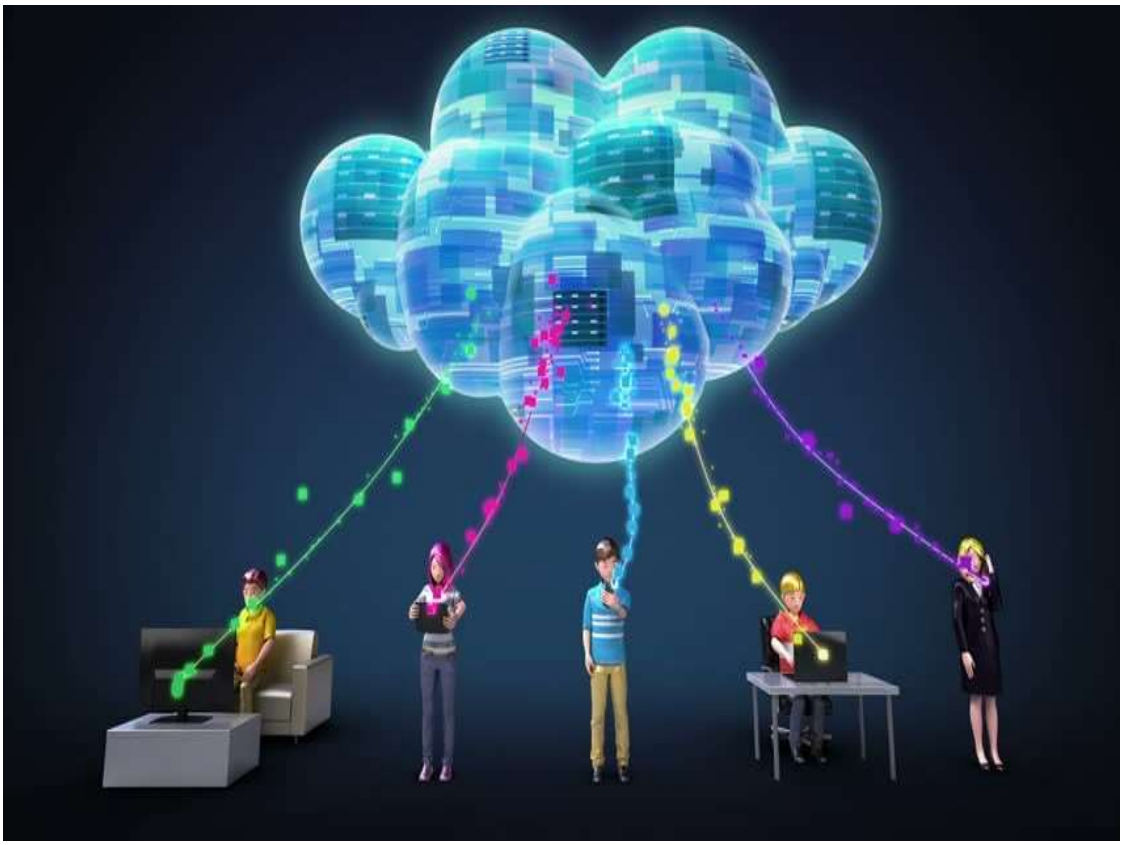
There are two fundamental models for service discovery: (i) *Push model*—service providers or directory servers actively push their service information into the network so that the requestors can be aware of their existence; however, periodical announcements would waste the bandwidth and cause large network overheads. As a result, most approaches only push service information when the service is newly registered or updated and rely on the pull model for service discovery. (ii) *Pull model*—query the network for the desired service when needed. This model is client initiated, and thus is based on demand in order to reduce unnecessary announcements as in the push model.

1.7.2.2 Matching Strategy

Service matching is also an important component in service discovery, as it determines the relevance of the service on the user's request and then concludes whether it should be returned as a result. As mentioned in service description, service matching is strongly related to service description. The richer the information provided by service description, the more complex the matching strategy tends to be. On the whole, there are two main factors to consider when developing such a strategy: *matching content* and *matching technique*. Matching content refers to the information considered for service matching whereas the matching technique refers to the detailed method used for matching on the basis of content.

1.7.3 Service Access

After retrieving the desired service information, for example, either the location (can be URL) of the service or WSDL in the case of Web services, the next step is to access them. For conventional business services, such as shop or restaurant services, the user may approach the location of the service to get served, and this explains the importance of context awareness in service discovery. For Web services, communications can be done through the HTTP/SOAP protocols, and messages are usually in XML format. Note that Jini provides different mechanisms for service invocation, which utilizes Java RMI protocol. A user may also subscribe to a service he is interested in to receive information concerning changes. The subscription can be either between the user and the directory server or between the user and the service directly.



1.8 Considerations of Service Discovery for Mobile Computing

many solutions are available when designing components of a service discovery framework. However, there is no standard benchmark to show which one is optimal, as the considerations introduced in this section should be taken into account when designing a service discovery framework. Note that although the considerations presented in the following are for service discovery in mobile computing, they can be applied to any discovery framework.

