E-ISSN: 2321–9637
Volume 2, Issue 1, January 2014
International Journal of Research in Advent Technology
Available Online at: http://www.ijrat.org

Zero Configuration Networking: Automatic Service Discovery and Service Discovery Protocols: sdAvahi

Shirin Hijaz Matwankar , Dr. Subhash K. Shinde
Computer Engineering
Lokamanya Tilak College Of Engineering, Navi Mumbai
Mumbai University
shirinmatwankar@gmail.com
8879788737

ABSTARCT:
Smart network is the network where devices want to access services immediately when they will enter into the network without user intervention. There is no structure to the network, and each member is typically able to communicate with every other member.
Zero configuration networking is a best example of smart network. Zero configuration networking is a method of network devices without requiring configuration and administration. Zero configuration is able to allocate addresses without a DHCP server, translate between domain names and IP addresses without a DNS server, and find services, such as a printer, scanners, backups, mail servers, web services, file servers and instant messaging ; without a director service.
The main task of a service discovery protocol is to locate services for users and to advertise services for service providers. To relieve the end user of the configuration burden that arises when new devices are introduced in the network. These enable automatic discovery of the services these devices offer. Five protocols of service discovery are studied which are Bonjour, Avahi, Service location protocol, Jini, UPnP (universal plug and play).An algorithm is introduced sdAvahi which is used to filter and categorize services.

Keywords: zero configuration, service discovery protocols, Bonjour, UPnP, SLP, Jini, Avahi

1. INTRODUCTION TO ZERO CONFIGURATION

For a consistent use of the term “Zero configuration”, we handle the following definition:
“Zero configuration IP networking, a method of network devices via an Ethernet cable without requiring configuration and administration. Zero configuration is able to allocate addresses without a DHCP server, translate between domain names and IP addresses without a DNS server, and find services, such as a printer, without a director service.”
The Internet Engineering Task Force (IETF) developed Zero configuration around 1999 when it became obvious that basic network configuration can be done without manual configuring at all. The Zero configuration Working Group of the IETF is chaired by Erik Guttman of Sun Microsystems and Stuart Cheshire from Apple Computer, with Thomas Narten (IBM) and Erik Nordmark (Sun) serving as area directors. It was chartered in September 1999.
Zero Configuration Networking is a set of technologies used together to allow for automated network configuration of devices and services, without the use of central services such as DNS or DHCP. These technologies cover Addressing, Name Resolution and Service Discovery.
Zero configuration deployments are typically small in size, to support home networking or ad-hoc networks where simplicity, ease of use and customer experience are essential. Tablets, printers, and scanners that use Zero configuration protocols have seen widespread adoption in the consumer electronics market and home networking. However, with the large increase of “Bring Your Own Device” (BYOD) occurring in the

E-ISSN: 2321–9637
Volume 2, Issue 1, January 2014
International Journal of Research in Advent Technology
Available Online at: http://www.ijrat.org

Enterprise, users want the same experience they have at home within the Enterprise Network, creating the need to support Zero configuration across the Enterprise or university campus.
Zero configuration, initially proposed by Apple, but now it is not limited to only Apple products. In fact, almost every printer with network connectivity supports Zero configuration by default. Cameras, network speakers and network appliances (such as storage appliances, gateways, wireless routers, etc.) also use Zero configuration to facilitate initial service discovery, configuration and ease of use in daily operation.
The goal of the Zero Configuration Networking is to enable networking in the absence of configuration and administration. Zero configuration networking is required for environments where administration is impractical or impossible, such as in the home or small office, embedded systems 'plugged together' as in an automobile, or to allow impromptu networks as between the devices of strangers on a train.
Correct and appropriate use would include Home and small office networks, Ad hoc networks at meetings and conferences (especially wireless networks), two devices needing to spontaneously share or exchange information
Figure 1 illustrates a simplified functional diagram explain logical interface provided by the zero configuration system and method of the invention between various applications and the multiple wireless networks that are available.

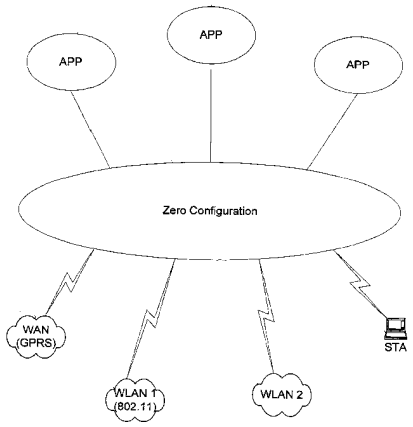

Figure1. Functional diagram of zero configuration

The zero configuration system operates as an interface between the various applications and various wireless networks such as WAN, WAN1, WAN2, STA) to which they may be connected. The main functions of any networking are addressing, naming, and service discovery.

