Constella: A Complete IP Network Topology Discovery Solution

Fawad Nazir[1,2,4], Tallat Hussain Tarar[3,4], Faran Javed[4], Hiroki Suguri[5], Hafiz Farooq Ahmad[4,5], and Arshad Ali[4]

[1] National ICT Australia (NICTA), Australia
[2] University of New South Wales (UNSW), Australia
fawad.nazir@nicta.com.au
[3] European Oeganization For Nuclear Research (CERN), Switzerland
tallat.hussain@cern.ch
[4] NUST Institute of Information Technology (NIIT), Pakistan
{43faran,arshad.ali}@niit.edu.pk
[5] Communication Technologies, Japan
{farooq,suguri}@comtec.co.jp

Abstract. Network topology discovery for the large IP networks is a very well studied area of research. Most of the previous work focus on improving the efficiency in terms of time and completeness of network topology discovery algorithms and less attention has been given to the deployment scenarios and user centric view of network topology discovery. In this paper we propose a novel network topology discovery algorithm and a flexible architecture. The silent features of our work are *loosely coupled architecture, network boundary aware architecture, discovering the transparency of dumb/incorporative elements, flexible network Visualization, and intelligent algorithm for quick response to user discovery request*. To the best of our knowledge no existing solution has focused on the above mentioned requirements. After several years of research experience in developing a complete, flexible and scalable solution for network topology discovery we propose to divide it into three loosely coupled components: topology discovery algorithm, topology object generation and persistence, and topology visualization. In this paper we will present our proposed integrated complete network topology discovery solution, discuss the motivation of our proposed architecture, the efficiency and user-friendliness of our work. Our results show that the average accuracy of our algorithm is 92.4% and takes one second to discover 100 network elements.

Keywords: Network Monitoring and Management, Simple Network Management Protocol (SNMP).

1 Introduction

Network topology is a representation of the interconnection between directly connected peers in a network. A physical topology corresponds to many logical topologies each at a different level of abstraction. At the IP level, peers are hosts or routers one IP hop away from each other and at the workgroup level the peers are

S. Ata and C.S. Hong (Eds.): APNOMS 2007, LNCS 4773, pp. 425–436, 2007.

hosts connected by a logical link. Network topology constantly changes as nodes and links join a network, personnel move offices, and network capacity is increased to deal with added traffic. Keeping track of network topology manually is a frustrating and often impossible job. Network topology knowledge including the path between endpoints, can play an important role in analyzing, engineering, and visualizing networks. Most of the previous work [7][9][10][11][13][14] focus on improving the efficiency in terms of time and completeness of network topology discovery algorithms and less attention has been given to the deployment scenarios and user centric view of network topology discovery. The goal of our work is to automatically discover network topology and visualize it. We have been doing research on network topology discovery for the last few years now [1][2][3][5][6] and this paper is summary of our earlier work in order to propose an integrated complete IP network topology solution. In addition to that in the next section we explain the novel objectives of our work.

2 Contribution of This Paper

The motivation of this paper comes from the problem of discovering network topology at European Oeganization for Nuclear Research (CERN), Switzerland. At CERN there are thousands of machine, hundereds of routers/L3 switches/L2 switches, hundereds of network printers and huge clusters of computer for data analysis. Each department at CERN has its own network management team. These teams differ in the type of network management and monitoring polocies/requirements. Therefore there is a need for an autonomous network topology discovery solutions that is able to support multiple clients at the same time, flexible enough to discover the network topology given a network segments as a referecne start and management solutions and support different views for different administrator clients. Currently, to the best of our knowledge there is no solutions thats supports these type of features. With these challenges in mind, we introduce *Constella* that encompasses the following advantages:

Loosely coupled architecture: Loosely coupled architecture is an innovative, challenging and an important concept in order to discover the network topology and share it with large numbers of clients. We address this problem through decoupling the topology discovery and topology visualization algorithms by adding an additional layer “topology object generation and persistence” in between. More detail about decoupling is discussed in our previous work[5].

