
ISSN: 2321-2152
IJMECE
International Journal of modern
electronics and communication engineering
E-Mail
editor.ijmece@gmail.com
editor@ijmece.com
www.ijmece.com

ISSN 2321-2152
www.ijmece.com
Vol 7, Issue 2, 2019

NOVEL, SWITCH PORT MAPPER NETWORK DISCOVERY TOOL FOR ROUTNG TECHNICS

Tata Jagannadha Swamy
Professor
Department of Electronics and Communication Engineering
Gokaraju Rangaraju Institute of Engineering and Technology, Bachyupally, Hyderabad-90
Jagan.tata@griet.ac.in

Abstract: Abstract: The Switch Port Mapper can discover the ports on any manageable Cisco Catalyst Switch and detail the devices connected to those ports by MAC address, IP address and hostname. This tool eliminates tracing of LAN cables to determine the physical location of devices and makes the network documentation easy.The port mapping is done by discovering and correlating Port/MAC/Interface information from the switch. The MAC/IP address information is discovered from a Router that is directly connected to the same subnet as the switch. SNMP (Simple Network Management Protocol), Perl language and HTML/JavaScript are used to develop this Research Paper. SNMP protocol collects the required Port/MAC/Interface information from the switch; Perl is used to run these SNMP queries and HTML/Javascript to display the results. Since this is a real-time discovery you can view the operational status and port speed of each port. The Output can be exported to Microsoft Excel® or mySQL database.

Keywords: MAC address, Perl Language, SNMP, HTML, mySQL

1. Introduction

In a switched network environment where hundreds of PCs get connected to tens of Layer-2 switches, Network Administrator often finds it difficult to trace a PC using its IP or MAC address. To get the info about which port the PC is on, he has to login to each and every Switch and run few commands to determine the port. This will be a time-consuming process if the number of switches increases beyond five. Switch Port Mapper does the required mapping within seconds. It maintains the database of the Switched Network by querying the information from the Switches and Routers at periodic intervals and provides a user-friendly interface to fetch and display the data at the click of the mouse. The first chapter focuses on various technologies used to develop this tool that helps the reader to better understand the design and implementation aspects of the tool. To make the reader walk through the basic concepts of various technologies used, which would help him better understand the operation of the tool.

1.1 Cisco Switches:

The Cisco Catalyst 3550 Series Switch is a stackable, multilayer switch that provides high availability, quality of service (QoS), and security to enhance network operations. With a range of Fast Ethernet and Gigabit Ethernet configurations, the Cisco Catalyst 3550 Series is a powerful option for enterprise and metro access applications. The Catalyst 3550 switch provides 48 10/100Mbps Fast Ethernet ports and 2 GBIC-based Gigabit Ethernet ports. The 10/100Mbps Fast Ethernet ports are used to connect PCs and Gigabit ports for interconnecting switches. These switches are SNMP enabled, meaning they understand SNMP protocol. “Switch Port Mapper” uses Perl’s Net-SNMP module and its related methods to query the data from these switches[[1], [4].

ISSN 2321-2152
www.ijmece.com
Vol 7, Issue 2, 2019

1.2 Understanding SNMP:

SNMP is based on the manager/agent model consisting of a manager, an agent, a database of management information, managed objects and the network protocol. The manager provides the interface between the human network manager and the management system. The agent provides the interface between the manager and the physical device(s) being managed (see the illustration below)[1], [2], [3].

SNMP is based on the manager/agent model of network management architecture.

The manager and agent use a Management Information Base (MIB) and a relatively small set of commands to exchange information. The MIB is organized in a tree structure with individual variables, such as interface status or description, being represented as leaves on the branches. A long numeric tag or object identifier (OID) is used to distinguish each variable uniquely in the MIB and in SNMP messages. SNMP uses five basic messages (GET, GET-NEXT, GET-RESPONSE, SET, and TRAP) to communicate between the manager and the agent. The GET and GET-NEXT messages allow the manager to request information for a specific variable. The agent, upon receiving a GET or GET-NEXT message, will issue a GET-RESPONSE message to the manager with either the information requested or an error indication as to why the request cannot be processed[5], [6].

