DATA CENTER AUTO DISCOVERY

Shynu Thankam Mathew, PG Student, Mtech in CSE with Specialisation in Cloud Computing, Vellore Institute of Technology, Chennai, shynu-thankam.mathew2015@vit.ac.in
Aparna V, Assitant Professor, Vellore Institute of Technology, Chennai,aparna.v@vit.ac.in

Abstract— As data growth is continuing, the number of data centers is also growing at an enormous rate and it has become a challenge to detect and manage the active devices or resources in a network especially for large organizations. Connectivity and management issues have become a great concern. Auto Discovery tool has greatly helped in this concern. It checks for the active devices in a network and provides with the basic configuration details (name, description, IP address, location, contact person, etc.). Not only the PC devices, its unique in a way that it also detects the network resources such as printers, switches, routers, and gateways. It means it can find information about any devices that has an IP address. All the detected devices are discovered by using any of the enabled discovery protocols – Simple Network Management Protocol (SNMP), Common Information Model (CIM), Windows Management Instrumentation (WMI). This feature is used in the NetApp's clustered Data ONTAP OS and OneCollect. With SNMP, it allows the network applications to study about the device sort and to send SNMP queries to the active devices. With this, it also helps to learn about the solution stack and identify the neighboring devices which a device is connected to.

Index Terms— Auto Discovery, SNMP, WMI

1.INTRODUCTION

1.1 Theoritical Background

Data Center is to store the systems, work station frameworks and their related devices or resources. It mostly incorporates reinforcement power supplies, excess information correspondences connections, ecological connections, and di erent security units. They range from a couple of servers in a space to immense independent structures measuring a huge number of square feet with a huge number of servers and other going with equipment. Their sizes and the sorts of hardware they contain change contingent on the necessities of the substance or elements they are supporting. The arrangement of servers, the system topology and the supporting gear can uctuate signi cantly relying on the organization, reason, area, development rate and starting plan idea of the server farm. Its format can incredibly in uence the pro ciency of information stream and the ecological conditions inside the middle. A few destinations may partition their servers into gatherings by capacities, for example, isolating web servers, application servers and database servers, and some may have each of its servers playing out various obligations. There are no immovable principles, and there aren't numerous official guidelines.

Because of the emerging technologies and the use of lesser and smaller hardware, there has been a greater demand for larger, powerful, and faster processing powers and data retrieval and storage. Mainly because of cloud computing technology, all the data is being stored in physical storage drives i.e., data centers. Absence of quick and depend-able access to information intend a powerlessness should gatherings give indispensable administrations or reduction from client satisfaction and income. It collects all the data from various companies. Since the growth rate of data increases, the need for more number of data centers have become an essential factor.

Emergence of cloud technology has migrated all the data and information from various organization to their own private, public or hybrid cloud. It means instead of housing the data in their own enterprise network, the data is being accessed through the service providers. All these data are stored in a remote location called data centers.

Because of such complex interconnections and tons of IT devices and resources, the management of the assets have become a tedious job for the data center executives.

Auto Discovery is a tool which helps to reduce this concern. It checks for the active devices in a network and provides with the basic con guration details (name, descrip-tion, IP address, location, contact person, etc.). Not only the PC devices, its unique in a way that it also detects the network resources such as printers, switches, routers, and gateways. It means it can nd information about any devices that has an IP address. All the detected devices are discovered by using any of the enabled discovery protocols Simple Network Management Protocol (SNMP), Common Information Model (CIM), Windows Management Instrumentation (WMI). This feature is used in the NetApp's clustered Data ONTAP OS and OneCollect. With SNMP, it allows the network applications to study about the device sort and to send SNMP queries to the active devices. With this, it also helps to learn about the solution stack and identify the neighboring devices which a device is connected to.

It helps to:

- Locate, monitor, and manage IT assets Collect data from them via protocols
- Know the entire network con gurations
- Know about the type of device and basic information content...

Auto discovery can be con gured in a single IP, IP range or subnet mask. It mainly uses SNMP i.e., Simple Network Management Protocol to detect the devices, their subnets, and routers. It sends SNMP queries to the device to retrieve the specific data or information that are con gured on that server. It scans the range of IP addresses to discover the active resources together with some of their essential data/information (name, description, IP address, location, contact person, etc.). All the available resources in the system is found by utilizing one of the discovery methods that the resource supports. The protocols used determine which TCP ports are active and checks whether they respond to the queries. Using this data, the kind of device can be found from the IP address - a Windows/Linux hosts, storage device, switches, routers etc.

Depending upon each device, the discovery uses different protocols or probes to obtain the data about the active device and the software used in them: Windows hosts or servers, the probe used is shell commands or WMI queries. For UNIX/Linux hosts, shell commands via SSH protocol is used. For storage devices, WBEM or CIM queries are used. For printers or routers, switches or uninterruptible power supplies, SNMP queries are used. For web servers, HTTP queries is used. The greater part of the PCs, switches, printers, et cetera that you have in your Con guration Management Database (CMDB) will appear each time you run Discovery. At that point there are those devices that you don't think about yet - resources as of late added to the system maybe - that are not in your CMDB. Given the circumstances, it bodes well that you ought to add those new gadgets to your database, and it too bodes well to overhaul existing gadgets, especially on the o chance that somebody has put in new memory, new programming, on the other hand included a plate drive. Discovery can do these things consequently. Discovery can dispatch tests that arrival character information from the device it finds.

1.2 Motivation for the proposed work

There are several reasons behind the motivation to do this work. Some of them are:

- For a storage admin in a datacenter, it is a tedious job to refer the device list and con gure systems for data collection.
- In some test labs and common lab infrastructures, the devices connected to each other may change frequently (e.g. switches connected to a controller may be swapped) which becomes very difficult.

This can be used by the other tools who wants to use this feature.

1.3 Aim for the proposed work

This paper aims to build a tool for data center device discovery service. This tool focus on data collection or con guration analysis of the system. It also provides health check results, performance metrics, capacity trending and other analytics. It scans the range of IP addresses to discover the active resources together with some of their essential data/information (name, description, IP address, location, contact person, etc.). All the available resources in the system is found by utilizing one of the discovery methods that the resource supports. The protocols used determine which TCP ports are active and checks whether they respond to the queries. It also ensures that other tools using this feature can also access the data by Django RESTful API.

2 Literature Survey

Various studies have been conducted in Auto Discovery of devices and their network topology and have shown great output; however, they fail to address many issues like discovering various types of devices, network topology visualization, discovering com-plete topology. This paper uses a simple algorithm to connect the end host with the network by using SNMP and CIM. It will be based on heuristics. Management infor-mation base(MIB) objects are required to implement this algorithm. These are then utilized to make a discovery algorithm, which is basically divided into three di erent units, namely discovery of gadgets, gadgets sort and its connectivity information. Vari-ous algorithms have been used for this auto discovery.

2.1 Ping/ Broadcast Ping

Each IP host is required to resound an ICMP "ping" packet back to its source. The ping utility consequently precisely demonstrates whether the pinged machine is on the Internet or not (really, since ping packet can get lost, we continually ping an address twice, considering it inaccessible just if both don't evoke an answer). With sensibly little parcels, ping moreover has a low overhead. Pings to dynamic hosts has prevail inside a solitary round-trip time, which is several numerous milliseconds, so the gadget is speedy. Pings to dead or non-existent hosts, regardless, timeout after a preservationist interval of 20 seconds, so pings to such has are exorbitant.

A broadcast ping is given to all hosts in the subnet, each of which should answer to originator of the ping. This is valuable in discovering every one of the machines in a subnet. Ping broadcast however is not bolstered completely in all systems. In a few systems, just the switch in charge of that subnet reacts to the communicate ping (we allude to this as the frail ping communicate supposition). In di erent systems, this ping is not reacted to by any stretch of the imagination.

2.2 Traceroute

Traceroute finds the course between a test point and an objective have by sending bundles with logically growing TTLs. Switches en route, on observing a bundle with a zero TTL, send ICMP TTL lapsed answers to the sender, which numbers these to discover the way. Traceroute is for the most part correct because all Internet changes are required to send the TTL-passed ICMP message. Regardless, some ISPs are known to cover their changes from traceroute by controlling these answers to fall their internal topology. This lessens both the precision and the culmination of topologies discovered using traceroute. Traceroute sends two tests to every switch en route, so it creates broadly more overhead than ping. Since tests to consecutive switches are scattered isolated to minimize the snappy system stack, a perfect chance to complete a traceroute is in like manner any more drawn out than a ping.

2.3 Zone Transfer from DNS Server

A domain's DNS name server keeps an o cial from each name in the domain to its IP address. Most DNS servers react to a 'zone exchange' command by giving back a list of each name in the domain. Therefore, DNS zone move is valuable in discovering all hosts and switches inside an area. This technique has low overhead, is fast and is exact. It may not be done, notwithstanding, since hosts getting IP addresses using DHCP are not served by DNS. Additionally, some system chiefs cripple DNS zone trade because of security concerns.

2.4 Auto-Discovery of Routers in Autonomic network

This approach was required for understanding the auto-discovery and auto-conguration includes in a GANA conformant organize. The approach sketched out is reasonable and pushes forward from the customary scripting and mechanization strategies utilized for conguration administration. Facilitate, the significant issue of system security, regularly sidelined amid a talk on conguration administration, is given fundamental significance. Subsequently a few security issues, for example, veri cation, get to control and trust are considered amid the plan of the empowering agents and calculations. In conclusion from the execution, this approach gave here is down to earth and effective.

2.5 Auto-Discovery at Network Layer

Auto-discovery capacities of administration frameworks ordinarily relate to network components in general, that is the capacity to naturally distinguish which network com-ponents are associated with a system and to nd their sort and physical and consistent design. Administration and network layer data, then again, is when all is said in done not found but rather provisioned and gave by the association working the system and services.