2. SERVICE DISCOVERY PROTOCOLS

When a service is introduced into a network of computers, every computer that wants to use the service needs to know the location of the service and some kind of configuration to use the service. This requires a lot of work for system administrators in large computer networks and can be very expensive.
The main task of a service discovery protocol is to locate services for users and to advertise services for service providers. To relieve the end user of the configuration burden that arises when new devices are introduced in the network, service discovery protocols have been devised. These enable automatic discovery of the services these devices offer. Currently several protocols could be used for Zero configuration. We are going to describe five amongst them which are Bonjour, Service location protocol, Avahi, UPnP, Jini.
In figure 2, there are six operations of zero configuration. Those are address assignment, name resolution, service discovery, service description, service invocation, service representation. On the basis of these, we have differentiated the protocols.

E-ISSN: 2321–9637
Volume 2, Issue 1, January 2014
International Journal of Research in Advent Technology
Available Online at: http://www.ijrat.org

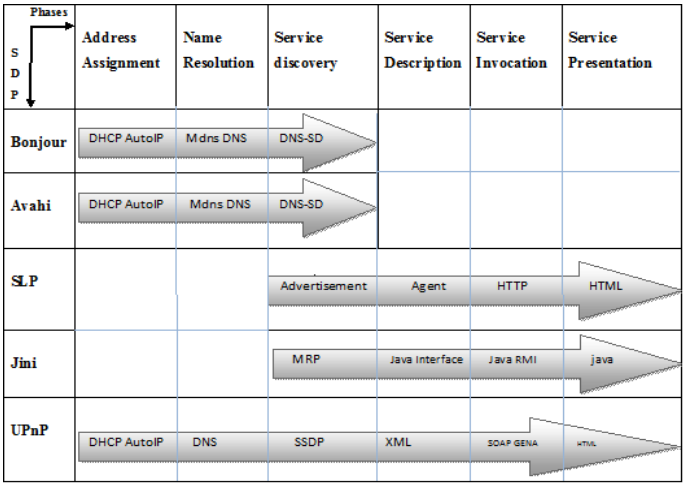

Figure2.Service discovery protocols and their operations

2.1Bonjour

The Bonjour zero-configuration networking architecture provides support for publishing and discovering TCP/IP-based services on a local area or wide area network. Bonjour is Apple's implementation of a suite of zero-configuration networking protocols. Bonjour is designed to make network configuration easier for users.

It mainly covers three areas addressing (allocating IP addresses to hosts), naming (using names to refer to hosts instead of IP addresses), service discovery (finding services on the network automatically).

The addressing problem is solved by self-assigned link-local addressing. Link-local addressing uses a range of addresses reserved for the local network, typically a small LAN or a single LAN segment. The IPv6 specification includes self-assigned link-local addressing as part of the protocol. IPv6 link-local addressing is simpler than IPv4 link-local addressing, and thus is more reliable. Because of this, it is important that your app support IPv6.When your host computer encounters a local network, it finds an unused local address and adopts it. No action on your part is required.

The solution for name-to-address translation on a local network uses Multicast DNS (mDNS), in which DNS-format queries are sent over the local network using IP multicast. Because these DNS queries are sent to a multicast address, no single DNS server with global knowledge is required to answer the queries. Each service or device can provide its own DNS capability—when it sees a query for its own name, it provides a DNS response with its own address. Additionally Bonjour provides responder that handles mDNS queries for any network service on the host computer

The final element of Bonjour is service discovery. Service discovery allows applications to find all available instances of a particular type of service and to maintain a list of named services and port numbers.

The network services architecture in Bonjour includes an easy-to-use mechanism for publishing, discovering, and using IP-based services. Bonjour supports three fundamental operations, each of which is a necessary part of zero-configuration network services publication (advertising a service), Discovery (browsing for available services), Resolution (translating service instance names to addresses and port numbers for use).

To publish a service, an application or device must register the service with a Multicast DNS responder, either through a high-level API or by communicating directly with the responder (mDNSResponder).