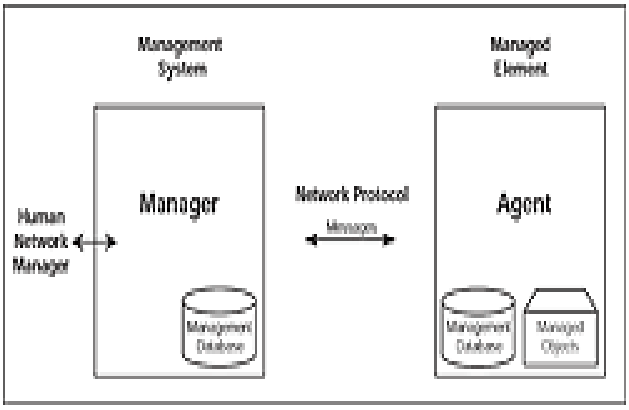

Fig. 1 SNMP protocol

1.1.1 Management Information Base (MIB):

Each SNMP element (eg. Switch) manages specific objects (eq. Switch port) with each object having specific characteristics (eg port status, description etc). Each object / characteristic has a unique object identifier (OID) consisting of numbers separated by decimal points (i.e., 1.3.6.1.2.1.2.2.1.2). These object identifiers naturally form a tree as shown below. The MIB associates each OID with a readable label (i.e., ifDescr) and various other parameters related to the object. The MIB then serves as a data dictionary or codebook that is used to assemble and interpret SNMP messages.

When an SNMP manager wants to know the value of an object / characteristic, such as the description of a switch port, the system name, or the system uptime, it will assemble a GET packet that includes the OID for each object / characteristic of interest. The element receives the request and looks up each OID in its codebook (MIB). If the OID is found (the object is managed by the element), a response packet is assembled and sent with the current value of the object / characteristic included. If the OID is not found, a special error response is sent that identifies the unmanaged object[7], [8],[9].

ISSN 2321-2152
www.ijmece.com
Vol 7, Issue 2, 2019

Apart from GET and SET commands, we have certain SNMP utilities named as 'snmpwalk' and 'snmpget' that are used more commonly to query the SNMP enabled devices for information stored in the MIB.

This research paper is organized as in section 2, discussed the related research work, in section 3, Theory calculations, in section 4, Experimental results and discussions, followed by conclusion.

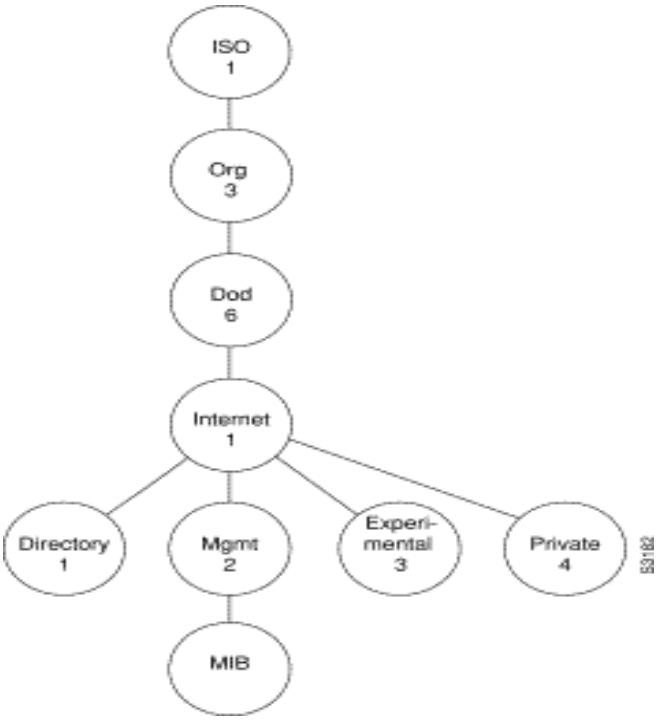

Fig.2 Internet MIB Hierarchy

2. Related Work

Perl and its Modules: Perl is an interpreted programming language known for its power and flexibility. It combines the familiar syntax of C, C++, grep, sh, and csh into a tool that is more powerful than the separate pieces used together. Among the best things with Perl is the huge number of freely available Perl modules. These modules contain pre-written Perl code that helps you complete your Perl scripts in a lot less time. A *module* provides a way to package Perl code for reuse. Available modules include support for access to Oracle and other databases; networking protocols such as HTTP (Web), POP3 (email), SNMP and FTP (file transfers); and special Win32 modules for access to the Windows 95 or NT operating systems. Many modules support object-oriented concepts. *Switch Port Mapper* uses Net-SNMP module to fetch the information from the network devices and DBI module to store the data in MySQL database server[10], [11].