In general, auto-discovery concentrates at the system component administration layer. In any case, there are legitimate and imperative reasons to stretch out auto-discovery to higher administration layers too. The concepts talked about in this approach have been used by administration frameworks for Open Packet Telephony and for Metro Ethernet, with persuading comes about. The most vital angle maybe is a side e ect of the auto-discovery itself, to be speci c the capacity to distinguish organize layer irregularities in the system where bene t items are not ready to achieve a discovery state that indicates they are reliable and complete. Bene t layer auto-discovery is however a costly operation that ought to be utilized sparingly. It is connected once amid initial cool begin of the framework and consequently on administrator ask for; basically, in circumstances where the administration supplier is motivated to trust that the administration data is no more drawn out exact. A critical component of our frameworks is the capacity to con ne bene t layer auto-discovery to a certain extension, for example, in bundle communication a H.323 zone. The degree ought to obviously be characterized to such an extent that the service layer ideas inside it are independent, so that super uous and incorrect hailing of appearing irregularities - where service layer perspectives reach out to data outside the extension is avoided.

2.6 mDNS

Systems that support multicast DNS (mDNS) can resolve hostnames into IP addresses without the requirement for a Domain Name Server. The customer machine (e.g. your PC), searching for the given host's IP (Tesira server), sends a mDNS multicast (UDP port 5353) question message to every one of the hosts sharing its neighborhood arrange. At that point the comparing host answers with a multicast message that declares itself. With this answer, all machines in the subnet (other Tesira servers) can redesign their mDNS reserve with the given host's data (Host Names, IP addresses, and so forth.). The Hosts Table is likewise valuable when a Tesira framework traverses di erent subnets, and the Tesira servers in each subnet need to determine each other's hostnames without the assistance of a DNS server. Take note of that the Hosts Table is just valuable if IP locations are static; on the o chance that they are progressive (since they are being relegated by a DHCP server), then the data in the Hosts Table will get to be wrong when a Tesira server is appointed an alternate IP address by DHCP.

2.7 ARP

Device presence is at rst determined by perusing the ARP Cache table of a system framework gadget, (for example, a switch or switch). The ARP Cache table is perused from the gadget, and after that every gadget in the table is exclusively reached and found. This discovery technique gives the discovery motor an arrangement of devices to nd. Be that as it may, ARP Cache table entries are expelled after generally brief periods of latency. This implies the scan without anyone else's input doesn't know about inactive systems. This issue can be xed by sending a ping to every gadget in an objective network. This procedure invigorates the table, which has most the devices in it.

2.8 Service-Now

Windows MID Servers utilize the login details of the MID Server bene t on the host machine to nd Windows hosts in the system. This login is arranged when the MID Server is introduced and should have area or nearby manager bene ts. For Linux and UNIX machines and system gadgets, the MID Server utilizes the SSH and SNMP ac-creditations designed in the Service-Now case. MID Servers utilized by Orchestration must have entry to the important logins to execute summons on PCs in the system as determined by the Work ow exercises Orchestration can utilize the same SSH what's more, SNMP quali cations as Discovery, yet has two extra accreditations intended for Work ow exercises: Windows (for PowerShell) and VMware. For basic authentication, the elds available in the credentials form are those of Name, Username, and Password.

Device characterization happens just when a Service-Now Discovery Schedule is arranged to nd Con guration things. This output sort empowers Service-Now Discovery Iden-ti ers and is the main sweep that can be utilized to overhaul the CMDB. At the point when Service-Now Discovery has decided the gadget's class, it dispatches a personality test - a multiprobe - that is designed to run at least one orders with a solitary veri ca-tion. The character test in the out-of-box framework can be designed to approach the gadget for data, for example, its serial numbers (there can be more than one), name, and system recognizable proof. The aftere ects of this sweep are prepared by a character sensor, which then passes the outcomes to the Identi er. The Identi er then endeavors to locate a coordinating gadget in the CMDB. If the identifier nds a coordinating CI, the Identi er either upgrades that CI or does nothing. On the o chance that the identi er can't locate a coordinating CI, it either makes another CI or does nothing. If Discovery is arranged to proceed with, the Identifier dispatches the investigation tests designed in the Classi cation record to accumulate extra data about the gadget. Investigation tests can be multiprobes or basic tests.

3 Overview of the Proposed System

Auto Discovery tool is used to detect all the active devices in a network. This feature is being implemented in the NetApps OneCollect project. OneCollect project is a tool implemented for data collection. It collects the data from storage network area (SAN), network attached storage (NAS) environments. This collected data can be send back to NetApp for better support and assessment of operations, such as no disruptive updates and data migration. Data collection proceeds through multiple data collection protocols and posts the data to AutoSupport.

Solution-based collection pro le and device-based collection profile are two types of data collection pro les. Former is a prede ned environment and later allows the user to create their own environment. OneCollect compiles data about all the host operating systems, switches, storage arrays and other devices, which comprise most systems found in a typical enterprise ecosystem. Typical systems include Windows, Linux, VMware, FreeBSD, Solaris, AIX, HP-UX, Cisco IOS/NXOS/SANOS/ UCS, Brocade, NetApp Data ONTAP, E-Series, and others. It uses Telnet, SSH, and HTTPS protocols for communicating with systems and devices. To perform data collection against remote hosts, it is necessary that you provide administrative credentials. To store the credentials, OneCollect provides password-protected encryption (passphrase) to keep your credentials con dential. The credentials are stored in a database and encrypted using Advanced Encryption Standard (AES).

OneCollect enables data collection from different types of devices and solutions.

- Device or Component Based profile: This helps you collect data from single or mul-tiple homogenous/heterogeneous set of devices that could be storage controllers, switches, or hosts. You need to enter the credentials of the device or component for collecting data.
- Auto Discover profile: This helps you automatically discover devices in an IP, IP range, or subnet and collecting data.
- Solution Based profile: This helps you collect data from a set of precon gured solution stacks.

To build a tool for data center device discovery service. It checks for the active devices in a network and provides with the basic con guration details (name, description, IP address, location, contact person, etc.). Not only the PC devices, its unique in a way that it also detects the network resources such as printers, switches, routers, and gateways. It means it can nd information about any devices that has an IP address. This tool could focus on data collection or con guration analysis of systems. Collection of data should happen in di erent types of devices and solutions.

There are various reasons for the necessity of this tool in datacenter device discovery:

- Auto Discovery tool can be used for data collection to detect the type of devices and the connected devices.
- This tool helps the user to see the necessary information.
- Solutions should be available to identify devices independently or as part of other existing tools.
- Ability to access the entire network
- Ability to learn what devices are on the network and con rm the correct commu-nity string information.
- This utility could be built as a service that could be easily plugged into any solution requiring the feature and in addition provide the capability to identify connected devices and determine the solution stack (FlexPod, E-series SAN, Ontap SAN etc)

3.1 Simple Network Management Protocol (SNMP)

SNMP is a protocol used at the application-layer protocol in the TCP/IP protocol suite, its purpose is to monitor and retrieve the information about the management and pass it to other network devices. It gives you the information regarding the hardware and software details be it from any manufacturer. The transport-layer protocol SNMP uses is User Datagram Protocol (UDP). In other words, it is an Internet-standard conven-tion for gathering and sorting out data about oversaw devices on IP systems and for altering that data to change gadget conduct. Devices that regularly support SNMP incorporate switches, switches, servers, workstations, printers, modem racks and more. SNMP is broadly utilized as a part of system administration for system checking. SNMP uncovered administration information as factors on the oversaw frameworks sorted out in an administration data base which portray the framework status and arrangement. These factors can then be remotely questioned (and, in a few conditions, controlled) by overseeing applications. Three variants of SNMP have been created and conveyed. SN-MPv1 is the rst form of the convention. Later forms, SNMPv2c and SNMPv3, include upgrades in execution, adaptability, and security.

Basic things needed for SNMP to work is: SNMP manager, MIBs, and SNMP agents. SNMP is responsible for querying with other SNMP agents, getting responses from them and setting variables. By default, SNMP is disabled on all devices. If SNMP is enabled, the agent collects the MIB available so that it can be queried for whenever needed. MIB contains the information about the local environment and their management. It consists of some object identi ers that are unique and which speci es the characteristics of the device that is managed. They are structured hierarchically like a tree form in the MIB. It is represented as the list of dotted integers. e.g. 1.3.6.1.2.1.2.2. Each OIDs give specific information about the managed element. MIBs translates these OIDs to the SNMP manger so that the SNMP responses received from the element could be understood. Some of the basic OIDs used are:
'1.3.6.1.2.1.1.1.0, - System Description
'1.3.6.1.2.1.1.2.0', - System object identifier
'1.3.6.1.2.1.1.3.0', - System up time

'1.3.6.1.2.1.1.4.0', - System Contact
'1.3.6.1.2.1.1.5.0', - System Name
'1.3.6.1.2.1.1.6.0, - System location

SNMP also has community strings such as public, private that have the same purpose as passwords allowing only authorized users to access the information. The SNMP Community certi cation sort oversees access to nd numerous sorts of gadgets (switches, printers, and so on.) utilizing the SNMP convention. This quali cation sort is accessible for Discovery and Organization. Quali cations for SNMP do exclude a client name, only a secret key (the group string). The default perused just group string for some SNMP gadgets is open, and Discovery will attempt that consequently. Enter the tting SNMP quali cations if they vary from people in general group string. These elds are accessible in the Credentials frame for SNMP and they are Credential id, Username, Authentication protocol, Authentication key, Privacy protocol, Privacy key, tag and external storage. MIB les are maintained because it gives the basic information such as numeric OIDs, datatype, indexes for tables.