Boundary aware architecture: Boundary awareness is an important feature of network topology discovery for distributed management and probing restriction. To facilitate distributed network topology discovery, we have developed a boundary aware network topology discovery algorithm using which we can discover networks within a predefined management scope/domain.

Transparency of Dumb/incorporative elements: Networks contain Hubs that do not participate in switching protocols and, thus, are essentially transparent to switches and bridges in the network. Similarly, the network may contain switches from which no address-forwarding information can be obtained either because they do not speak

SNMP or because SNMP access to the switch is disabled. Clearly, inferring the physical interconnections of hubs and "uncooperative" switches based on the limited AFT information obtained from other elements poses a non-trivial algorithmic challenge.

Efficient User friendly Visualization of Large Networks: Visualization of network topology after the complete network discovery is what most of the software's support and in most of the current solutions network visualization is tightly coupled with the discovery algorithm. In our solutions the network discovery is what is done at the server side (could be one or multiple) and the network visualization is completely separate from network discovery and this is done on the client side (at CERN we need more than 100 clients at one time). The communication between discovery machine and visualization machine is done by the topology object.

Intelligent Algorithm for Quick Response to Discovery Request: Sending ICMP Echo request to all the possible IP addresses is not feasible to determine the availability of in large networks. Therefore there is a need to devise an intelligent algorithm that generated a list of IP addresses having a high probability of being assigned to devices in the network. We devised an algorithm that utilizes the entries in the ARP cache of the router to generate the list of IP addresses to be tested [3].

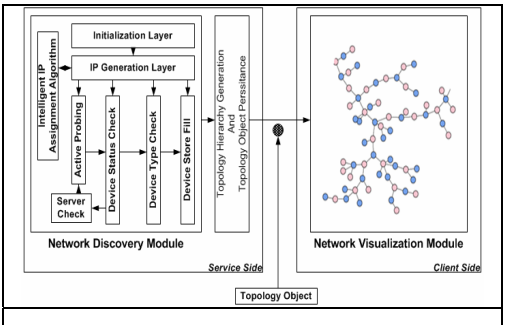


Fig. 1. Revised Constella Architecture

3 Constella: IP Network Topology Solution

This section will discuss two different proposed architectures of Constella. This section will discuss the functionnality and architectural importance of three main modules of our topology discovery algorithm i.e. topology discovery algorithm, topology object generation and persistence, and topology visualization Furthermore, two revisions of this architecture are discussed.

3.1 Constella System Architecture

The basic Constella Architecture is discussed in details in[5]. Here we discuss in detail the revised architecture which has minor changes from the our previously proposed architecture [5]. The major revision in this architecture is that in the first architecture we need SNMP Agent to be installed on the machine which is running Constella but in the second architecture we don't need SNMP agent on the machine running Constella. This makes Constella more deployable and mobile. In (Fig-1) this architecture we get rid of the ARP-MAC mapping layer which was used to get the ARP information from the local host using SNMP requests. Furthermore we introduce another layer "Server Check" layer. This layer is responsible of identifying DNS,

Mail etc servers in the network and get useful topology or active host information from them and feed it in to the Active probing layer.

3.2 Constella Algorithms

In this section we will discuss all the individual algorithms and their relationship with each other in order to discover the network topology in a collaborative way.

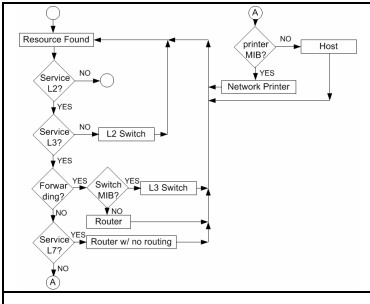


Fig. 2. Device Discovery Algorithm Flowchart

3.2.1 Device Discovery Algorithms

In this paper (Fig 2) we discuss the discovery of mainly five type of devices. These devices are network router, network printer, L2/L3 switches, computers/servers/host and network HUBs / unmanaged switches. The two main protocols used for device discovery are Internet Control Message Protocol (ICMP) Echo request/reply and Simple Network Management Protocol (SNMP). ICMP is used to discover the device availability/connectivity and SNMP is used to identify the type of device and its interconnections.