When a service is registered, three related DNS records are created: a service (SRV) record maps the name of the service instance to the information needed by a client to actually use the service, a pointer (PTR) record enable service discovery by mapping the type of the service to a list of names of specific instances of that type of service, and a text (TXT) record. The TXT record contains additional data needed to resolve or use the service, although it is also often empty.

2.2Universal plug and play (UPnP)

The UPnP Device Architecture (UDA) is more than just a simple extension of the plug and play peripheral model. It is designed to support zero-configuration, "invisible" networking, and automatic discovery for a

E-ISSN: 2321–9637
Volume 2, Issue 1, January 2014
International Journal of Research in Advent Technology
Available Online at: http://www.ijrat.org

breadth of device categories from a wide range of vendors. This means a device can dynamically join a network, obtain an IP address, convey its capabilities, and learn about the presence and capabilities of other devices.
The UPnP (Universal Plug and Play) implementations come from different manufacturers of electronic devices such as Intel Corporation, Microsoft Corporation, Motorola, Nokia Corporation, Philips electronics, Pioneer, Sony Electronics, Conexant System, Allegro Software Development Corporation. These companies are joined to DLNA (Digital Living Network Alliance) alliance.

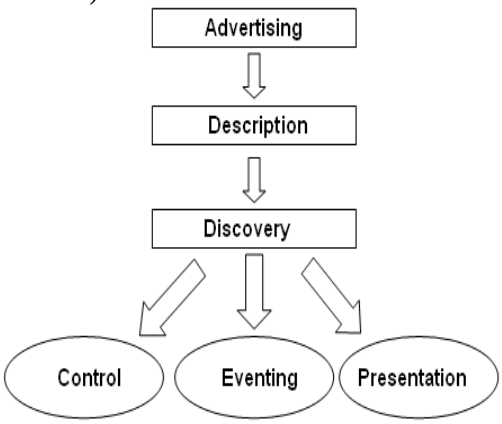

Figure3. Phases of UPnP

The UPnP consists of six layers, which are addressing, discovering, description, controlling, eventing and presentation. For addressing it uses dynamic configuration of link-local addresses, which is called auto-IP.
The UPnP is using dynamic allocation because it does not need any administrative tasks. It gives an address for a limited period of time. Client has to lease IP address for limited time and if needed the client can renew the lease. If the client does not need IP address any more then it is possible to release the lease. If during the DHCP transaction, the device obtains a domain name, e.g., through a DNS server or via DNS forwarding, the device should use that name in subsequent network operations; otherwise, the device should use its IP address.
In discovery state the UPnP device is able to advertise its own services and discover available services from other devices. Through discovery, control points find devices. After a control point has discovered a device, the control point still knows very little about the device. For the control point to learn more about the device and its capabilities, or to interact with the device, the control point must retrieve the device's description from the URL provided by the device in the discovery message. Devices may contain other, logical devices, as well as functional units, or services. The UPnP description for a device is expressed in XML and includes vendor-specific, manufacturer information like the model name and number, serial number, manufacturer name, URLs to vendor-specific Web sites, list of any embedded devices or services, as well as URLs for control, eventing, and presentation. After a control point has retrieved a description of the device, the control point can send actions to a device's service. To do this, a control point sends a suitable control message to the control URL for the service. Control messages are also expressed in XML using the Simple Object Access Protocol (SOAP).
UPnP description for a service includes a list of actions the service responds to and a list of variables that model the state of the service at run time. The service publishes updates when these variables change, and a control point may subscribe to receive this information. The service publishes updates by sending event messages. Event messages contain the names of one of more state variables and the current value of those variables.
If a device has a URL for presentation, then the control point can retrieve a page from this URL, load the page into a browser and depending on the capabilities of the page, and allow a user to control the device and/or view device status.

2.3Service location protocol

The Service Location Protocol (SVRLOC) working group has been active in the IETF for several years. The Service Location Protocol is an Internet Engineering Task Force standard for enabling network based applications to automatically discover the location including address or domain name and other configuration information of a required service. Clients can connect to and make use of services using SLP.
SLP establishes a framework for resource discovery that includes three “agents” that operate on behalf of the network-based software:

- User Agents (UA) perform service discovery on behalf of client software.