SNMP APISet object creation involve:

- Obtaining the SNMP constructor related inputs Version discovery.
- Loading CDEFs (MIB les dumped as CDEFs) Establishing the session

By using SNMP, auto discovery can nd servers and SAN in NetApp and EMC. It takes the following as the input:

- IP /IP range /subnet port
- Version
- Community string

3.2 Windows Management Instrumentation (WMI)

WMI protocol is used mainly by windows host implemented by Microsoft based on Web-Based Enterprise Management (WBEM) by adding some enhancements to the earlier version. This is used by not only by Windows but also by devices like routers, switches, storage arrays. By default, it is found in all windows systems. It uses CIM (Common Information Model) protocol as the underlying standard for windows hosts. Windows Management Instrumentation is a core Windows management technology; you can uti-lize WMI to oversee both local and remote PCs. WMI provides an unaltered approach to execute our day-to-day administration errands with programming or scripting lan-guages. Windows PowerShell is used to access the data via WMI. Its an administration innovation that permits programming to screen and control oversaw assets all through the system. Such oversaw assets incorporate hard drives, record frameworks, settings of working framework, forms, administrations, o ers, registry settings, organizing seg-ments, occasion logs, clients, bunches, and so forth. WMI permits observing of execution counters also. The reason for WMI is to characterize an exclusive arrangement of condi-tion free which permit administration data to be shared between administration appli-cations. WMI recommends endeavor administration principles and related advances for Windows that work with existing administration guidelines, for example, Desktop Man-agement Interface (DMI) and SNMP. WMI supplements these different principles by giving a uniform model. This model speaks to the oversaw condition through which administration information from any source can be gotten to normally. It takes username, password, and host as the input. Mostly used WMI Namespace is rootncimv2.

Some of the commonly used requirements of WMI are:

- Connection to a remote machine
- Connection to a remote machine as named computer or user Specific namespace connection

3.3 Web Framework – Django

The web framework used in this project is Django. Django is the server-side Python web framework. It helps in the development of secure websites rapidly. It has consistent design policies. It can be integrated with any client-side framework and give the result in any format. It can be run on any platform and on any versions of host systems. It supports many databases, caching and template engines. It helps in developing the program that is:

- Complete
- Versatile
- Secure
- Portable
- Scalable
- Maintainable

Working of Django:

HTTP requests is send from the controller to the application. It is send in the form of the URL with the needed parameters through HTTP POST or HTTP GET data. Based on what is needed, the computations are performed in the backend and the response is sent back to the web browser.

The files handled by the Django framework are:

- Urls.py - While it is conceivable to process demands from each URL by means of a solitary capacity, it is considerably more viable to compose a different view capacity to deal with every asset. A URL mapper is utilized to divert HTTP solicitations to the suitable view considering the demand URL. The URL mapper can likewise coordinate speci c examples of strings or digits that show up in a URL, and pass these to a view work as information.
- Views.py - A view is a demand handler work, which gets HTTP asks for and returns HTTP responses. They get to the information expected to fulfill demands by means of models, and delegate the organizing of the responses needed by templates.

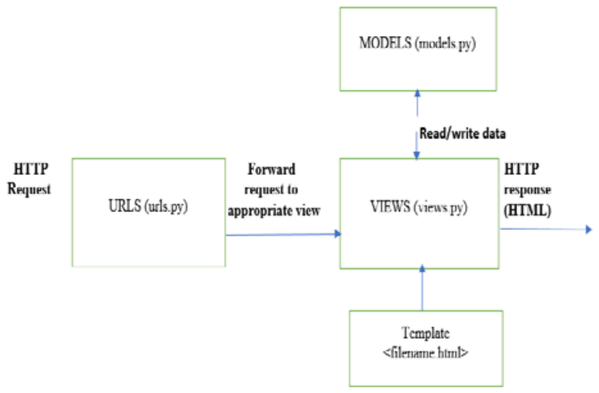


Fig 3.1 Flow of Django RESTful API

- Model.py - they contain Python objects that characterize the structure of an ap-plication's information, and give components to manage (include, change, erase) and inquiry records in the database.
- Html page - it is a content document characterizing the structure or design of a record, (for example, a HTML page), with placeholders used to speak to real content. A view can powerfully make a HTML page utilizing a HTML format, pop-ulating it with information from a model. A layout can be utilized to characterize the structure of a document; it doesn't need to be HTML!

Django can be:

- Installed on various working frameworks Used with Python 3 and Python 2
- Installed from source, from the Python Package Index (PyPi) and by and large from the host PC's bundle supervisor application
- Configured to utilize one of a few databases, which may likewise should be inde-pendently introduced and arranged
- Run in the fundamental framework Python condition or inside isolated Python virtual situations

3.4 Development Framework – AngularJS and JavaScript

AngularJS is used for the dynamic creation of web pages. UI web developers mostly use AngularJS, as it is a very powerful and interactive JavaScript framework. It uses a very modular approach and aims at the simpli cation of the code and easier to test. It is based on MVC (Model View Controller) structure. Google developed it. The libraries employed are based on JavaScript and Html with very few modi cations. JS is used to make it easier to use by making some modi cations to the existing content. Your application is characterized with modules that can depend from one to the others. It improves HTML by connecting orders to your pages with new properties or labels and expressions to characterize e ective layouts straightforwardly in your HTML. It additionally exempli es the conduct of your application in controllers which are instantiated because reliance infusion. Because the utilization of reliance infusion, AngularJS helps you structure and test your Javascript code e ortlessly. Finally, utility code can without much of a stretch be factorized into administrations that can be infused in your controllers.

4 Proposed System Design and Setup

4.1 Process Flow

Auto Discovery finds or searches for the active computers and other devices that are connected to an organizations network, and then updates or stores the MIB with those information.it uses di erent protocols to extract the information from the devices.

If device is windows host and servers then remote WMI commands are used. If storage, network gears (switches, routers, etc.), printers are used then SNMP queries are used. SNMPv2 is used in this discovery process.

Data collection using Auto discovery happens in the mentioned flow:

- User enters the IP, IP Range, or subnet in the field
- Request passed for SNMP and WMI queries
- If successful, device type and subtype are loaded
- Additional data include device details, location, system name, system uptime, system ID etc.
- If unsuccessful, manual mapping of device type and subtype is possible if the user recognizes the device.
- Select the devices for collecting and saving data
- Credentials are entered
- Devices are validated
- Device is saved and data is collected from it

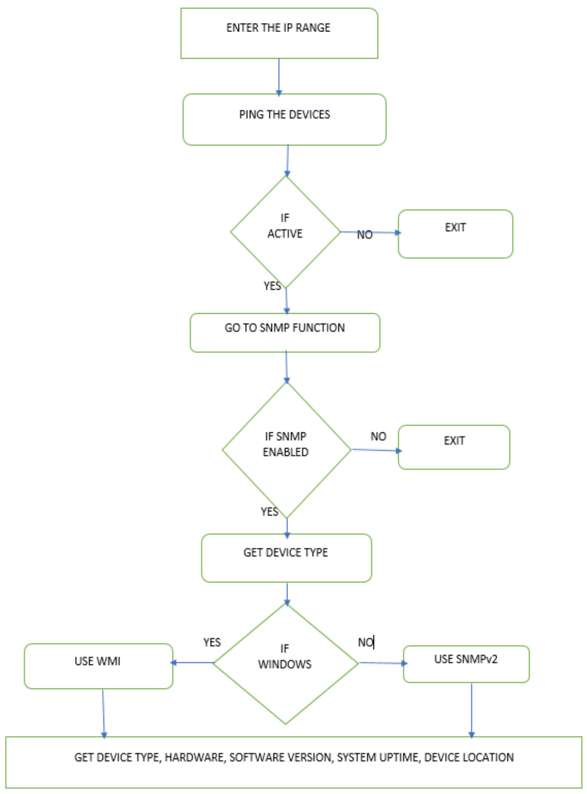


Fig 4.1 Data flow of Auto Discovery project

4.2 Project Modules

This project is divided into three phase:

• Discovery of devices using SNMPv2 network protocols • Discovery of devices using WMI network protocols • Creation of Django RESTful API services • Integration of API services with the GUI.

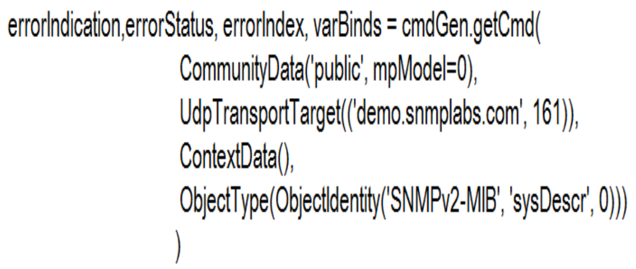
4.2.1 Discovery of devices using SNMPv2 network protocols

The main work done under this module is discovering the devices using SNMP queries. SNMP is a protocol used at the application-layer protocol in the TCP/IP protocol suite, its purpose is to monitor and retrieve the information about the management and pass it to other network devices. It gives you the information regarding the hardware and software details be it from any manufacturer. The transport-layer protocol SNMP uses is User Datagram Protocol (UDP).

Process flow in this module is:

- The user enters the IP, IP range or subnet in CIDR format. For example, 10.141.32.23 or 10.141.32.20.50 or 10.141.32.10/29.
- Using regex, the format of the IP entered is found and is directed to the appropriate function
- All the devices are pinged to find out the active devices in the network
- If active, then SNMP is queried to extract the details
- If SNMP is being enabled, the inventory details are extracted
- If is disabled, then no SNMP response is shown

Pseudocode for snmp:



```
errorIndication,errorStatus, errorIndex, varBinds = cmdGen.getCmd(
                CommunityData('public', mpModel=0),
                UdpTransportTarget(('demo.snmplabs.com', 161)),
                ContextData(),
                ObjectType(ObjectIdentity('SNMPv2-MIB', 'sysDescr', 0)))
                )
```

Fig 4.2 Pseudocode for SNMP connection

Note:

- CommunityData represents the community strings which can be either public or private.
- mpModel can be 0 or 1 depending on the version of SNMP used. Here 0 indicates SNMPv2
- UdpTransportTarget repsents the host and 161 is the UDP port number
- ObjectIdentity is the various OIDs used

OIDs for SNMP connection:

’1.3.6.1.2.1.1.1.0’, - System Description
’1.3.6.1.2.1.1.2.0’, - System object identifier
’1.3.6.1.2.1.1.3.0’, - System up time
’1.3.6.1.2.1.1.4.0’, - System Contact
’1.3.6.1.2.1.1.5.0’, - System Name
’1.3.6.1.2.1.1.6.0, - System location

4.2.2 Discovery of devices using WMI network protocols

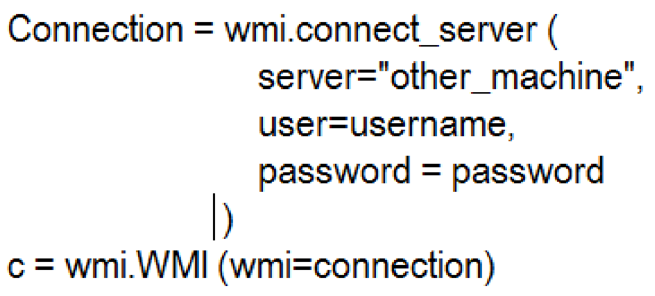
The main work done under this module is discovering the devices using WMI queries. WMI protocol is used mainly by windows host implemented by Microsoft based on WebBased Enterprise Management (WBEM) by adding some enhancements to the earlier version. This is used by not only by Windows but also by devices like routers, switches, storage arrays. By default, it is found in all windows systems. It uses CIM (Common Information Model) protocol as the underlying standard for windows hosts. Windows Management Instrumentation is a core Windows management technology; you can utilize WMI to oversee both local and remote PCs. WMI provides an unaltered approach to execute our day-to-day administration errands with programming or scripting languages.