The Net-SNMP module implements an object-oriented interface to the Simple Network Management Protocol. Perl applications use the module to retrieve or update information on a remote host using the SNMP protocol. The module supports SNMP version-1, SNMP version-2c (Community-Based SNMPv2), and SNMP version-3. The Net-SNMP module abstracts the intricate details of the Simple Network Management Protocol by providing a high level-programming interface to the protocol. Each Net-SNMP object provides a one-to-one mapping between a Perl object and a remote SNMP agent or manager. Once an object is created, it can be used to perform the basic protocol exchange actions defined by SNMP. This module has certain methods, which the tool uses to achieve desired results. The

following are some methods, which are used in this research paper. Figure.1 and Figure.2 represents the SNMP Protocol and Internet MIB Hierarchy.

3. Theory/Calculation

Port-to-IP mapping algorithm:

The main function of this Research Paper is to get the port to IP mapping information from the switches and the procedure is outlined below:

Choose a switch: Select the first port on the switch, i.e., FastEthernet0/1 Get the index value of the port by running SNMP query using ifDescr OID 1.3.6.1.2.1.2.2.1.2 and grep for FastEthernet0/1. The value obtained will be the OID itself with a number appended to it at the end. Separate that number from the OID and that gives the port index value.

Use the index value to get the VLAN information of the port using the OID '1.3.6.1.4.1.9.9.68.1.2.2.1.2.index'. The result will be an Integer value. This value is required to change the SNMP community string to 'public@VLAN' and will be used in the next sessions.

Using the index and VLAN values get the bridge port information using the OID 1.3.6.1.2.1.17.1.4.1.2 and grep for the value index. This value is required to get the MAC address of the device (in decimal form) connected to this switch port. The value obtained will be the OID with a number appended to it at its rear. Separate this number from the OID to get the bridge port value.

With the above value run SNMP query on the switch to get decimal MAC address using OID 1.3.6.1.2.1.17.4.3.1.2 and grep for the bridge port value. The resultant will be a dotted decimal number appended to the rear of the OID value. Separate this dotted number to get the MAC address of the device in decimal form.

1. Get the equivalent Hex value from the above decimal value using OID 1.3.6.1.2.1.17.4.3.1.1 and grep for the decimal MAC value. The output will be a MAC address in Hexa-decimal form.

2. Use this Hex value and query the Router to get the IP address of the device connected to the port using OID 1.3.6.1.2.1.4.22.1.2 and grep for the Hex MAC value. The output obtained will be the OID with dotted decimal number appended to its rear. Separate this number to get the IP address of the device. To implement these steps manually, will be a time consuming process because same steps need to be repeated for each port on the switch and if the number of switches are more…just imagine. Switch Port Mapper is designed to overcome this difficulty. It uses the Perl script that does the required mapping using the above steps and even more, which are included below. Connect to the database server and dump the data.

Repeat steps 3 to 9 by selecting the next port of the switch.

Repeat steps 3 to 10 until the port value becomes equal to 48.

Repeat steps 2 to 11 by selecting new switch in the list.

ISSN 2321-2152
www.ijmece.com
Vol 7, Issue 2, 2019

Block Diagram:

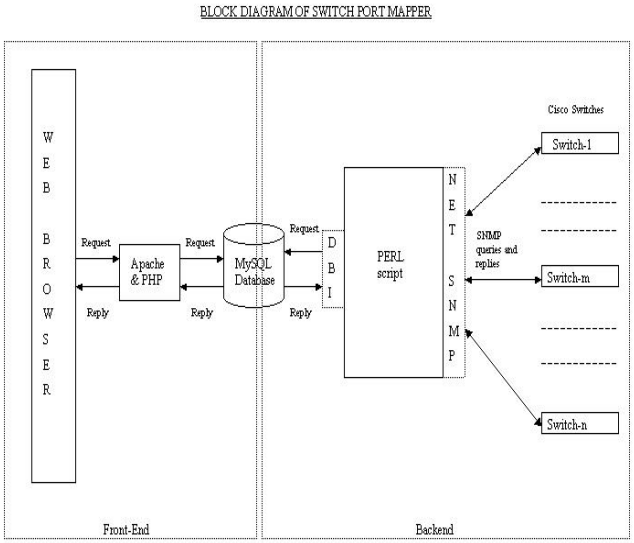

Fig.3 Block Diagram

Backend:

As shown in the diagram, a perl script queries the Switches for the required info and passes the collected data to MySQL database server for storage. The Perl script uses two of its modules named Net-SNMP to create an interface with the network switches to pass SNMP requests and DBI to connect to MySQL server database. Let's see how this is implemented in the perl script[12], [13], [14. Fig.3 represents the block diagram of the entire flow.