A. Host Discovery

First of all we discover the devices with the help of ICMP Echo Request/reply. After that we send SNMP request to all the discovered devices. The devices that reply to the SNMP request will be considered for further test. The devices which will not reply will be considered to be Network Hosts, as we take an assumption that all the managed devices (routers, switches, printers etc) in the network have SNMP deamon running. However, there could be another possibility here that a device that is a network host is also an SNMP daemon. So in this case we have to check whether it's a host or some other device. If a device is a network host then it should work on Layer 7.

Now we will discuss the tests which will be performed to actually see whether a device is a network host or not.

```
bash-2.05b$ snmpwalk -c public 10.10.21.8  .1.3.6.1.2.1.1.7.0
SNMPv2-MIB::sysServices.0 = INTEGER: 76
```

In the above command we have mentioned an OID .1.3.6.1.2.1.1.7.0 which is used to check the service provided by the specific host. To check a device whether its providing services on Layer 7 and 4 should have a value of sysServices as 72.

$$L4 \ \& \ L7 \ services: (2 \wedge (4 - 1) + 2 \wedge (7 - 1) = 72$$

This means if a network devices has services value greater than $2 \wedge (7 - 1) = 64$ than it's a network host.

B. Switch Discovery

The switch discovery contains discovery of Layer 2, Layer3 and un managed switches. In the following section we will discuss about the discovery of all these switch types.

a. Managed L2 Switch Discovery

The layer 2 switches work on Layer 2 and Layer 1. The figure bellow shows an snmpwalk commands to retrieve sysServices. As shown in the output of this command the sysService value for a L2 switch (10.10.6.99) is 3. That means that the L2 switch works on Layer 1 and Layer 2.

```
bash-2.05b$ snmpwalk -c public 10.10.6.99  .1.3.6.1.2.1.1.7.0
SNMPv2-MIB::sysServices.0 = INTEGER: 3
```

This can also be proven mathematically as:

$$L1 \ \& \ L2 \ services: \ (2 \ ^ \ (1 - 1) \ + \ 2 \ ^ \ (2 - 1) = 3$$

In this equation we have shown that *L1 mean* $2 \ ^ \ (1 - 1) = 1$ and *Layer 2 means* $2 \ ^ \ (2 - 1)$. Another way to identify the L2 switch is to convert the 3 into binary and upto 7 digits and then check which bits are on. The layer 2 switch will be working on those layers. The sysServices value for L2 switch is 3 and if we convert this to binary it makes 1100000. So the binary representation shows that the L2 switch works on Layer 1 and Layer 2. In this we can identify the L2 switches.

1	1	0	0	0	0	0
Layer 1	Layer 2	Layer 3	Layer 4	Layer 5	Layer 6	Layer 7

b. Managed L3 Switch Discovery

Layer 3 switches as the name states works on Layer 3 and as it's a switch so it also work on Layer 2 and Layer 1. The layer 3 switches have the capability of working on the IP Layer and switching. It can also be discovered by checking its sysServices parameters.

$$L1 \ \& \ L2 \ \& \ L3 \ services: \ (2 \ ^ \ (1 - 1) \ + \ 2 \ ^ \ (2 - 1) + 2 \ ^ \ (3 - 1) = 7$$

Now looking at the binary conversion method. Converting 7 to binary will be 1110000. This clearly shows that this device is a L3 switch and it works on Layer 1, Layer2 and Layer 3.

1	1	1	0	0	0	0
Layer 1	Layer 2	Layer 3	Layer 4	Layer 5	Layer 6	Layer 7

c. Un-Managed Switch Discovery

Besides SNMP-enabled bridges and switches that are able to provide access to their AFT, a switched network can also deploy "dumb" elements like hubs to interconnect switches with other switches or hosts. Hubs do not participate in switching protocols and, thus, are essentially transparent to switches and bridges in the network. Similarly, the network may contain switches from which no address-forwarding information can be obtained either because they do not speak SNMP or because SNMP access to the switch is disabled. This was also a challenge to discover the unmanaged switches or hubs. The algorithm which we have developed can not find

out the hierarchy of the unmanaged switches or hubs but at least it can identify that the link which is attached to unmanaged device or devices. It can also identify that with a particular port of certain device an unmanaged switch/switches are connected.