Process flow in this module is:

- The user enters the IP, IP range or subnet in CIDR format. For example, 10.141.32.23 or 10.141.32.20.50 or 10.141.32.10/29.
- Using regex, the format of the IP entered is found and is directed to the appropriate function
- All the devices are pinged to find out the active devices in the network

- If active, then WMI is queried to extract the details
- WMI is enabled by default in all Windows devices

Pseudocode for WMI connection



```
Connection = wmi.connect_server (
                    server="other_machine",
                    user=username,
                    password = password
                )
c = wmi.WMI (wmi=connection)
```

Fig 4.3 Pseudocode for WMI connection

4.2.3 Creation of Django RESTful API services

The main work done under this module is to define the API services using Django RESTful services. This makes it is browsable on any web browsers and theres no need for consumer built application. So, it allows you to see the JSON responses. It is easy to study and follow. It supports serialization, pagination and allows to extend to other frameworks. They give us high performance web APIs and HTTP caching will be in optimal usage. So the working of the Django Python framework can be summarized as:

- Mapping the request to the urls
- Request handling in views
- Defining data objects in models
- Data querying in views
- Rendering data via templates

4.2.4 Integration of API services with the GUI

The main work done under this module is to integrate the Django RESTful API with the web framework AngularJS. AngularJS is used for the dynamic creation of web pages. UI web developers mostly use AngularJS, as it is a very powerful and interactive JavaScript framework. It uses a very modular approach and aims at the simplification of the code and easier to test. It is based on MVC (Model View Controller) structure.

4.3 Project Requirements

4.3.1 Protocols

This project uses the following protocols for device discovery in the organisations network:

- SNMP (Simple Network Management Protocol)
- WMI (Windows Management Instrumentation)

4.3.2 Software Requirements

- Programming Language: Python, Javascript
- Backend Frameworks: Django RESTful API
- Front-end Framework : AngularJS
- Operating System: Windows 7 (64-bit), Windows 8 (64-bit), Windows 10 (64-bit), Windows 2008 R2 Server (64-bit), Windows 2012 R2 Server (64-bit), Mac OS X 10.10 and later (64-bit), Red Hat Enterprise Linux (RHEL) 6.3 and later (64-bit), UBUNTU 7.0 and later (64-bit)
- Browser: Mozilla FireFox 35 and later, Google Chrome 43 and later, Apple Safari 9.1 and later, Microsoft Internet Explorer 10 and 11 (64-bit)

4.3.3 External Python Packages or Libraries

- pysnmp
- netaddr
- wmi
- pywbem
- texttable
- re

5 Test Cases and Results

5.1 Test Cases

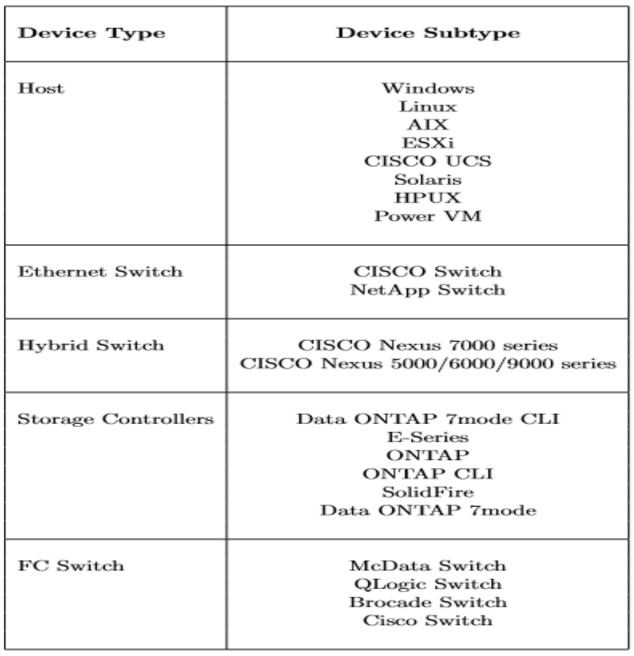
OneCollect is the project which focuses on data collection. Auto discovery is the tool used here for identifying the device in the network. Once the device is identified, the data is collected from it. It helps to understand the customer solution ecosystem

Devices from storage attached network(SAN) and network attached storage(NAS) are being tested. The devices such as storage systems, hosts ans switches are also discovered using uto discovery. There are two types of data collection profiles: Solution-based collection and Component-based collection. Former is a pre-defined environment and latter allows the user to create their own environment.

Main key feature is it collects the data from FlexPod, ONTAP, E-Series, SolidFire. It collects from host devices, FC and network related data from CISCO switches. Customer support use cases include collecting data from storage controllers, network switches, hosts and enterprise applications.

After the installation of the OneCollect, it automatcally launches on any default browser. If it does not, then user can manually enter thr URL in a browser: http://localhost:8044/oc/

The devices with their type and subtypes that are tested are listed in the following table:



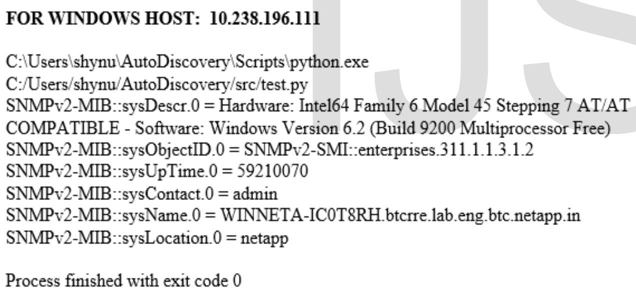
Device Type	Device Subtype
Host	Windows Linux AIX ESXi CISCO UCS Solaris HPUX Power VM
Ethernet Switch	CISCO Switch NetApp Switch
Hybrid Switch	CISCO Nexus 7000 series CISCO Nexus 5000/6000/9000 series
Storage Controllers	Data ONTAP 7mode CLI E-Series ONTAP ONTAP CLI SolidFire Data ONTAP 7mode
FC Switch	McData Switch QLogic Switch Brocade Switch Cisco Switch

TABLE 5.1: List of device types and subtypes

5.2 Results

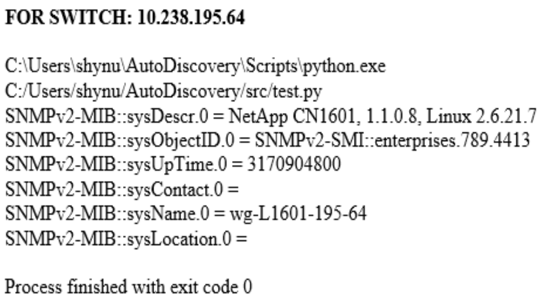
5.2.1 SNMP Responses

SNMP responses have been obtained from various devices. Some of the obtained responses are shown below:



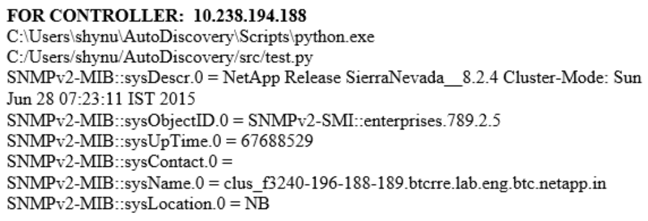
```
FOR WINDOWS HOST:  10.238.196.111

C:\Users\shynu\AutoDiscovery\Scripts\python.exe
C:/Users/shynu/AutoDiscovery/src/test.py
SNMPv2-MIB::sysDescr.0 = Hardware: Intel64 Family 6 Model 45 Stepping 7 AT/AT
COMPATIBLE - Software: Windows Version 6.2 (Build 9200 Multiprocessor Free)
SNMPv2-MIB::sysObjectID.0 = SNMPv2-SMI::enterprises.311.1.1.3.1.2
SNMPv2-MIB::sysUpTime.0 = 59210070
SNMPv2-MIB::sysContact.0 = admin
SNMPv2-MIB::sysName.0 = WINNETA-IC0T8RH.btcrre.lab.eng.btc.netapp.in
SNMPv2-MIB::sysLocation.0 = netapp

Process finished with exit code 0
```

Fig 5.1: Response from Windows Host



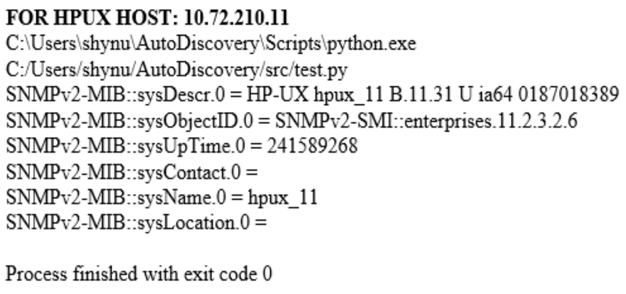
```
FOR SWITCH: 10.238.195.64

C:\Users\shynu\AutoDiscovery\Scripts\python.exe
C:/Users/shynu/AutoDiscovery/src/test.py
SNMPv2-MIB::sysDescr.0 = NetApp CN1601, 1.1.0.8, Linux 2.6.21.7
SNMPv2-MIB::sysObjectID.0 = SNMPv2-SMI::enterprises.789.4413
SNMPv2-MIB::sysUpTime.0 = 3170904800
SNMPv2-MIB::sysContact.0 =
SNMPv2-MIB::sysName.0 = wg-L1601-195-64
SNMPv2-MIB::sysLocation.0 =

Process finished with exit code 0
```