Front-end:

In the front-end process, the Client program sends requests to Apache web server and invokes the PHP scripts, which initiates connection to MySQL server to run SQL commands to collect data. The requested data obtained from the database server is passed back to the client in a predetermined format.

Interface program:

Once the data is in the database, we need a program that acts as an interface between the client and the server to retrieve the data as and when required. The interface should be simple and user-friendly. Switch Port Mapper has used Javascript program to build this interface. This program is a freeware downloaded from Internet. We made some changes to the code to use it in the Research Paper. Basically, we include this Javascript code in a HTML file that resides on web server's **/root/switchportmapper** directory. Whenever we connect to **http:/webservername/switchportmapper**, this file gets loaded onto web browser (used as client program)[15], [16], [17].

4. Experimental Design and Results

ISSN 2321-2152
www.ijmece.com
Vol 7, Issue 2, 2019

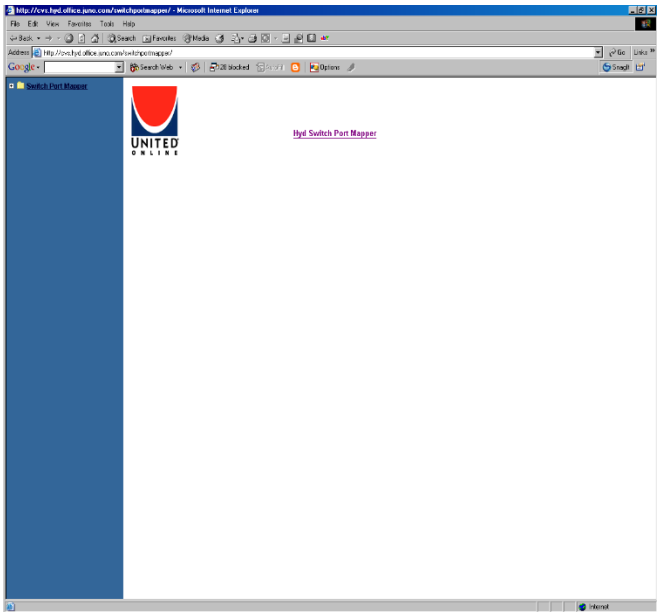

Fig.4 Main Page

This is the page that gets loaded into the browser when we connect to Switch Port Mapper main page. The frame on the left displays the folder tree view built using Javascript, which is a freeware available on Internet. Fig.4 represents the main page. On the left frame, click on the '+' sign to display the folders under the Root folder 'Switch Port Mapper'.The Root folder 'Switch Port Mapper" has the following folders:

Hyd Switches -> Individual Switch info is displayed
Search -> Provides options to display data
Help -> Documentation
Useful Links -> Links useful to our organization.