E-ISSN: 2321–9637
Volume 2, Issue 1, January 2014
International Journal of Research in Advent Technology
Available Online at: http://www.ijrat.org

- Service Agents (SA) advertise the location and attributes on behalf of services
- Directory Agents (DA) aggregate service information into what is initially a stateless repository. When there are many different clients and/or services available, SLP is adapted to make use of Directory Agents that act as a centralised repository for advertised services.

Services are advertised using a Service URL, which contains the service's location: the IP address, port number and, depending on the service type, path. Client applications that obtain this URL have all the information they need to connect to the advertised service. There are three types to advertise the services which are:

I. UA issues a request for a service, specifying the characteristics of the service, which the client requires. SLP allows a UA to directly send its requests to a SA. In this case, the request is send via multicast. The SA then advertises its service via unicast containing the location of the service.

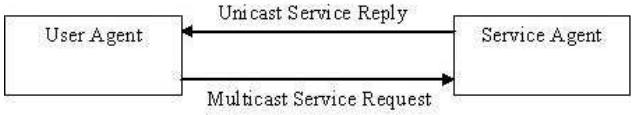

Figure 4. Case I: working of SLP

II. In larger networks, DAs can be used to function as cache. Server Agents send Service Registry messages to a DA containing all the services they advertise and receive acknowledgements in reply. These advertisements must be refreshed with the DA before they expire.

Figure 5. Case II: working of SLP

III. User and Server Agents discover Directory Agents in two ways. First, they send a multicast Service Request for the DA service when they start up. Second, the DA sends an unsolicited advertisement via multicast on random times, which the SA and UA listen for. In both cases, the Agents receive a DA advertisement.

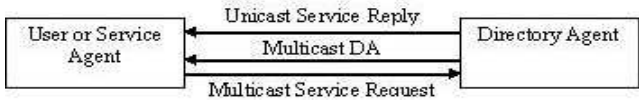

Figure 6. Case III: working of SLP

There two different methods for DA discovery: active and passive. In active discovery, UAs and SAs multicast SLP requests to the network. In passive discovery, DAs multicast advertisements for their services and continue to do this periodically in case any UAs or SAs have failed to receive the initial advertisement.

This protocol is scalable up to enterprise size networks because it provides capabilities for applications to register and deregister services and to locate services by type, by host, by attribute.

2.4Jini

Jini is based on the Java technology and extends the Java virtual machine (JVM) to services on the JVMs. It enables users to share services and resources over a network. It provides users easy access to resources anywhere on the network while allowing the network location of the user to change. It is used to turn the network into a flexible, easily administered tool with which resources can be found by human and computational clients. The focus of the system is to make the network a more dynamic entity that better reflects the dynamic nature of the workgroup by enabling the ability to add and delete services flexibly.

There are seven mechanisms:

1. Services

A Jini system consists of services that can be collected together for the performance of a particular task. Services in a Jini system communicate with each other by using a service protocol, which is a set of interfaces written in the Java programming language.

2. Lookup service

Services are found and resolved by a lookup service. A lookup service maps interfaces indicating the functionality provided by a service to sets of objects that implement the service.

3. Java remote method invocation (RMI)

E-ISSN: 2321–9637
Volume 2, Issue 1, January 2014
International Journal of Research in Advent Technology
Available Online at: http://www.ijrat.org

Communication between services can be accomplished using Java Remote Method Invocation. Java RMI provides mechanisms to find, activate, and garbage collect object groups.

4. Security

The design of the security model for Jini technology is built on the twin notions of a principal and an access control list. Jini services are accessed on behalf of some entity--the principal--which generally traces back to a particular user of the system. Services themselves may request access to other services based on the identity of the object that implements the service. Whether access to a service is allowed depends on the contents of an access control list that is associated with the object.

5. Leasing

Many of the services in the Jini system environment is lease based. A lease is a grant of guaranteed access over a time period. Each lease is negotiated between the user of the service and the provider of the service as part of the service protocol: A service is requested for some period; access is granted for some period. If a lease is not renewed before it is freed--either because the resource is no longer needed, the client or network fails, or the lease is not permitted to be renewed--then both the user and the provider of the resource may conclude that the resource can be freed. Leases are either exclusive or non-exclusive. Exclusive leases ensure that no one else may take a lease on the resource during the period of the lease; non-exclusive leases allow multiple users to share a resource.

6. Transactions

A series of operations, either within a single service or spanning multiple services, can be wrapped in a transaction. The Jini transaction interfaces supply a service protocol needed to coordinate a two-phase commit.