Fig 5.2: Response from Switch



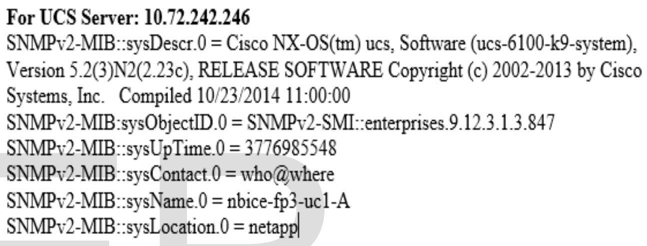
```
FOR CONTROLLER:  10.238.194.188
C:\Users\shynu\AutoDiscovery\Scripts\python.exe
C:/Users/shynu/AutoDiscovery/src/test.py
SNMPv2-MIB::sysDescr.0 = NetApp Release SierraNevada__8.2.4 Cluster-Mode: Sun
Jun 28 07:23:11 IST 2015
SNMPv2-MIB::sysObjectID.0 = SNMPv2-SMI::enterprises.789.2.5
SNMPv2-MIB::sysUpTime.0 = 67688529
SNMPv2-MIB::sysContact.0 =
SNMPv2-MIB::sysName.0 = clus_f3240-196-188-189.btcrre.lab.eng.btc.netapp.in
SNMPv2-MIB::sysLocation.0 = NB
```

Fig 5.3: Response from Controller



```
FOR HPUX HOST: 10.72.210.11
C:\Users\shynu\AutoDiscovery\Scripts\python.exe
C:/Users/shynu/AutoDiscovery/src/test.py
SNMPv2-MIB::sysDescr.0 = HP-UX hpux_11 B.11.31 U ia64 0187018389
SNMPv2-MIB::sysObjectID.0 = SNMPv2-SMI::enterprises.11.2.3.2.6
SNMPv2-MIB::sysUpTime.0 = 241589268
SNMPv2-MIB::sysContact.0 =
SNMPv2-MIB::sysName.0 = hpux_11
SNMPv2-MIB::sysLocation.0 =

Process finished with exit code 0
```

Fig 5.4: Response from HPUX Host



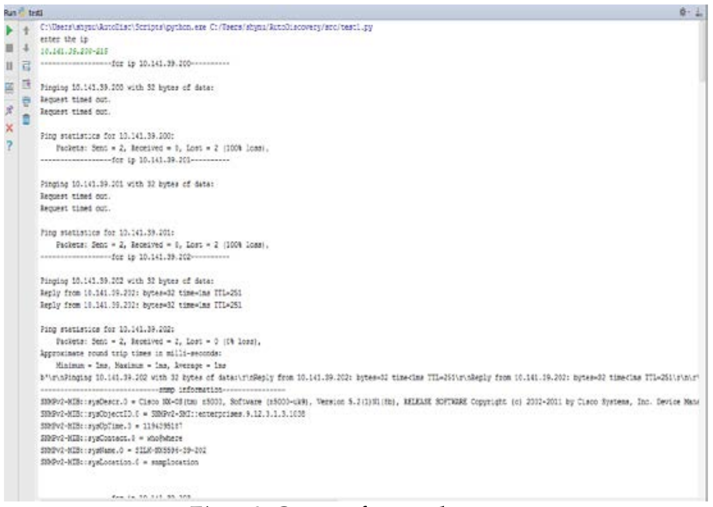
```
For UCS Server: 10.72.242.246
SNMPv2-MIB::sysDescr.0 = Cisco NX-OS(tm) ucs, Software (ucs-6100-k9-system),
Version 5.2(3)N2(2.23c), RELEASE SOFTWARE Copyright (c) 2002-2013 by Cisco
Systems, Inc.  Compiled 10/23/2014 11:00:00
SNMPv2-MIB:sysObjectID.0 = SNMPv2-SMI::enterprises.9.12.3.1.3.847
SNMPv2-MIB::sysUpTime.0 = 3776985548
SNMPv2-MIB::sysContact.0 = who@where
SNMPv2-MIB::sysName.0 = nbice-fp3-uc1-A
SNMPv2-MIB::sysLocation.0 = netapp
```

Fig 5.5: Response from UCS Server

5.2.2 Code Output

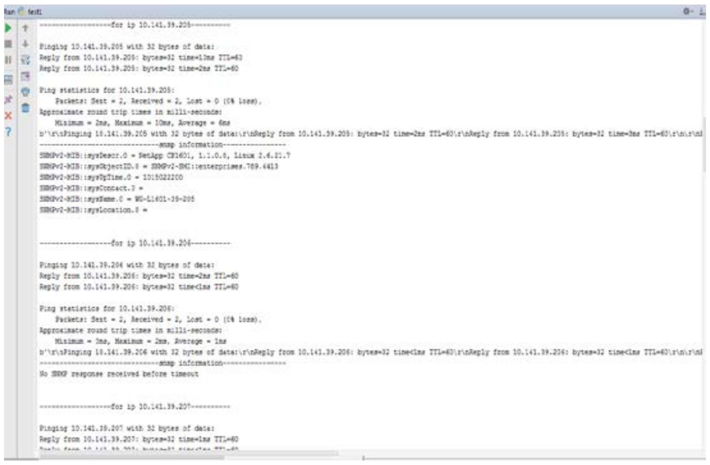
Here, the back-end code contains the code for pinging the device, as well as the connection code for SNMP and WMI. The input can be entered in IP address in the following format IP/ IP range/ subnet

In example, the IP is given in range 10.141.39.200-215. the output shows the ping status, snmp response and the result is tabulated.



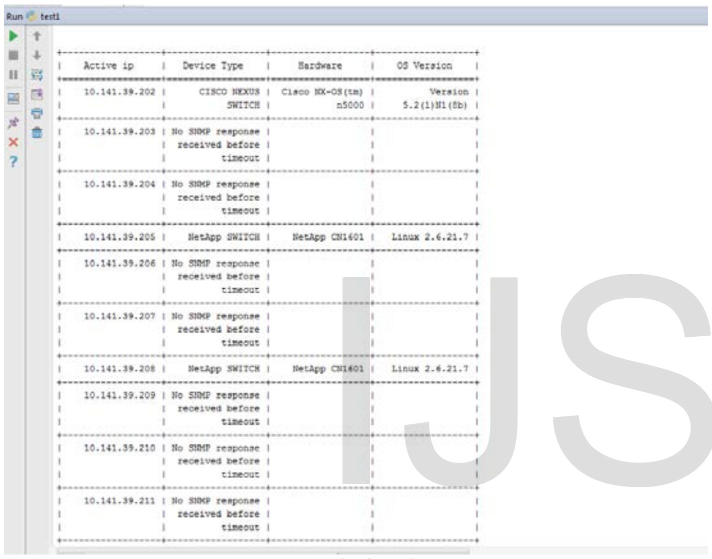
```
C:\Users\shynu\AutoDisc\Scripts\python.exe C:/Users/shynu/AutoDiscovery/src/test1.py
enter the ip
10.141.39.200-215
------------------for ip 10.141.39.200----------

Pinging 10.141.39.200 with 32 bytes of data:
Request timed out.
Request timed out.

Ping statistics for 10.141.39.200:
    Packets: Sent = 2, Received = 0, Lost = 2 (100% loss),
------------------for ip 10.141.39.201----------

Pinging 10.141.39.201 with 32 bytes of data:
Request timed out.
Request timed out.

Ping statistics for 10.141.39.201:
    Packets: Sent = 2, Received = 0, Lost = 2 (100% loss),
------------------for ip 10.141.39.202----------

Pinging 10.141.39.202 with 32 bytes of data:
Reply from 10.141.39.202: bytes=32 time<1ms TTL=251
Reply from 10.141.39.202: bytes=32 time<1ms TTL=251

Ping statistics for 10.141.39.202:
    Packets: Sent = 2, Received = 2, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 1ms, Average = 1ms
b'\r\nPinging 10.141.39.202 with 32 bytes of data:\r\nReply from 10.141.39.202: bytes=32 time<1ms TTL=251\r\nReply from 10.141.39.202: bytes=32 time<1ms TTL=251\r\n\r
------------------------------snmp information----------------
SNMPv2-MIB::sysDescr.0 = Cisco NX-OS(tm) n5000, Software (n5000-uk9), Version 5.2(1)N1(8b), RELEASE SOFTWARE Copyright (c) 2002-2011 by Cisco Systems, Inc. Device Man
SNMPv2-MIB::sysObjectID.0 = SNMPv2-SMI::enterprises.9.12.3.1.3.1038
SNMPv2-MIB::sysUpTime.0 = 1194395187
SNMPv2-MIB::sysContact.0 = who@where
SNMPv2-MIB::sysName.0 = SILK-NX5596-39-202
SNMPv2-MIB::sysLocation.0 = snmplocation


                  ---for ip 10.141.39.203
```