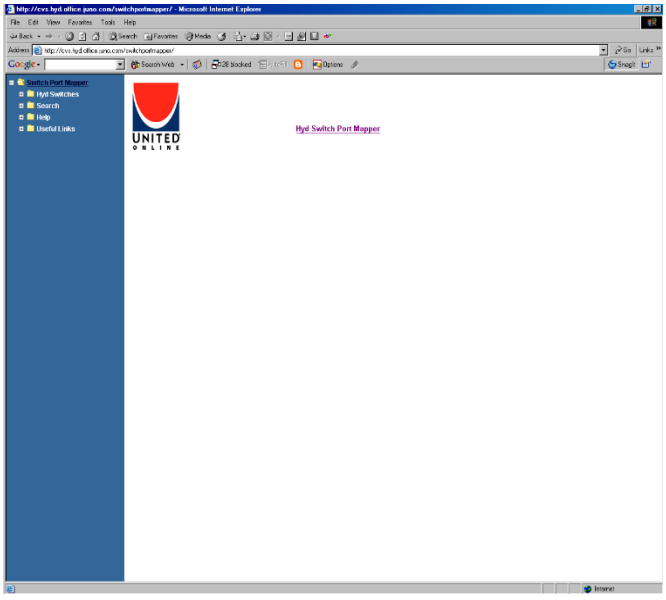

Fig. 5 Switch Port Mapper root folder

Click on the '+' sign beside these folders to display

ISSN 2321-2152
www.ijmece.com
Vol 7, Issue 2, 2019

sub-folders under them. Fi.4

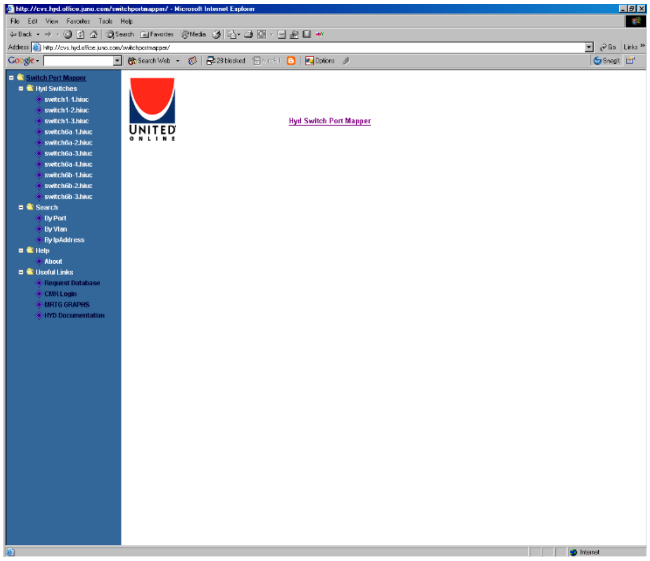

Fig.6 Hyd Switches

Under Hyd Switches, we have the sub-folders each representing a switch. Clicking on the names invokes a PHP script on apache server that connects to the database server and fetches the information specific to that switch. The screenshot below shows the output displayed when the user selects the first sub-branch under 'Hyd Switches'. Fig. 5 and Fig.6 represents the Switch Port Mapper root folder and hybrid Switches.

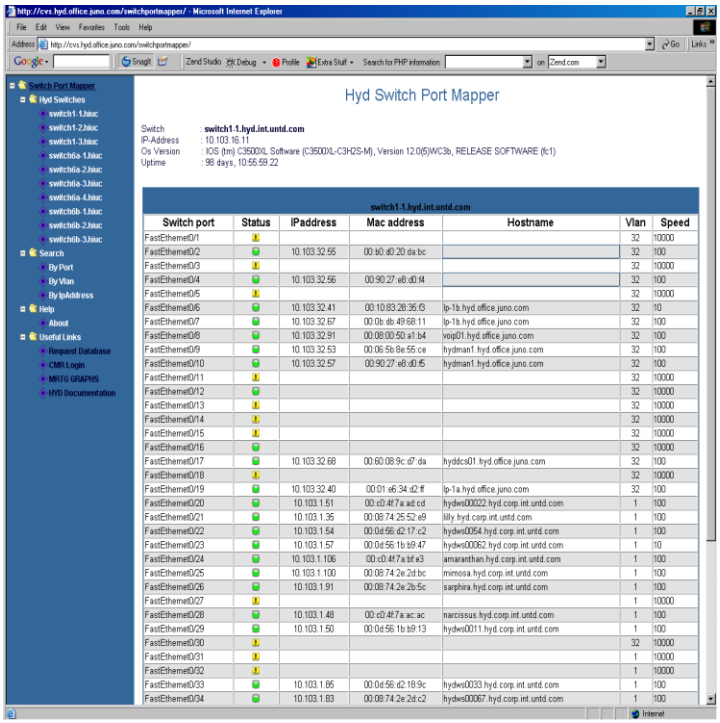

Fig.7 Switch Configuration Page

ISSN 2321-2152
www.ijmece.com
Vol 7, Issue 2, 2019

It shows that the information specific to each port, its status info, IP address of the device connected to it, its mac address, the hostname and the vlan info. This gives the complete picture of the switch and its busy ports.

Similar pages get displayed, but with some changes specific to each switch configuration, by clicking on the other switch names. So far we have seen that the tool can be used to get the complete configuration details of a switch just by clicking on its name. Sometimes it is required to search for specific info like to what switch-port a device is connected given the IP address. Focusing on such requirements we have come up with some options, which are included under Search folder. Fig.7 shows the Switch Configuration Page.