1.8.1 Network Scale

The first consideration is the network size or discovery scale, which may vary from a couple of devices to the Internet scale. We classify them into two major categories: *local scale* and *global scale*. Where local scale refers to those personal, home, office or enterprise networks, global scale refers to wide area or the Internet scale networks. The design and performance of service discovery frameworks are largely determined by the characteristics of the underlying network. For instance, in a local-scale network, like the LAN in a home, office or even a company, while the routing efficiency is not the primary concern, the searching effectiveness is rather highly demanded. Designers may choose to apply complex service matching strategy, such as semantic matching over Web service processes, to achieve high accuracy. On the other hand, for global-scale service discovery which may involve world-wide services, the scalability of the framework is always the major concern. Managing services or their information in an effective way to support efficient service discovery is challenging. Most existing frameworks deploy either a DNS-style distributed centralized approach or P2P-style organization scheme. Global-scale service discovery also impose high demands on the security, privacy, and trust of service information.

1.8.2 Network Type

Most industry standards do not address the underlying network type, as they focus on application-level service organization and discovery. Examples are UDDI, Jini, and SLP. However, these examples were developed in the early days when wireless technology was not matured. Although the concepts in wired infrastructure can still be applied in wireless topology, some topology-related issues need to be addressed separately.

For MANETs, two practical issues are limited topology knowledge due to wireless communication range and changing topology due to node mobility. The distributed centralized approach and the distributed approach are deployed for service discovery in MANETs. However, in fully distributed approaches, such as DEAPSpace, broadcasting and multicasting techniques are applied for service discovery, imposing large system overheads. To further reduce the routing overheads in MANETs and improve efficiency of energy consumption, some approaches integrate the service discovery process with the routing process. This type of cross-layer design piggybacks service information onto routing messages, and service discovery is done on the network layer so as to reduce unnecessary message exchange at the application level. To summarize, all these cross layer approaches have shown their efficiency in terms of network throughput, service acquisition time, and energy , but this kind of approach destroys the protocol stack in the Internet OSI model and tends to be protocol specific, which may limit the interoperability of the discovery framework, especially when deployed for large-scale and heterogeneous networks.

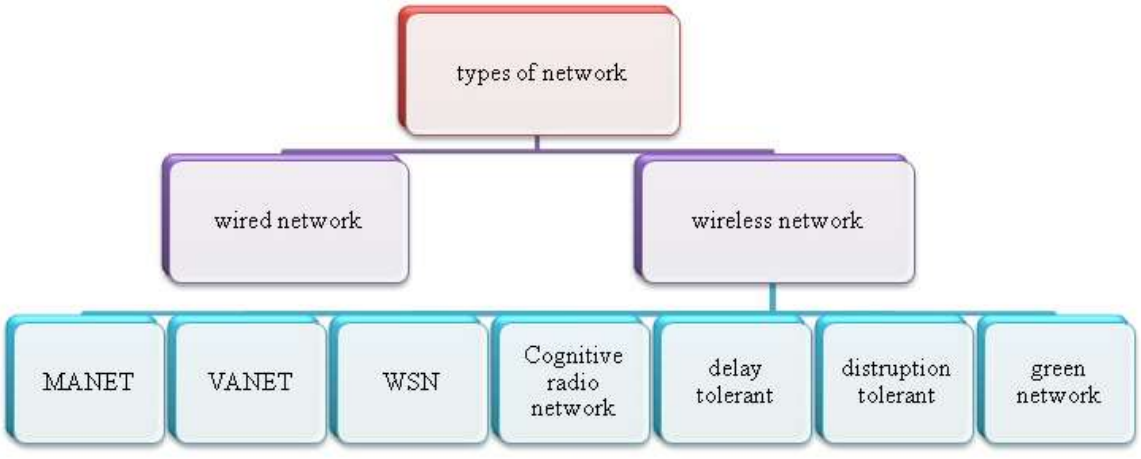


Figure (1.15) Types of networks

1.8.3 Context Awareness

Besides the network scale and type, another important consideration for service discovery in mobile computing is context awareness. The implicit context information may not be that crucial in the conventional P2P search, as the discovery is more based on stable resources, such as files or images. In fact, a service in mobile computing is a dynamic entity: its provider may be on the move, like for a taxi service. The current context of the user and that of the service may be quite different, which results in unavailability or irrelevance of the service, and thus influences the satisfaction of the user. We focus on existing frameworks that use contexts of users and services to better fulfill the discovery task

Context can come from various sources, from lower to higher levels. Low-level contextual information includes raw data, such as location information from a GPS system, time information and those from sensors, for example, temperature and light level. High-level contextual information refers to contexts that cannot be directly retrieved and may require a reasoning process. A typical example for high level contextual information can be a user's activity, which requires derivation on the basis of low-level contextual information, such as a user's location and objects touched. Context can also be classified on the basis of its source type, for example, environmental versus individual. A user's preferences can be categorized as an individual context. To define and store context data in a machine-readable format, a well-designed context model is needed in any context-aware system.