Fig 5.6: Output from code



```
------------------for ip 10.141.39.205----------

Pinging 10.141.39.205 with 32 bytes of data:
Reply from 10.141.39.205: bytes=32 time=13ms TTL=63
Reply from 10.141.39.205: bytes=32 time=2ms TTL=60

Ping statistics for 10.141.39.205:
    Packets: Sent = 2, Received = 2, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 2ms, Maximum = 10ms, Average = 6ms
b'\r\nPinging 10.141.39.205 with 32 bytes of data:\r\nReply from 10.141.39.205: bytes=32 time=2ms TTL=63\r\nReply from 10.141.39.205: bytes=32 time=3ms TTL=63\r\n\r\n
------------------------------snmp information----------------
SNMPv2-MIB::sysDescr.0 = NetApp CN1601, 1.1.0.6, Linux 2.6.21.7
SNMPv2-MIB::sysObjectID.0 = SNMPv2-SMI::enterprises.789.4413
SNMPv2-MIB::sysUpTime.0 = 1015022200
SNMPv2-MIB::sysContact.0 =
SNMPv2-MIB::sysName.0 = NG-L1601-39-205
SNMPv2-MIB::sysLocation.0 =


------------------for ip 10.141.39.206----------

Pinging 10.141.39.206 with 32 bytes of data:
Reply from 10.141.39.206: bytes=32 time=2ms TTL=60
Reply from 10.141.39.206: bytes=32 time<1ms TTL=60

Ping statistics for 10.141.39.206:
    Packets: Sent = 2, Received = 2, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 2ms, Average = 1ms
b'\r\nPinging 10.141.39.206 with 32 bytes of data:\r\nReply from 10.141.39.206: bytes=32 time<1ms TTL=60\r\nReply from 10.141.39.206: bytes=32 time<1ms TTL=60\r\n\r\n
------------------------------snmp information----------------
No SNMP response received before timeout


------------------for ip 10.141.39.207----------

Pinging 10.141.39.207 with 32 bytes of data:
Reply from 10.141.39.207: bytes=32 time=1ms TTL=60
```

Fig 5.7 Output showing ping status and SNMP response



```
+-----------------+-----------------+-----------------+-----------------+
|   Active ip     |   Device Type   |    Hardware     |   OS Version    |
+=================+=================+=================+=================+
|   10.141.39.202 |     CISCO NEXUS | Cisco NX-OS(tm) |         Version |
|                 |          SWITCH |           n5000 |   5.2(1)N1(8b)  |
+-----------------+-----------------+-----------------+-----------------+
|   10.141.39.203 | No SNMP response|                 |                 |
|                 |  received before|                 |                 |
|                 |          timeout|                 |                 |
+-----------------+-----------------+-----------------+-----------------+
|   10.141.39.204 | No SNMP response|                 |                 |
|                 |  received before|                 |                 |
|                 |          timeout|                 |                 |
+-----------------+-----------------+-----------------+-----------------+
|   10.141.39.205 |   NetApp SWITCH |   NetApp CN1601 |  Linux 2.6.21.7 |
+-----------------+-----------------+-----------------+-----------------+
|   10.141.39.206 | No SNMP response|                 |                 |
|                 |  received before|                 |                 |
|                 |          timeout|                 |                 |
+-----------------+-----------------+-----------------+-----------------+
|   10.141.39.207 | No SNMP response|                 |                 |
|                 |  received before|                 |                 |
|                 |          timeout|                 |                 |
+-----------------+-----------------+-----------------+-----------------+
|   10.141.39.208 |   NetApp SWITCH |   NetApp CN1601 |  Linux 2.6.21.7 |
+-----------------+-----------------+-----------------+-----------------+
|   10.141.39.209 | No SNMP response|                 |                 |
|                 |  received before|                 |                 |
|                 |          timeout|                 |                 |
+-----------------+-----------------+-----------------+-----------------+
|   10.141.39.210 | No SNMP response|                 |                 |
|                 |  received before|                 |                 |
|                 |          timeout|                 |                 |
+-----------------+-----------------+-----------------+-----------------+
|   10.141.39.211 | No SNMP response|                 |                 |
|                 |  received before|                 |                 |
|                 |          timeout|                 |                 |
+-----------------+-----------------+-----------------+-----------------+
```

Fig 5.8 Tabulated output

The below tabulated output indicates that 10.141.39.216 is a Linux Host and 10.141.39.220 is a controller. Other three devices give no snmp response as SNMP is disabled in those devices.

```
+-----------------+-----------------+-----------------+-----------------+
|   Active ip     |   Device Type   |    Hardware     |   OS Version    |
+=================+=================+=================+=================+
|   10.141.39.216 |           Linux |                 | 2.6.32-220.el6.x |
|                 |                 |                 |           86_64 |
+-----------------+-----------------+-----------------+-----------------+
|   10.141.39.217 | No SNMP response |                |                 |
|                 |  received before |                |                 |
|                 |          timeout |                |                 |
+-----------------+-----------------+-----------------+-----------------+
|   10.141.39.218 | No SNMP response |                |                 |
|                 |  received before |                |                 |
|                 |          timeout |                |                 |
+-----------------+-----------------+-----------------+-----------------+
|   10.141.39.219 | No SNMP response |                |                 |
|                 |  received before |                |                 |
|                 |          timeout |                |                 |
+-----------------+-----------------+-----------------+-----------------+
|   10.141.39.220 |      CONTROLLER |                 |  NetApp Release |
|                 |                 |                 |   8.2P2 Cluster- |
|                 |                 |                 |            Mode |
+-----------------+-----------------+-----------------+-----------------+
```

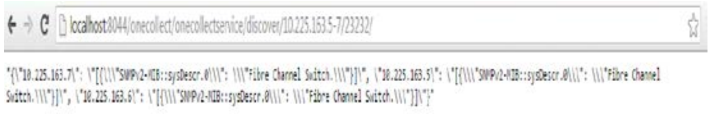
Fig 5.9 Tabulated response from various device

5.2.3 Django Responses

A Python web framework – Django RESTful API is created so that it can be used and integrated by any tool or browsers using the feature defined in this. The JSON response can be seen from the URL without having the need to built in any apps. It helps to view the response that API passes to the HTML.

Below are the JSON responses observed from some of the devices that are been tested. In the URL, the IP address and a unique parameter is appended. Unique parameter is used for the threading purposes where the SNMP responses of various devices can be accessed in parallel

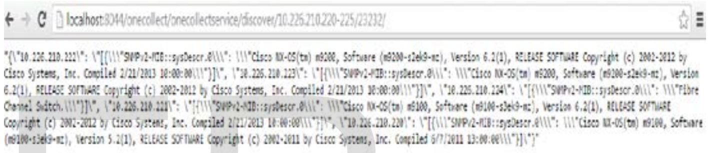
The figure shows the JSON response obtained from the range 10.225.163.5-7 which is an FC Switch


localhost:8044/onecollect/onecollectservice/discover/10.225.163.5-7/23232/

```
"{\"10.225.163.7\": \"[{\\\"SNMPv2-MIB::sysDescr.0\\\": \\\"Fibre Channel Switch.\\\"}]\", \"10.225.163.5\": \"[{\\\"SNMPv2-MIB::sysDescr.0\\\": \\\"Fibre Channel Switch.\\\"}]\", \"10.225.163.6\": \"[{\\\"SNMPv2-MIB::sysDescr.0\\\": \\\"Fibre Channel Switch.\\\"}]\"}"
```

Fig 5.10 JSON response from 10.225.163.5-7

The figure shows the JSON response obtained from the range 10.141.39.220-225


localhost:8044/onecollect/onecollectservice/discover/10.226.210.220-225/23232/

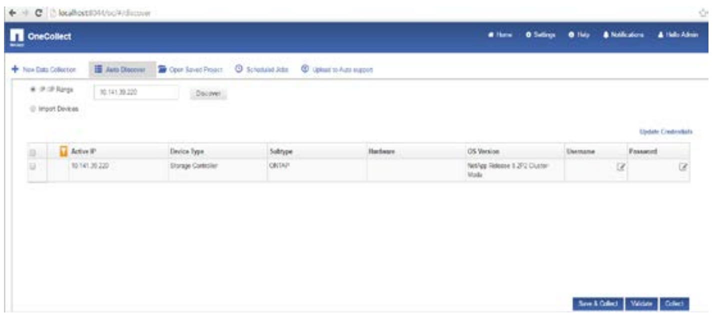
```
"{\"10.226.210.222\": \"[{\\\"SNMPv2-MIB::sysDescr.0\\\": \\\"Cisco NX-OS(tm) m9200, Software (m9200-s2ek9-mz), Version 6.2(1), RELEASE SOFTWARE Copyright (c) 2002-2012 by Cisco Systems, Inc. Compiled 2/21/2013 10:00:00\\\"}]\", \"10.226.210.223\": \"[{\\\"SNMPv2-MIB::sysDescr.0\\\": \\\"Cisco NX-OS(tm) m9200, Software (m9200-s2ek9-mz), Version 6.2(1), RELEASE SOFTWARE Copyright (c) 2002-2012 by Cisco Systems, Inc. Compiled 2/21/2013 10:00:00\\\"}]\", \"10.226.210.224\": \"[{\\\"SNMPv2-MIB::sysDescr.0\\\": \\\"Fibre Channel Switch.\\\"}]\", \"10.226.210.221\": \"[{\\\"SNMPv2-MIB::sysDescr.0\\\": \\\"Cisco NX-OS(tm) m9100, Software (m9100-s3ek9-mz), Version 6.2(1), RELEASE SOFTWARE Copyright (c) 2002-2012 by Cisco Systems, Inc. Compiled 2/21/2013 10:00:00\\\"}]\", \"10.226.210.220\": \"[{\\\"SNMPv2-MIB::sysDescr.0\\\": \\\"Cisco NX-OS(tm) m9100, Software (m9100-s3ek9-mz), Version 5.2(1), RELEASE SOFTWARE Copyright (c) 2002-2011 by Cisco Systems, Inc. Compiled 6/7/2011 13:00:00\\\"}]\"}"
```

Fig 5.11 JSON response from various devices

5.2.3 GUI Results

This section shows the GUI part or the final outcome of the Auto Discovery project. AngularJS is used for developing this dashboard. There are five tabs for this project. Each selected tab will give a grey highlight at the background.

User can enter the IP address in the textbox given. If the format is wrong an error will be displayed. In the below figure, a device is discovered for a single IP address. SNMP response gives the OS version.