Search has the following sub-folders:

- By Port
- By VLAN
- By Ipaddress

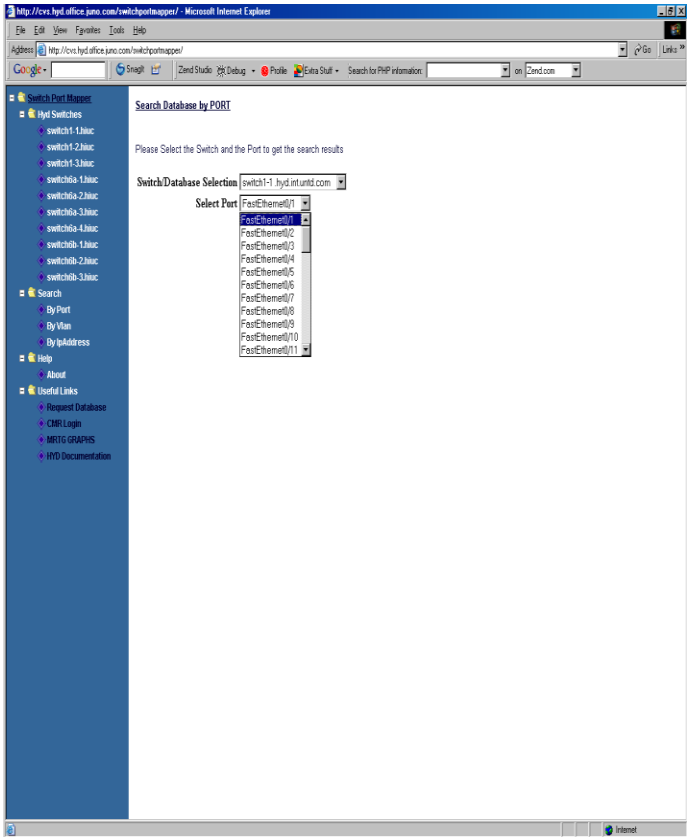

Fig.8 Search by Port Page

The screenshot displays the output when user intends to search the database based on port. The page displays two drop-down boxes, one with switch names and the other with port numbers. User selects the switch name and then any port to get the details of it. This operation is shown in the next screenshot. On selecting switch 'switch1-1.hyd.int.untd.com' and port 'FastEthernet0/1' and clicking on the button 'Get

ISSN 2321-2152
www.ijmece.com
Vol 7, Issue 2, 2019

the Host' will actually send the request to a PHP script, passing the switch name and port number as arguments. Fig.8 gives the search by port page.

Search by Port results

The Script connects to database server and queries the information and displays the data as shown above. Next option search by IP address displays the below page that prompts the user to enter the IP address of the device whose switch port information needs to be queried. Figs 9 and 10 represents the, search by port and search by IP address.