Now we will talk about identifying an unmanaged switch which is connected to some managed switch. One Ethernet card has one MAC address and if one network card is attached to one port of the switch then in the Address Forwarding table of the switch only one MAC will be mapped with one Port. Similarly if we have a switch connected to certain port of another managed switch then you can see in the Address forwarding table (AFT) it will display multiple MAC for a single port. So if we see some multiple MAC for a single port this means that there is some switch connected to that port.

We have successfully discovered that switch is connected to certain port of the managed switch. Now we have to discover that this switch is a managed or an unmanaged switch. To discover this the MAC address of the managed discovered switch is to be compared with the all MAC address connected to a particular switch port. If the MAC address is there then this means the switch connected is a managed switch other wise if we don't find any MAC this mean it's a unmanaged switch.

C. Router Discovery

In this section we will discuss how to discover the routers in the network. In the case of the router we have to check two things. First of all we will check that whether forwarding is enabled or not. This can be checked by using SNMP IP-MIB. We check the ipForwarding parameter of IP-MIB. If the ipForwarding is one then this means it a router and if its 0 then this means it a switch.

```
bash-2.05b$ snmpwalk -c public 10.10.22.3  .1.3.6.1.2.1.4.1
SNMPv2-MIB::sysServices.0 = INTEGER: 3
```

The OID for ipForwarding is .1.3.6.1.2.1.4.1.0. Now we have discovered that the IP is a router or not. Now we also have to check the services it provides to check whether it's a soft router or a proper hardware router. For this we have to check the services it provides with the help of sysServices parameter.

$$L1 \ \& \ L2 \ \& \ L3 : 2 \wedge (1\text{-}1) + 2 \wedge (2\text{-}1) + 2 \wedge (3\text{-}1) = 7$$

The equation above shows that's it's a router and its works on Layer 1, Layer 2 and Layer 3. Now we have to check one more thing this machine can be a Layer 3 switch. As the layer 3 switches also works on Layer 3. Now to check whether is a Layer 3 switch to a router we have to do one more check. This check is for switching services. To check that a device is providing switching services or not we have to check for the Bridge-MIB. To check that a devices has implemented bridge MIB or not. We will send a request to a device for OID : .1.3.6.1.2.1.17.1 . If the machine repyles then that means it a L3 switch otherwise it's a hardware router.

3.2.2 Single Subnet Discovery Algorithm

Single subnet means devices that are under one interface of a router. The single subnet discovery is different from multiple subnet discovery, as in the case of multi subnet discovery router posses many limitations to the protocols. The MAC level