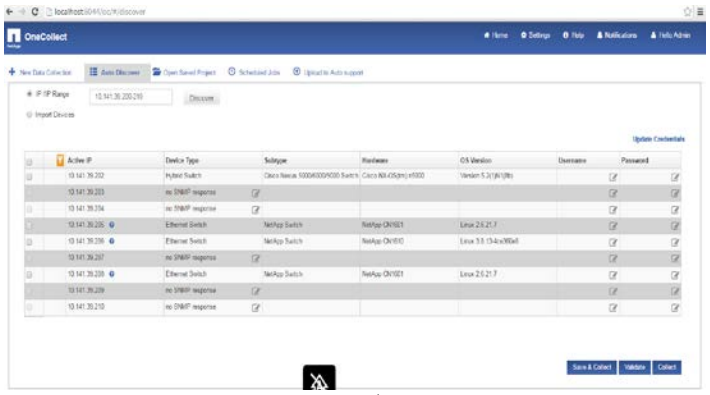


IP/IP Range: 10.141.39.220

Active IP	Device Type	Subtype	Hardware	OS Version	Username	Password
10.141.39.220	Storage Controller	ONTAP		NetApp Release 8.2P2 Cluster-Mode		

Fig 5.12 Discovery for a single IP

In the below figure, the discovery happens for a range of IPs: 10.141.39.220-210



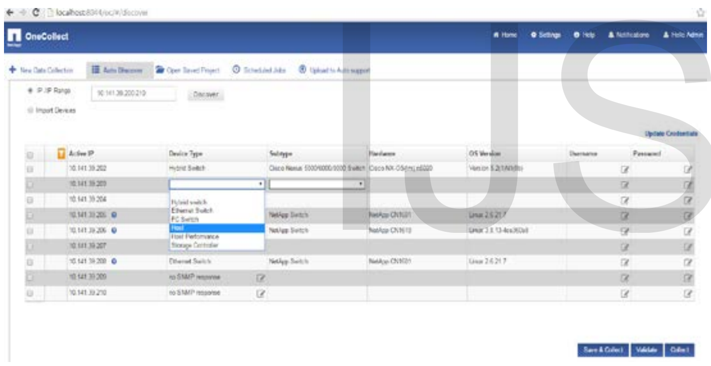
Active IP	Device Type	Subtype	Hardware	OS Version	Username	Password
10.141.39.202	Hybrid Switch	Cisco Nexus 5000/6000/9000 Switch	Cisco NX-OS(tm) n5000	Version 5.2(1)N1(8b)		
10.141.39.203	no SNMP response					
10.141.39.204	no SNMP response					
10.141.39.205	Ethernet Switch	NetApp Switch	NetApp CN1601	Linux 2.6.21.7		
10.141.39.206	Ethernet Switch	NetApp Switch	NetApp CN1610	Linux 3.8.13-4ce360e8		
10.141.39.207	no SNMP response					
10.141.39.208	Ethernet Switch	NetApp Switch	NetApp CN1601	Linux 2.6.21.7		
10.141.39.209	no SNMP response					
10.141.39.210	no SNMP response					

Fig 5.13 Discovery for an IP range

Note:

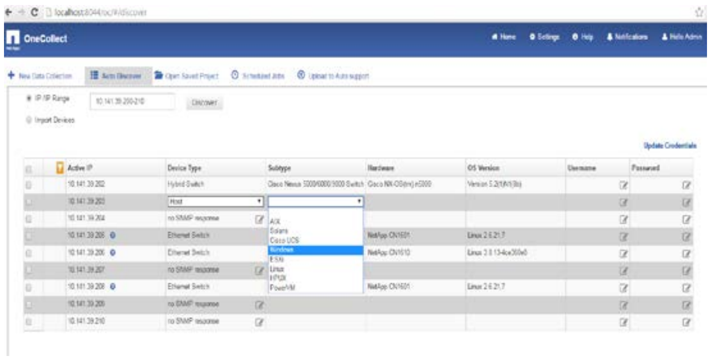
- no snmp response is shown if the snmp is not enabled for active devices.
- Against such cases an edit button is given so that if the user knows which device it is, he can edit it.
- A blue question mark is shown for the devices which have privileged password
- If username and password if known, then it can be entered and validated.

If "edit" button to the right is clicked then the device and its subtype can be chosen, if the customer recognizes it.



Active IP	Device Type	Subtype	Hardware	OS Version	Username	Password
10.141.39.202	Hybrid Switch	Cisco Nexus 5000/6000/9000 Switch	Cisco NX-OS(tm) n5000	Version 5.2(1)N1(8b)		
10.141.39.203						
10.141.39.204						
10.141.39.205		NetApp Switch	NetApp CN1601	Linux 2.6.21.7		
10.141.39.206		NetApp Switch	NetApp CN1610	Linux 3.8.13-4ce360e8		
10.141.39.207						
10.141.39.208	Ethernet Switch	NetApp Switch	NetApp CN1601	Linux 2.6.21.7		
10.141.39.209	no SNMP response					
10.141.39.210	no SNMP response					

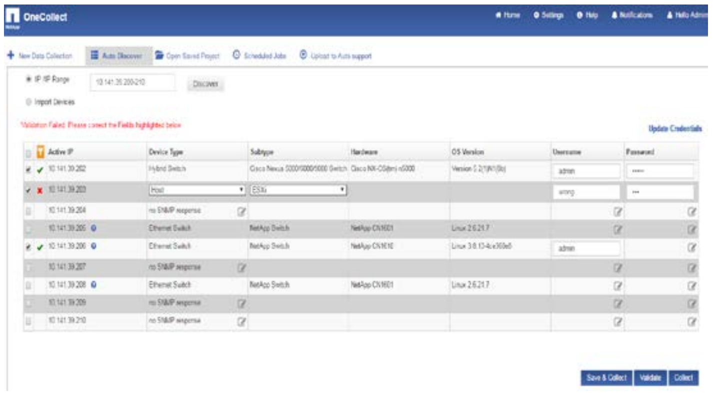
Fig 5.14 Editing device type



Active IP	Device Type	Subtype	Hardware	OS Version	Username	Password
10.141.39.202	Hybrid Switch	Cisco Nexus 5000/6000/9000 Switch	Cisco NX-OS(tm) n5000	Version 5.2(1)N1(8b)		
10.141.39.203	Host					
10.141.39.204	no SNMP response					
10.141.39.205	Ethernet Switch		NetApp CN1601	Linux 2.6.21.7		
10.141.39.206	Ethernet Switch		NetApp CN1610	Linux 3.8.13-4ce360e8		
10.141.39.207	no SNMP response					
10.141.39.208	Ethernet Switch		NetApp CN1601	Linux 2.6.21.7		
10.141.39.209	no SNMP response					
10.141.39.210	no SNMP response					

Fig 5.15 Editing device subtype

The user can enter the credentials in the textbox given by clicking the "edit" icon which is given to the right of it. Then the user can select the devices needed for data collection by selecting the checkboxes. On clicking "Validate", the loading mask appears while validation computes at the back-end. Below figure shows the result of validation. Green tick mark appears if validation is successful and red cross mark appears if validation is wrong. An error message will be appear on hovering on the red cross mark.

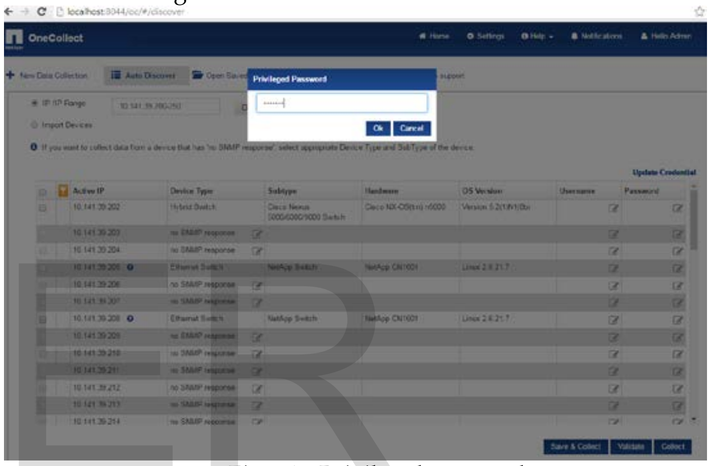


Validation Failed. Please correct the Fields highlighted below

	Active IP	Device Type	Subtype	Hardware	OS Version	Username	Password
✓	10.141.39.202	Hybrid Switch	Cisco Nexus 5000/6000/9000 Switch	Cisco NX-OS(tm) n5000	Version 5.2(1)N1(8b)	admin	·····
✗	10.141.39.203	Host	ESXi			wrong	···
	10.141.39.204	no SNMP response					
	10.141.39.205	Ethernet Switch	NetApp Switch	NetApp CN1601	Linux 2.6.21.7		
✓	10.141.39.206	Ethernet Switch	NetApp Switch	NetApp CN1610	Linux 3.8.13-4ce360e8	admin	
	10.141.39.207	no SNMP response					
	10.141.39.208	Ethernet Switch	NetApp Switch	NetApp CN1601	Linux 2.6.21.7		
	10.141.39.209	no SNMP response					
	10.141.39.210	no SNMP response					

Fig 5.16 Validation result

A "blue question mark" can be seen next to Ethernet Switch. When the blue question mark is clicked a dialog box appears to enter the Privileged Password.

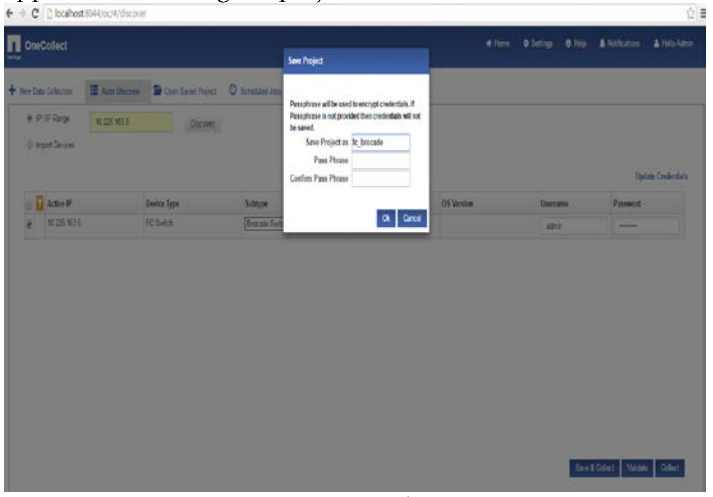


If you want to collect data from a device that has 'no SNMP response' select appropriate Device Type and SubType of the device.