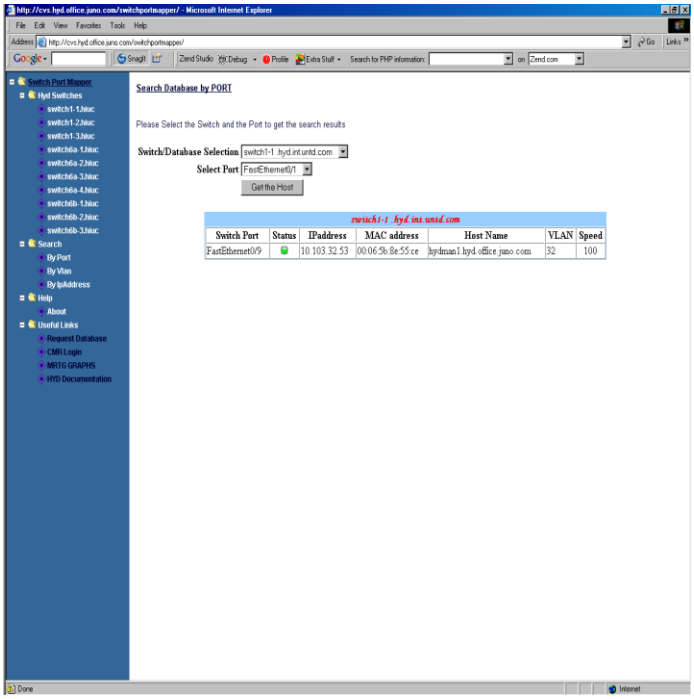

Fig.9 Search by port

ISSN 2321-2152
www.ijmece.com
Vol 7, Issue 2, 2019

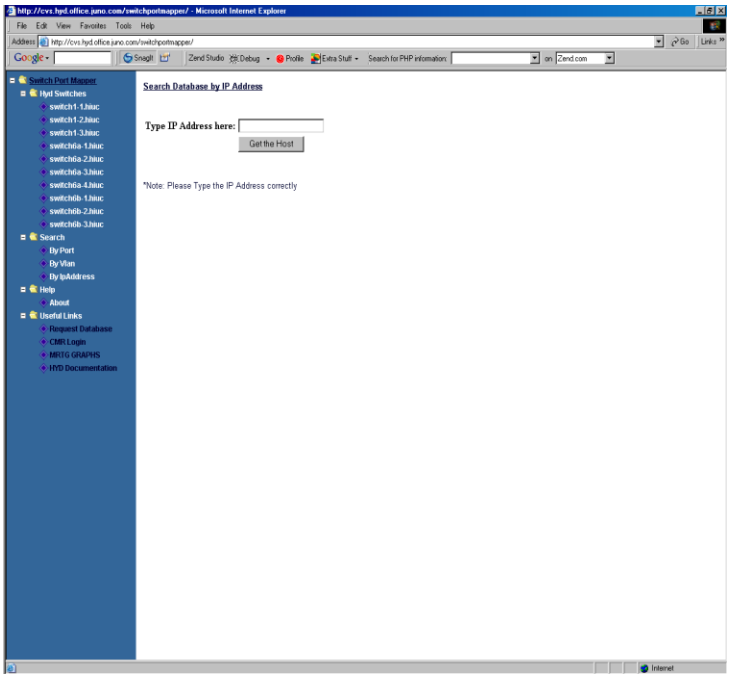

Fig.10 Search by IP address Page

User enters the IP address in the text box and clicks on the button 'Get the Host'. This action invokes the PHP script on the server and passes the IP address as the argument to the script. Using this argument PHP connects to the database server and searches the tables for switch port details and displays the information regarding the name of the switch and the port.

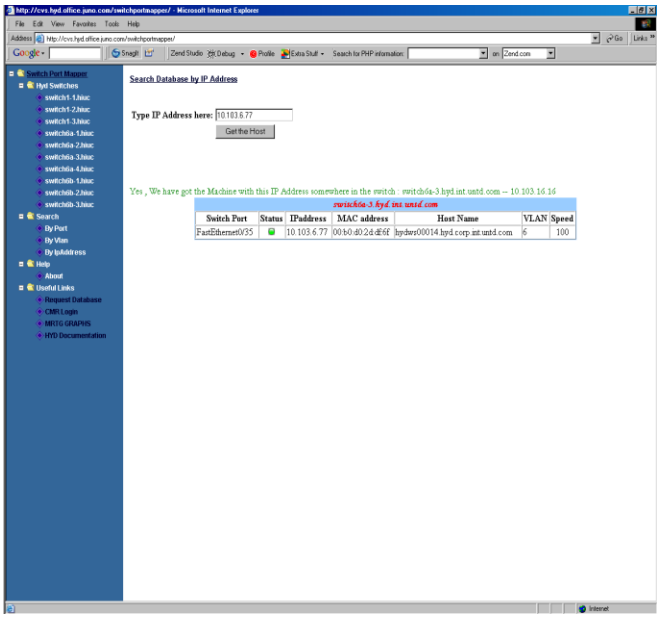

Fig.11 Search by IP results Page

The other folders are not so important and it is added them here to facilitate the users in my organization to connect to those sites directly without opening a new client program to connect to when they are using this tool [18], [19]. Fig.11 represents the search by IP results page.

5. Conclusions

The MAC/IP address information is discovered from a Router that is directly connected to the same subnet as the switch. SNMP (Simple Network Management Protocol), Perl language and HTML/JavaScript are used to develop this Research Paper. SNMP protocol collects the required Port/MAC/Interface information from the switch; Perl is used to run these SNMP queries and HTML/Javascript to display the results. Since this is a real-time discovery you can view the operational status and port speed of each port. The Output can be exported to Microsoft Excel® or mySQL database. In corporate offices where hundreds of computers are interconnected using switches, Switch Port Mapper greatly helps in tracing the computers affected by virus within seconds thereby reducing the damage to almost zero.