7. Events

An object may allow other objects to register interest in events in the object and receive a notification of the occurrence of such an event. This enables distributed event-based programs to be written with a variety of reliability and scalability guarantees.

The heart of the Jini system is a trio of protocols called discovery, join, and lookup. Pair of these protocols, discovery and join occurs when a device is plugged in.

Discovery occurs when a service is looking for a lookup service with which to register. Join occurs when a service has located a lookup service and wishes to join it. Lookup occurs when a client or user needs to locate and invoke a service described by its interface type.

Jini uses Java RMI for service invocation. Sun Microsystems has employed its Remote Method Invocation (RMI) technology. The primary Lookup Service behaviour is defined by the ServiceRegistrar interface which is implemented as an RMI proxy.

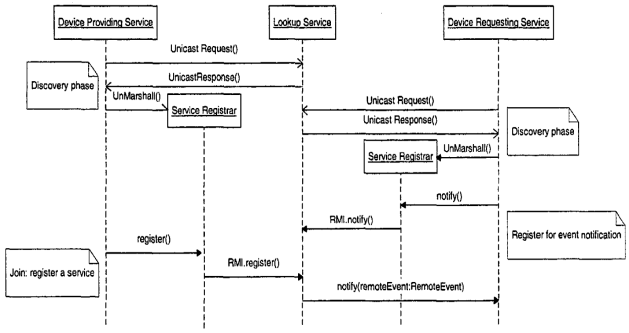

Figure7.RMI-based jini lookup service operation.

When a client device needs to use the Lookup Service (LUS), it downloads the ServiceRegistrar object. Execution of ServiceRegistrar methods is done via the proxy that makes RMI calls back to the device hosting the LUS.

From the service client's point of view, there is no distinction between services that are implemented by objects on a different machine, services that are downloaded into the local address space, and services that are implemented in hardware. All of these services will appear to be available on the network; will appear to be objects written in the Java programming language, and, only as far as correct functioning is concerned, one kind of implementation could be replaced by another kind of implementation without change or knowledge by the client.

E-ISSN: 2321–9637
Volume 2, Issue 1, January 2014
International Journal of Research in Advent Technology
Available Online at: http://www.ijrat.org

Downloadable code is also used for service representation. In this case the code downloaded from the device implements a graphical user interface to access or to configure the device.

2.5Avahi

The Avahi system facilitates service discovery on a local network. Avahi allows a new host on the local network to view other hosts and communicate with them (chat, find printers or shared files). Avahi runs in Linux.
Avahi has been developed by Lennart Poettering and Trent Lloyd. It allows programs to publish and discover services and hosts running on a local network with no specific configuration. It is licensed under the GNU Lesser General Public License (LGPL).
The major components of Avahi are as follows:
Libavahi-common: contains some functions used by the Avahi-daemon and the mDNS stack

- Libavahi-core: implements a flexible mDNS/DNS-SD stack and is highly used by developers.
- Avahi-daemon: implements the MDNS stack by using Libavahi-core.
- Libavahi-client: facilitates the use of DBUS API by hiding DBUS internals.

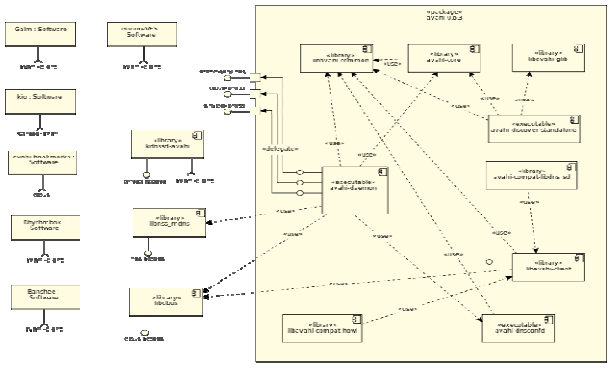

Figure8. Architectural view of Avahi

Avahi is a cross-platform software implementing the main functionalities of Zeroconf: multicast Domain Name System (mDNS), Domain Name System- Service Discovery (DNS-SD) and Internet Protocol version 4 Link Local addressing (IPv4LL). Avahi also allows programs to publish and discover services and hosts within a local network without any specific configuration.
There are two phases in the deployment of Avahi. The first phase involves deploying the Avahi client and second phase involves running the Avahi source code. Each of these phases requires proper configuration of a set of packages.