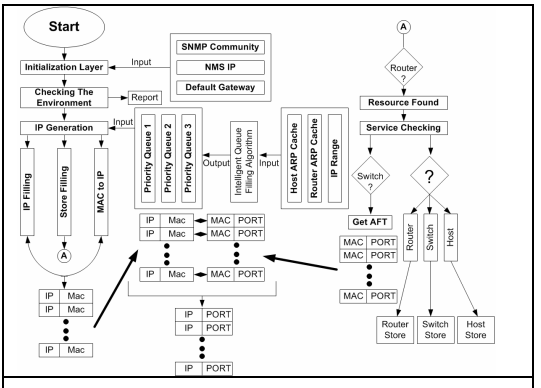


Fig 3. Single Subnet Topology Discovery Algorithm Flowchart

broadcast are blocked at the router so ARP information from multi-subnets cannot be retrived. On the other hand single subnet can utilize RARP and ARP information for topology discovery. In multi-subnet case we can not use the ICMP echo request/ reply broadcast message as the devices that are beyond the first router will not reply because router block IP level broadcasts. Due to these reasons we choose to separate algorithms for single subnet and multiple subnets. In this section we will limit our discussion to the discovery of a single subnet and then in the next sections we will extend our discussion to the discovery of multiple subnets. In the single subnet discovery we will discuss discovery of hosts, stub router, managed switches & unmanaged switches that are all inside a single subnet. Our single subnet discovery algorithm only requires one input i.e. the SNMP community string. This input is given at the initialization layer (Fig-3). Other inputs to this layer are extracted automatically by extracting information from the local operating system like NMS (Network Management Station) IP address and the gateway address of the NMS. The community string is used by SNMP to query the devices in the network. The NMS IP is used to get the range of the IP in that network. The default gateway information is used to get the ARP cache of the router in order to guess the active IP ranges in the network. Furthermore, we also get the next hop information form the router.

The next layer is the "checking the network environment" layer. In this layer we will check the network and the NMS environment that whether it's feasible for topology discovery or not. In this layer we check the following things, IP address is assigned to the NMS, ping and ARP commands are available in the NMS, gateway is assigned to the NMS, is the gateway device up, is SNMP enabled on the gateway router or not etc. This layer checks all these questions if some thing is missing then it gives report to the user otherwise it continues toward topology discovery interface. The next layer is the IP generation layer. This layer generates the IP's in different priority queues. The priority is given to IP's that are most likely to be assigned and discovered in the network. The algorithm used in this layer is discussed in detail in a our previous work[3]. This layer use information from multiple sources like ARP cache of the gateway router, ARP cache from the NMS and the IP range of the NMS. After this information is gathered by the IP generation layer this layer initiates three more layers IP Filling, Store Filling and MAC to IP Layer. These three layers are three different threads that will work in parallel. The IP Filling layer is responsible to check for alive computers and then store them into a data structure. The MAC to IP layer builds a MAC to IP mapping of each of the resource discovered. The third layer that is the store filling layer is responsible of identifying the type of resource and

inserting it into the particular store. During store filling if we encounter a managed switch our algorithm will get the port to Mac mapping and then compare the port to Mac mapping with the IP to MAC mapping we already have to get the meaningful IP to Port mapping. This process in explained in the (Fig-3). In this way we get the information about the switch port to device connectivity. Similarly in this way we get the connection of router with switch. All these device to device connections are discussed in detail in our previous work[5].

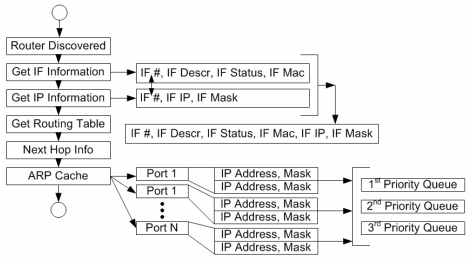


Fig. 4. Multi-Subnet Topology Discovery Algorithm Flowchart

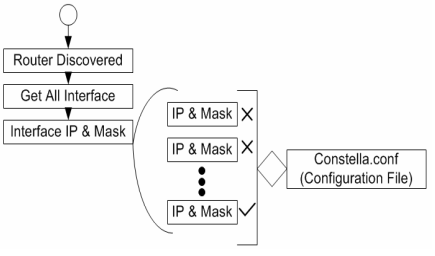


Fig. 5. Boundary Aware Topology Discovery Algorithm Flowchart

3.2.3 Multi-subnet Discovery Algorithms

In multiple subnet topology discovery we have to discover subnets beyond a single router interface. In this case the most useful information is the routing information and *ARP* information from the routers. Now lets look (Fig 4) at how we can utilize this information from the routers to discover multiple subnets. First of all we will discuss how to discover all the subnets that are attached to a particular router. To start our algorithm we get the first router from the router store. Then get the *IF* (interface) information from the *IF-MIB*. The *IF-MIB* will contain the information about the Interface number which is a sequence number of all the interfaces in the router, the interface description that explains what type of interface it is, the interface status which tell about the interface status that whether its up or down and the interface MAC address. Now for mapping the interface to an IP-address and subnet mask we have to get the information from the *IP-MIB*. From the IP-MIB we will get information about the Interface number, the associated IP address to that interface and the subnet mask. The interface number will be used as an index between IF-MIB and IP-MIB. Therefore after mapping the IP-MIB to the IF-MIB using IF number we can get some useful information like Interface number, interface description, interface status, interface MAC address, interface IP address and the interface MASK of a particular interface. In this way you will get the IP address of all the interfaces and their subnet masks. In order to get the next Hop information we will utilize the default gateway information in the routing table of that router. Now we will discuss about the process of topology discovery of each subnet that is attached to a router. As discussed earlier that the input to our topology discovery for single subnet is decided by the IP generation layer. Originally for single subnet the IP generation layer get the input as ARP cache from the gateway router, ARP cache from the NMS and IP and Mask from the NMS. Now in the case of multiple subnets this information will be a bit