5. References

[1] Lara, A.; Kolasani, A.; Ramamurthy, B. Network Innovation Using OpenFlow: A Survey. IEEE Commun. Surv. Tutor. 2013, 16, 1–20.
[2] Astuto, B.N.; Mendonça, M.; Nguyen, X.N.; Obraczka, K.; Turletti, T. A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks. IEEE Commun. Surv. Tutor. 2014, doi:10.1109/SURV.2014.012214.00180.
[3] Jain, R.; Paul, S. Network Virtualization and Software Defined Networking for Cloud Computing: A Survey. IEEE Commun. Mag. 2013, 51, 24–31.
[4] Open Networking Foundation. Available online: https://www.opennetworking.org/ (accessed on 22 July 2013).
[5] Dorie,1. A.; Salim, J.H.; Haas, R.; Khosravi, H.; Wang, W.; Dong, L.; Gopal, R.; Halpern, J. Forwarding and Control Element Separation (ForCES) Protocol Specification. RFC 5810 (Proposed Standard), 2010. Available online: https://datatracker.ietf.org/doc/rfc5810/ (accessed on 22 July 2013).
[6] Yang, L.; Dantu, R.; Anderson, T.; Gopal, R. Forwarding and Control Element Separation (ForCES) Framework. RFC 3746 (Informational), 2004. Available online: https://datatracker.ietf.org/doc/rfc3746/ (accessed on 22 July 2013).
[7] 7. Hares, S. Analysis of Comparisons between OpenFlow and ForCES. Internet Draft (Informational), 2012. Available online: https://datatracker.ietf.org/doc/ draft-hares-forces-vs-openflow/ (accessed on 17 February 2014).
[8] Haleplidis, E.; Denazis, S.; Koufopavlou, O.; Halpern, J.; Salim, J.H. Software-Defined Networking: Experimenting with the Control to Forwarding Plane Interface. In Proceedings of the European Workshop on Software Defined Networks (EWSDN), Darmstadt, Germany, 25–26 October 2012; pp. 91–96.
[9] Lakshman, T.V.; Nandagopal, T.; Ramjee, R.; Sabnani, K.; Woo, T. The SoftRouter Architecture. In Proceedings of the ACM Workshop on Hot Topics in Networks (HotNets), San Diego, CA, USA, 15–16 November 2004.
[10] Zheng, H.; Zhang, X. Path Computation Element to Support Software-Defined Transport Networks Control. Internet Draft (Informational), 2014. Available online: https://datatracker.ietf.org/doc/draft-zheng-pce-for-sdn-transport/ (accessed on 2 March 2014).
[11] Rodriguez-Natal, A.; Barkai, S.; Ermagan, V.; Lewis, D.; Maino, F.; Farinacci, D. Software Defined Networking Extensions for the Locator/ID Separation Protocol. Internet Draft (Experimental), 2014. Available online: http://wiki.tools.ietf.org/id/ draft-rodrigueznatal-lisp-sdn-00.txt (accessed on 2 March 2014).
[12] Rexford, J.; Freedman, M.J.; Foster, N.; Harrison, R.; Monsanto, C.; Reitblatt, M.; Guha, A.; Katta, N.P.; Reich, J.; Schlesinger, C. Languages for Software-Defined Networks. IEEE Commun. Mag. 2013, 51, 128–134.
[13] Foster, N.; Harrison, R.; Freedman, M.J.; Monsanto, C.; Rexford, J.; Story, A.; Walker, D. Frenetic: A Network Programming Language. In Proceedings of the ACM SIGPLAN International Conference on Functional Programming, Tokyo, Japan, 19–21 September 2011.
[14] Monsanto, C.; Reich, J.; Foster, N.; Rexford, J.; Walker, D. Composing Software-Defined Networks. In Proceedings of the USENIX Syposium on Networked Systems Design & Implementation (NSDI), Lombard, IL, USA, 2–5 April 2013; pp. 1–14.
[15] Voellmy, A.; Kim, H.; Feamster, N. Procera: A Language for High-Level Reactive Network Control. In Proceedings of the ACM Workshop on Hot Topics in Software Defined Networks (HotSDN), Helsinki, Finland, 13–17 August 2012; pp. 43–48.
[16] Facca, F.M.; Salvadori, E.; Karl, H.; Lopez, D.R.; Gutierrez, P.A.A.; Kostic, D.; Riggio, R. NetIDE: First Steps towards an Integrated Development Environment for Portable Network Apps. In Proceedings of the European Workshop on Software Defined Networks (EWSDN), Berlin, Germany, 10–11 October 2013; pp. 105–110.
[17] Tennenhouse, D.L.; Wetherall, D.J. Towards an Active Network Architecture. ACM SIGCOMM Comput. Commun. Rev. 1996, 26, 5–18.
[18] Campbell, A.T.; De Meer, H.G.; Kounavis, M.E.; Miki, K.; Vicente, J.B.; Villela, D. A Survey of Programmable Networks. ACM SIGCOMM Comput. Commun. Rev. 1999, 29, 7–23.
[19] Feamster, N.; Rexford, J.; Zegura, E. The Road to SDN: An Intellectual History of Programmable Networks. ACM Queue 2013, 12, 20–40.