3. OVERALL SYSTEM APPROACH OF AVAHI

As we have seen Avahi is service discovery protocol. It is used with Linux, Ubuntu, and fedora. In Linux, Avahi packages are inbuilt given. The proposed algorithm for avahi is given figure 9.
As we have seen there are three main components of avahi. Those are avahi-core, avahi-daemon, avahi-client. Avahi daemon is the environment where the services are located and discovered. For that required packages are extracted

E-ISSN: 2321–9637
Volume 2, Issue 1, January 2014
International Journal of Research in Advent Technology
Available Online at: http://www.ijrat.org

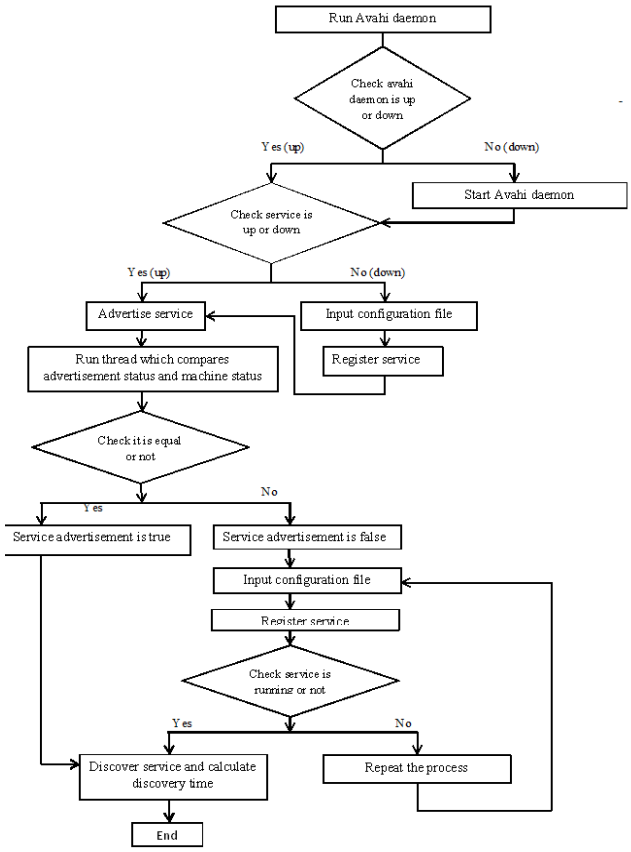

Figure9. Overall view of proposed system approach (dsAvahi algorithm)

There are many types of services on machine. Check the services whether they are running or not. If not, then extract packages, edit configuration file, register service. If service is running then check status of advertisement and status of machine. If they are same then advertisement is true and discovers the service. If service advertisement is false then input configuration file and register service.
Before the DSAvahi algorithm the system is

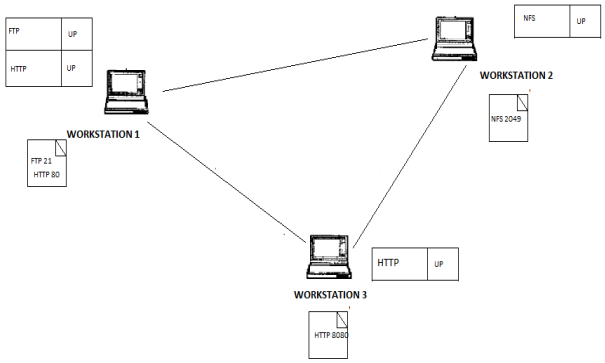

Figure 10. Scenario in the adhoc network

Three workstations are running with their own services. When avahi is installed means the devices are zero configuration enabled.

E-ISSN: 2321–9637
Volume 2, Issue 1, January 2014
International Journal of Research in Advent Technology
Available Online at: http://www.ijrat.org