Active IP	Device Type	Subtype	Hardware	OS Version	Username	Password
10.141.39.202	Hybrid Switch	Cisco Nexus 5000/6000/9000 Switch	Cisco NX-OS(tm) n5000	Version 5.2(1)N1(8b)		
10.141.39.203	no SNMP response					
10.141.39.204	no SNMP response					
10.141.39.205	Ethernet Switch	NetApp Switch	NetApp CN1601	Linux 2.6.21.7		
10.141.39.206	no SNMP response					
10.141.39.207	no SNMP response					
10.141.39.208	Ethernet Switch	NetApp Switch	NetApp CN1601	Linux 2.6.21.7		
10.141.39.209	no SNMP response					
10.141.39.210	no SNMP response					
10.141.39.211	no SNMP response					
10.141.39.212	no SNMP response					
10.141.39.213	no SNMP response					
10.141.39.214	no SNMP response					

Fig 5.17 Privileged password

On clicking "Save & Collect" button, a prompt window appears for entering the project to be saved under a name.



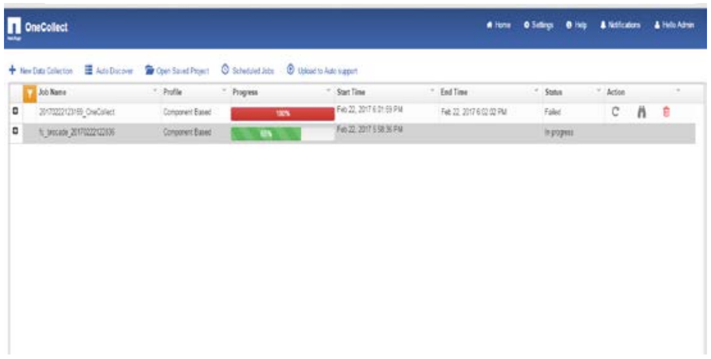
Passphrase will be used to encrypt credentials. If Passphrase is not provided then credentials will not be saved.

Save Project as	fc_brocade
Pass Phrase	
Confirm Pass Phrase	

Active IP	Device Type	Subtype	OS Version	Username	Password
10.225.163.6	FC Switch	Brocade Swi		admin	······

Fig 5.18 Saving the project

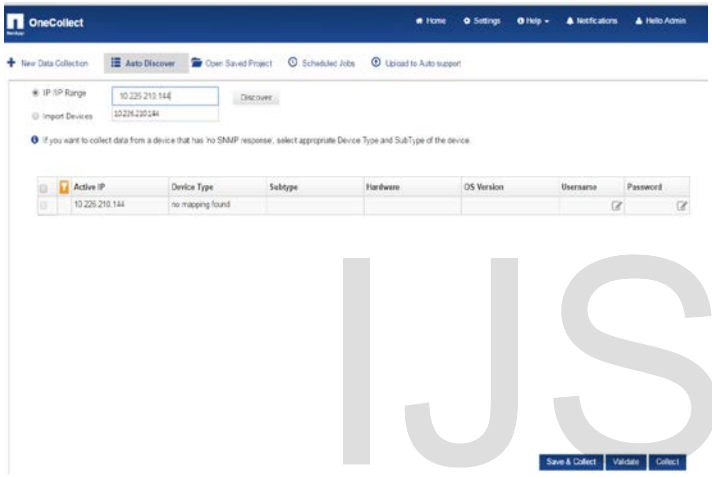
Data is being collected from the saved project



Job Name	Profile	Progress	Start Time	End Time	Status	Action
20170222123159_OneCollect	Component Based	100%	Feb 22, 2017 6:01:59 PM	Feb 22, 2017 6:02:02 PM	Failed	
fc_brocade_20170222122936	Component Based	63%	Feb 22, 2017 5:58:36 PM		In progress	

Fig 5.19 Data collection status

If the SNMP gives the error status while discovering the device then it does not give any other details. So, in the table for the corresponding device it will be shown as no mapping found. For such cases data, cannot be collected.



IP /IP Range: 10.225.210.144
Import Devices: 10.226.210.144

If you want to collect data from a device that has 'no SNMP response', select appropriate Device Type and SubType of the device

Active IP	Device Type	Subtype	Hardware	OS Version	Username	Password
10.226.210.144	no mapping found					

Fig 5.20 Error status in SNMP connection

6 Conclusion

By the end of this project a tool has been successfully developed to discover all the active devices in a a data center by using different protocols that are enabled in the devices. From their responses, the inventory information such as the device type, subtype, OS version, Hardware type, their built numbers have been extracted. The main two protocols used in this is SNMP and WMI protocols.

The device has been discovered by just entering a single IP, a range of IPs or subnet. From each discovered device, data has been collected successfully.

With the use of Django RESTful API, this feature can be integrated with any tool that wants to use this feature. This feature is available in all browsers. It supports many OS versions also.

This feature will enable the user to understand the entire network in an organization and access necessary information about those devices. This feature saves a lot of time for the datacenter admin who otherwise must make a list od devices in the network manually.

Acknowledgment

Primarily, I praise & hearts fully thank the Almighty god, from bottom of my heart, which has been the unfailing source of strength, comfort & inspiration in the completion of this project work.
I wish to express my sincere thanks to Dr. G. Viswanathan, Chancellor, Mr.Sankar Viswanathan, Vice President, Ms. Kadhambari S. Viswanathan, Assistant Vice President, Dr. Anand A. Samuel, Vice Chancellor and Dr. P. Gunasekaran, Pro-Vice Chancellor for providing me an excellent academic environment and facilities for pursuing M.Tech. program.
I am grateful to Dr. Vaidehi Vijayakumar, Dean of School of Computing Science and Engineering, VIT University, Chennai and to Dr. V. Vijayakumar, Associate Dean. I wish to express my sincere gratitude to Dr B.Rajesh Kanna, Program chair of M.Tech CSE CloudComputing for providing me an opportunity to do my project work.
I would like to express my gratitude to my internal guide Prof.Aparna V and my external guide Mr.Nithin Radhakrishnan who inspite of their buy schedule guided me in the correct path.
I am thankful to NetApp, Bangalore for giving me an opportunity to work on my project and helped me gain knowledge.
I thank my family and friends who motivated me during the project work

References

[1] Cisco, SNMP Community String Indexing, http://www.cisco.com/warp/public/ 477/SNMP/camsnmp40367.html.
[2] R.Siamwalla, R. Sharma, and S. Keshav, Discovering internet topology, Cornell Univ., Ithaca, NY, Techical Report, May 1999.
[3] Suman Pandey, Mi-Jung Choi, Sung-Joo Lee, James W.Hong, IP Network Topology Discovery Using SNMP ,November 2009
[4] http://mysupport.netapp.com/onecollect
[5] B. Lowekamp, D. R. OHallaron, T. R. Gross, Topology discovery for large Ethernet networks, ACM SIGCOMM, August 2001, San Diego, CA, USA, pp. 237 248.
[6] https://wikid.netapp.com/w/QA/projects/Libraries Initiative/LibrariesArchitecture/ APILayer/SNMP Design Spec
[7] https://wikid.netapp.com/w/Project OneCollect
[8] https://wikid.netapp.com/w/SNMP
[9] G.R. Malan and F. Jahanian. An extensible probe architecture for network protocol performance measurement, Proceedings of SIGCOMM'98, Sept. 1998, Online version available from http://www.merit.edu/ipma/docs/paper.html
[10] S. Seshan, M. Stemm, and R.H. Katz, SPAND: Shared Passive Network Performance Discovery, Proc 1st Usenix Symposium on Internet Technologies and Systems (USITS '97), Monterey, CA, December 1997
[11] R. Siamwalla, R. Sharma, and S. Keshav, “Discovering Network Topology”, IEEE INFOCOM’99
[12] Tayal, A.P., Patnaik, L.M.: An Address Assignment for the Automatic Configuration of Mobile Ad Hoc Networks. Personal Ubiquitous Comput. 8(1), 47–54 (2004)
[13] Harrington, D., Presuhn, R., Wijnen, B.: An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks. RFC 3411 (Standard) (December 2002), http://www.ietf.org/rfc/rfc3411.txt; updated by RFCs 5343, 5590
[14] Derbel, H., Agoulmine, N., Sala¨un, M.: ANEMA: Autonomic network management architecture to support self-configuration and self-optimization in IP networks. Comput. Netw. 53(3), 418–430 (2009)

[15] Suman Pandey, Mi-Jung Choi, Sung-Joo Lee, James W.Hong, "IP Network Topology Discovery Using SNMP" ,November 2009
[16] Y. Breitbart, M. Garofalakis, B. Jai, C. Martin, R. Rastogi, A. Silberschatz, "Topology Discovery in Heterogeneous IP Networks: The NetInventory System," IEEE/ACM Transactions on Networking, vol. 12, no. 3, June 2004, pp. 401~414.
[17] B. Lowekamp, D. R. O'Hallaron, T. R. Gross, "Topology discovery for large Ethernet networks," ACM SIGCOMM, August 2001, San Diego, CA, USA, pp. 237~248.
[18] F. Nazir, T.H. Tarar, F. Javed, H. Suguri, H.F. Ahmad, A. Ali, "Constella: A Complete IP Network Topology Discovery Solution," APNOMS 2007, October 2007, Sapporo, Japan, pp. 425~436.
[19] https://wiki.opennms.org/wiki/Discovery
[20] http://docs.librenms.org/Extensions/Auto-Discovery/
[21] http://docs.device42.com/auto-discovery/snmp-sanserver-auto-discovery/