different. In this case we will be getting information from the router about the specific ARP cache of each interface. In addition to that we will also have the IP and subnet mask of that interface. All this information about an interface will become an input to our single subnet topology discovery algorithm. Now similarly, iteratively we will be able to discover the topology of the whole network.

3.2.4 Boundary Aware Topology Discovery

In general, a network is owned and administrated by an enterprise, organization, or college. The network within an administrative domain is usually also the scope of network management softwares. Our Complete solution for boundary aware network topology discovery is discussed in[1]

3.2.5 Device to Device Connections

These are given in detail in out earlier work[5].

3.2.6 Alive Device Detection Algorithm

Sending ICMP Echo request to all the possible IP addresses in a network is not feasible to determine the availability of large networks. Therefore there is a need to devise an intelligent algorithm that generated a list of IP addresses having a high probability of being assigned to devices in the network. We devised an algorithm that utilizes the entries in the ARP cache of the router to generate this list of IP addresses. First of all we retrieve the IP addresses in the ARP cache of the router using SNMP e.g.

$IP_{\blacksquare}$ = {10.10.5.2, 10.10.0.1, 10.10.6.99, 10.10.5.1, 10.10.5.3, 10.10.100.1}

We then remove the last octet.s.

$IP_{\blacksquare\blacksquare}$ = {10.10.5., 10.10.0., 10.10.6., 10.10.5., 10.10.5., 10.10.100.}

Then remove duplicates.

$IP_{\blacksquare\blacksquare}$ = {10.10.5., 10.10.0., 10.10.6., 10.10.100.}

For all the above elements in the set now we generates 254 IP addresses by appending values from 1 to 254 in the fourth octet resulting in a list of High Priority IP addresses.

*The number of High Priority IP addresses = N * 254 (where N is the distinct first 3 octets in the routers ARP cache)*

$IP_{\blacksquare\blacksquare}$ = {(10.10.5.1->10.10.5.254), (10.10.0.1 -> 10.10.0.254), (10.10.6.1->10.10.6.254), (10.10.11.1->10.10.11.254)}.

For every entry generated as a result of S*tep 3* it will create more entries, One entry each by subtracting 1, subtracting 2, adding 1 and adding 2 to the third octet (if it results in a valid entry). An entry is not entered if it is already present in the list or if this address range is already in the High Priority IP address list.

$IP_{\blacksquare\blacksquare}$ = {10.10.4, 10.10.3, 10.10.7, 10.10.1, 10.10.2, 10.10.8, 10.10.99, 10.10.98, 10.10.101, 10.10.102}.

For every entry, created as a result of *step 5,* it generates , 254 IP addresses by appending values from 1 to 254 in the fourth octet resulting in a list of Low Priority IP addresses

IP_{lp} = {(10.10.4.1 ->10.10.4.254), (10.10.3.1->10.10.3.254), (10.10.7.1-> 10.10.7.254), (10.10.1.1 ->10.10.1.254), (10.10.2.1->10.10.2.254), (10.10.8.1->10.10.8.254), (10.10.99.1->10.10.99.254), (10.10.98.1->10.10.98.254), (10.10.101.1->10.10.101.254), (10.10.102.1->10.10.102.254)}