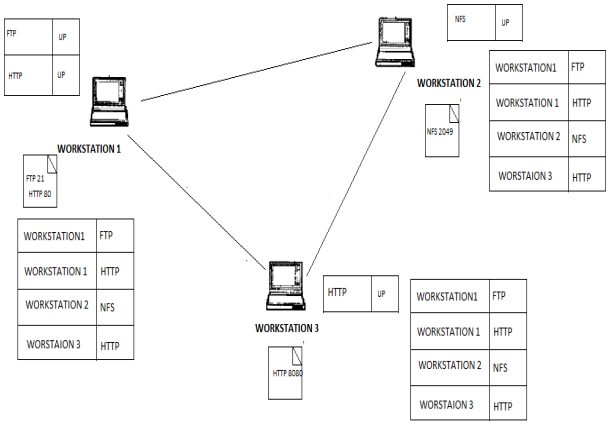

Figure11. System with Avahi

There are cases if
1. One of the services is down.
2. Asking for particular service but not providing data.
3. Check interdependency. (How many devices are using that service)
Check these conditions continuously or periodically. When any condition is true then after executing DSAvahi algorithm the system will be,

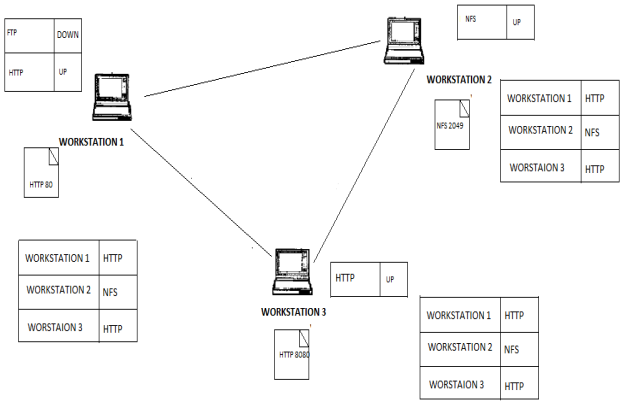

Figure12. System after failure

Remove the entry from the list of UP services of the workstations. Maintain the historic records for further use.To evaluate the performance of discovering services Grid5000 technology is used. The requirement of Grid 5000 is:

- Simple network configuration
- No degradation and resource security
- No need to re implement existing distributed applications.

N is number of workstation. Each node requests a registration for a given service at given time. Initially, all nodes have the needed codes to request a service but are inactive. Let α be the activation time. The kth request will be activated at time k×α.

Discovering time is the elapsed time between the end of the registration of a unique service and its discovery time by browser.

Note that the response time depends on the replicas number of a given services and the registered nodes. The browsing program listen any new event, i.e. a new registration or deleting services. We draw the chart where point (i, j, k) is the response time i for browser j when we use a total of k browsers.

Registration of one service on each machine, if the service is running, the machine is connected on the network otherwise (we de-activate the service) the machine is disconnected.

There are 2 types of registration. First sequential registration,

E-ISSN: 2321–9637
Volume 2, Issue 1, January 2014
International Journal of Research in Advent Technology
Available Online at: http://www.ijrat.org

- Every 60 sec, we activate a service
- Only multicast per minute

Second is Simultaneous registration,

- All services are activated on the same instant.
- N multicasts at a given time

Time of registration = Time between the demand and the receipt of notification.

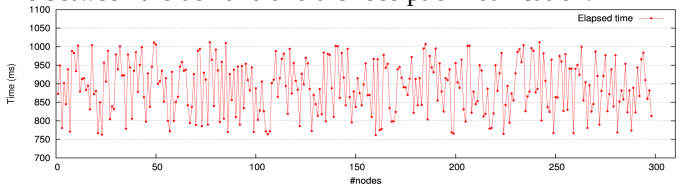

Figure13. Elapsed time for sequential registrations of Avahi services

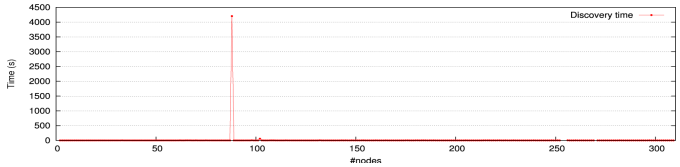

Figure14. Browse services after a sequential registrations

The elapsed time varies between 760 and 1110 ms.

Avahi loses almost 60% of registered services for workstations above 300. For lesser, the discovery time is much better (less than 2 seconds for all services). From the above theory we conclude that if there are two workstations (as per in our scenario) or nodes then result is,

Device name	No of devices	Average discovery time(ms)
Workstation Discovery	Workstation 1 (Zeroconf)	0.0166
	Workstation 2 (Zeroconf1)	0.0500

4. CONCLUSION AND FUTURE SCOPE

As we have seen there are number of service discovery protocols. User can select the protocol according to the need of network environment, number of workstations, type of service, size of TXT record, easy implementation.