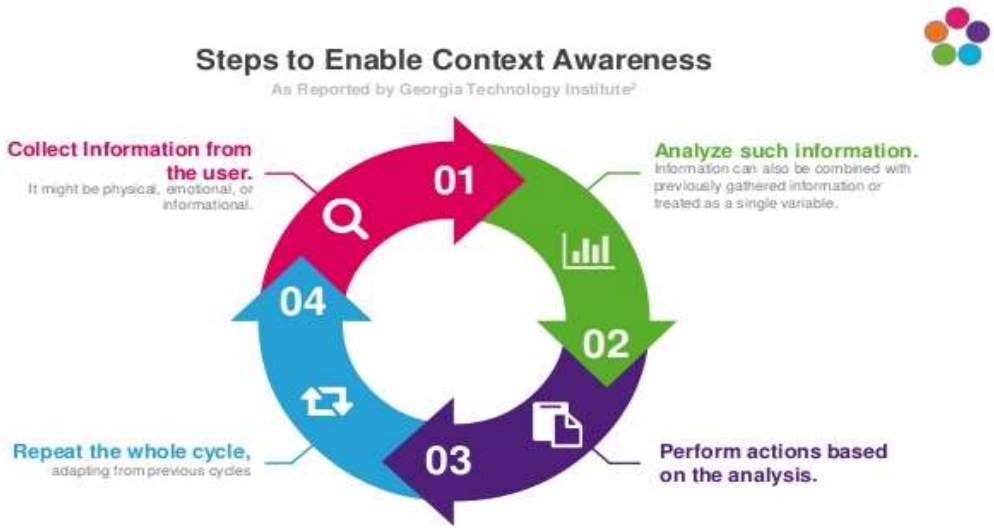


Figure (1.16) ***Context Awareness***

1.9 Security and Privacy

1.9.1 security

Security is extremely important for mobile computing, as the environment is changing and there are large numbers of uncontrolled data exchanges. From a service provider's perspective, the authentication and authorization of the user's identity need to be checked and service information should be stored at a trusted directory server if there is one; from a user's perspective, services retrieved must be reliable and personal information such as location and personal preferences should be kept confidential.

- The security model for a UDDI registry can be characterized by the collection of registry and node policies and the implementation of these policies by a UDDI node.

- Jini security framework provides mechanisms and abstractions for managing security, especially in the presence of dynamically downloaded code. It deploys three major mechanisms to provide authentication, integrity, and confidentiality.

- The UPnP protocol, as default, does not implement any authentication, as it runs on local systems and their users are assumed to be completely trustworthy.

- The Bluetooth Generic Access Profile specifies three security modes to achieve authentication, integration, and confidentiality: Security mode 1 is for applications without security concerns; in security mode 2, a Bluetooth device shall not initiate any security procedure before a channel establishment request has been received or a channel establishment procedure has been initiated by itself; in security mode 3, the Bluetooth device must initiate security procedures before the channel link is set up. In reality, the user must use an eight digit personal identification number (PIN) to secure devices.

- Other industry standards, such as Salutaion and Bonjour, apply simple password- based authentication and access mechanisms for security.

1.9.2 Privacy

Privacy is defined as the right of an individual to control information disclosure. As such, an invasion of privacy is considered to occur when information regarding an entity is disclosed without the entity's explicit consent . It depends on the willingness of the entity to share its sensitive information, such as service content and location information. The service provider may not want its service information to be stored outside of its current domain, and service users may not want to reveal their current positions or leave personal data to directory servers or services being used. Hashing and encryption techniques can be applied, like in SDS (Safety Data Sheets) . In SDS, bloom filters are used to protect the service and personal information of the user, such as identity, certificates, and attributes. Basically, the bloom filter allows an easy test of domain membership. Once the user authenticates with the directory that represents a domain, he or she can send a service query.

References

1. Wikipedia. Service definition. http://en.wikipedia.org/wiki/Servic . Last accessed on June 14, 2009.

2. Marin-Perianu, R., Hartel, P., and Scholten, H. A classification of service discovery protocols. Internal Report, Department of Electrical Engineering, Mathematics, and Computer Science, University of Twenty, Netherlands, 2005.

3. Zhu, F., Mutka, M.W., and Ni, L.M. Service discovery in pervasive computing environments. IEEE Pervasive Computing, 4(4), 2005, 81–90.

4. Hosseini-Seno, S.A., Budiarto, R., and Wan, T.C. Survey and new approach in service discovery and advertisement for mobile ad hoc networks. International Journal of Computer Science and Network Security (IJCSNS), 7(2), 2007, 275–284.

5. Ververidis, C.N., and Polyzos, G.C. Service discovery for mobile ad hoc networks: A survey of issues and techniques. IEEE Communications Surveys and Tutorials, 10(3), 2008, 30–45.

6. Meshkova, E., Riihijärvi, J., Petrova, M., and Mähönen, P. A survey on resource discovery mechanisms, peer-to-peer and service discovery frameworks. Computer Networks, 52(11), 2008, 2097–2128.

7. OASIS. UDDI version 3.0.2. 2005. http://www.uddi.org/pubs/uddi_v3.htm . Last accessed on June 14, 2009.