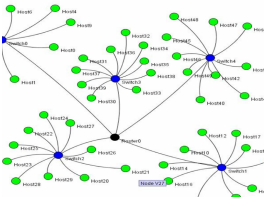


Fig. v1. Relaxed View of Topology Discovery

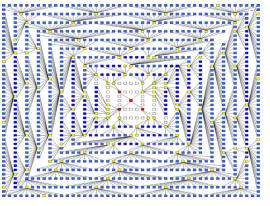


Fig. v2. Our Previous Work (SVG 2004)[6]

4 Network Visualization

In this section due to lack of space we will give a brief overview of our network visualization approach and present some snapshots (Fig v1-v2) from our implemented system. As our system is implemented in Java, therefore we use a java based visualization tool i.e. Java Universal Network and Graph (JUNG) Studio for network visualization. Moreover, we have also proposed a network visualization technique using Scalar Vector Graphics in our SVG 2004 paper[6]. The visualization using JUNG takes network topology information as input in a file and plots the nodes in the best possible manner. It also has a facility to locate a node or path between two nodes.

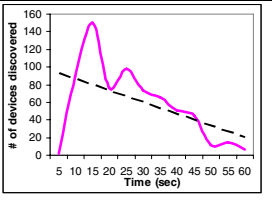
5 Performance Evaluation and Results

In this section we will discuss the performance evaluation and testing that we performed in order to validate the functionality and performance of our proposed topology discovery algorithm. Constella has been tested as a whole (Black Box Testing). Results have been drawn from the testing of each module. We have tested Constella in five different networks that have separate network configuration and devices. The algorithm was tested in Comtec Japan, CERN, Switzerland, KHU Korea, NUST research lab and NIIT, Pakistan. The accuracy of the algorithm is measured by the following formula:

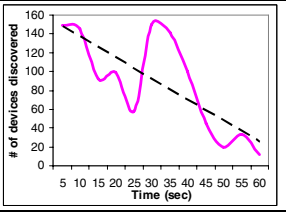
$$Accuracy(\%) = \sum_{i=0}^{n} X(i)/N$$

Here n=the total number of subnets discovered and N is the total number of nodes in the network. X(i), is a single instance of a network subnet.

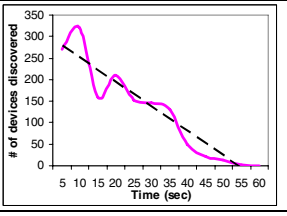
Table 1. First Five Graphs show the trends in nodes discovery and tables shows the algorithm accuracy



NIIT, Pakistan



KyongHee University, Korea



CERN, Switzerland

Site Name	*Algorithm Accuracy*
COMTEC	**100%**
CERN	**~82%**
Korea	**~95%**
NIIT Network	**~85%**
NIIT Research Lab	**100%**

In the above graphs dotted lines shows the node discovery trend-line and stable lines shows the actual nodes discovered.

Now a days, we have very large networks to be managed. To start discovering networks from first IP of the networks to the last IP would make it very slow and irritating for the network administrators. Currently all the software that discover the topology takes a lot of time to give initial response to the network administrator. The purpose of our approach was to discover the network in a way that large percentage of the devices in the network are displayed first and then rest of the network is discovered accordingly. The results in Table-1, show that our algorithms discovers the large percentage of the devices in the network initially. We tested the Constella algorithm for one hour in NUST Institute of Information Technology (NIIT) working network. The following are the results that we got from the log file generated. These results clearly show some very interesting trends (Fig-A). Every bar in the graph has two parts. The bottom part show the number of nodes discovered, the above part shows the nodes not discovered and the whole bar shows the total number of nodes checked/tested. This clearly shows that number of nodes discovered is directly proportional to number of nodes tested).

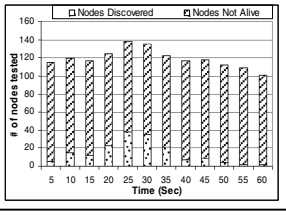


Fig-A. Relationship between nodes discovered, nodes not alive and nodes tested at NIIT.

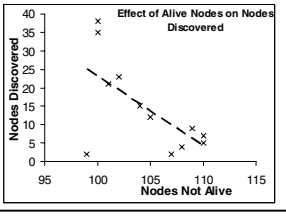


Fig-B. Shows the relation between nodes discovered and nodes alive for NIIT.