It is used when the network is small like in a meeting room, sharing of music files between friends, easy printing by using printer etc. Typically users want easy way to access services when they are in small network. So all these protocol gives a big relief to users. We examined five protocols Bonjour, Avahi, jinni, SLP, UPnP. We studied implementation of avahi.

As the number of workstations will get increase so it will be difficult to handle the network. Until now it is restricted to very small number of machines and small network such as home network or private network. So the aim will be to increase the capacity of number of devices.

When services are listed they must be according to priority. Priority may be according to usage or user's choice. So an algorithm will be designed to set priority.

When workstations want services information is exchanged (e.g. files, images, and music files) so it should be provided to that workstation according to their choice of extension. Like for image file, there are different extensions so which type of file the user want or with which type workstation is compatible with.

The algorithm can be extended as after calculating the service discovery time filter the services and give an easy way to discover services which is according to user's choice. So there will be four cases:

- Categorize according to priority:

E-ISSN: 2321–9637
Volume 2, Issue 1, January 2014
International Journal of Research in Advent Technology
Available Online at: http://www.ijrat.org

1. Timely manner: Discover the services from 9a.m. to 11 a.m. or in office allow access to music files in only lunch time.
2. Frequently accessed.

- According to file type and extension.
- On the basis of load (Repeat BAND method can be used to distribute load).
- On the basis of minimum collision and power consumption (Konark, Deepspace, Bluetooth methods can be used).

References

[1] Stuart Cheshire, Bernard Aboba, and Erik Guttman. Dynamic Configuration of IPv4 Link-Local Addresses. RFC 3927, Internet Engineering Task Force, May 2005.
[2] Stuart Cheshire and Marc Krochmal. DNS-Based Service Discovery. Internet-Draft, Internet Engineering Task Force, June 2005.
[3] Stuart Cheshire and Marc Krochmal. Multicast DNS. Internet-Draft, Internet Engineering Task Force, June 2005
[4] Microsoft corporation, understanding UPnP: A White paper, http://www.upnp.org/download/UPNP_Understanding UPNP.doc
[5] Zeroconf working group. [Online]. Available: http://www.zeroconf.org/
[6] Bonjour home page. [Online]. Available: http://developer.apple.com/networking/bonjour/
[7] 29341-1:2008 Information technology - UPnP Device Architecture -- Part 1: UPnP Device Architecture Version 1.0 2008.ISO/IEC
[8] S. Cheshire, Zero Configuration Networking, http://www.zeroconf.org/
[9] S. Cheshire, D. Steinberg, Zero Configuration Networking – The Definitive Guide, O'Reilly & Associates, 2005
[10] S. Cheshire, Multicast DNS, http://www.multicastdns.org/
[11] S. Cheshire, DNS Service Discovery (DNS-SD), http://www.dns-sd.org/
[12] Sun Microsystems, Inc., Introduction to Jini, http://www.jini.org/wiki/Category:Introduction_to_Jini
[13] Automatic Configuration and Service Discovery for Networked Smart Devices by Günter Obiltschnig Applied Informatics Software Engineering GmbH.
[14] Study of Bojour by Apple, 2013-04-23.
[15] Apple Inc. [Online]. Available: http://www.apple.com/
[16.]Avahi. [Online]. Available: http://avahi.org/
[17] Debian GNU/Linux. [Online]. Available: http://www.debian.org/
[18]Ubuntu. [Online]. Available: http://www.ubuntu.com/
[19] Bonjour for Windows. [Online]. Available: http://www.apple.com/support/downloads/bonjourforwindows.html
[20] Analysis of peer to peer protocols, performance for establishing a decentralized desktop grid middleware by heithem abbes, jean Christophe dubacq, august 2008.
[21]http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_5-4/zero_config_networking.html
[22] A J I N I LOOKUPSE RVICE FOR RESOURCE-CONSTRAINED DEVICES Lawrence (Tim) Smith, Cameron Roe and Knud Steven Knudsen
[23] Automatic Discovery of Thin Servers: SLP, Jini and the SLP-Jini Bridge Erik Guttman, James Kempf Sun Microsystems
[24]UPnP: breaking out of the LAN by Jim Grimmett and Eamonn O'Neill
[25] Zeroconf and UPnP techniques by petri palmila
[26] Home networking and control based on UPnP: an implementation by Lu Yiqin , Fang Fang,Liu wei @2009