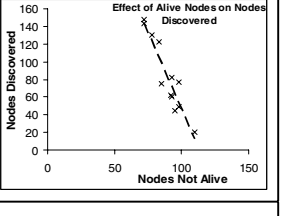


Fig-C. Shows the relation between nodes discovered and nodes alive for KHU.

The following are the conclusion from the results in Fig A,B and C.

$n \propto 1/\sim A$. *This equation shows, numbers of nodes discovered is inversely proportional to the number of nodes not alive(A). That mean if number of nodes not alive is less then number of nodes discovered will be more. (Fig A,B,C,D)*
$n \propto T$ *This equation shows, number of nodes discovered is directly proportional to number of nodes tested (T). (Fig A,B,C,D).* $\sim n \propto 1/T$ *. This equation shows, number of not nodes discovered is inversely proportional to number of nodes tested. (Fig A,B,C,D) (T).*

6 Conclusion

Despite the importance of network topology discovery, earlier research and commercial network management tools have typically concentrated on either the time efficiency or completeness property of the topology discovery algorithm. Not much attention is given to the user-centric topology discovery for multi-client dynamic networks. In this paper, we explain a detailed algorithm for discovering the topology map of IP Network using ICMP echo request/reply and SNMP-MIB and also the supporting algorithms which could help to improve the efficiency of our algorithm. We have implemented our algorithm in java and have been experimentally tested over different production networks. The results clearly validate our methodology, demonstrating the accuracy and practicality of the proposed algorithms and architecture. The average accuracy of our algorithm is 92.4%.

References

1. Ali, A., et al.: Restriction of Network Topology Discovery within a Single Administrative Domain. In: CIC'05, USA, June27-30, pp. 222–225 (2005)
2. Nazir, F., et al.: Standardizing IP Network Topology Discovery through MIB Development. In: CIC'05, USA, June 27-30, pp. 226–230 (2005)
3. Najeeb, Z., Nazir, F., et al.: An Intelligent Self-Learning Algorithm for IP Network Topology Discovery. In: LANMAN 2005, Greece (September 2005)
4. Javed, F., et al.: Loosely Coupled Architecture for Integrating Network Topology Discovery Algorithms. In: HONET 2005, Pakistan (2005)
5. Nazir, F., Jameel, M., et al.: Efficient Approach Towards IP Network Topology Discovery for Large Multi-subnet Networks. In: ISCC 2006, Sardinia, Italy (2006)
6. Ali, A., et al.: Single Snapshot Exploratory Approach for Visualizing Very Large Network Topologies. In: SVG Tokyo, Japan, September 7-10, 2004 (2004)
7. Bierman, Jones, K.: Physical Topology MIB. Internet RFC-2922 (September 2000), available from http://www.ietf.org/rfc/
8. Case, J., et al.: A Simple Network Management Protocol (SNMP). Internet RFC-1157 (May 1990), available from http://www.ietf.org/rfc/
9. Lowekamp, D.R., O'Hallaron,: Topology Discovery for Large Ethernet Networks. In: ACM SIGCOMM, San Diego, California (August 2001)
10. Breitbart, Y., et al.: Topology Discovery in Heterogeneous IP Networks. In: Proceedings of IEEE INFOCOM 2000, Tel Aviv, Israel (March 2000)
11. Faloutsos, M., et al.: On power-law relationships of the Internet topology. In: Conference on Applications, Technologies, Architectures and Protocols for Computer Communications, pp. 251–262. ACM Press, New York (1999)
12. Claffy, K.C., McRobb, D.: Measurement and Visualization of Internet Connectivity and Performance. http://www.caida.org/Tools/Skitter/
13. Stott, T.: Snmp-based layer-3 path discovery. Tech. Rep. ALR-2002-005, Avaya Labs Research, Avaya Inc. Basking Ridge, NJ (2002)
14. Govindan, R., et al.: Heuristics for Internet map discovery. In: INFOCOM 2000, IEEE, Los Alamitos, CA (2000)