PROJECT REPORT

NETWORK MAPPER

Guided by
Mr. Rajesh Dhakad

Co-guided by
Ms. Neha Mehra

Submitted by
Sarvagya Soni
Shivam Joshi
Sourabh Shrivastava
Shubham Kumawat
Yashil Jijhotiya

Computer Engineering Department
Shri G.S. Institute of Technology and Science
Indore (M.P.)
2015-16

Shri G.S. Institute of Technology and Science Indore (M.P.)

RECOMMENDATION

This is to certify that the project report entitled **"NETWORK MAPPER"** submitted by Sarvagya Soni, Shivam Joshi, Sourabh Shrivastava, Shubham Kumawat, Yashil Jijhotiya, students of 4th year B.E. (Computer Engineering) in the year 2015-2016 of this Institute, is a satisfactory account of their Project Phase-I work based on syllabus and the same is recommended for the Phase-I examination.

Mr. Rajesh Dhakad	Prof. D.A. Mehta
Guide	**Head**
Computer Engg. Department	Computer Engg. Department
S.G.S.I.T.S. Indore	S.G.S.I.T.S. Indore

Dean Academic
S.G.S.I.T.S.
Indore

Shri G.S. Institute of Technology and Science Indore (M.P.)

CERTIFICATE

This is to certify that the project report entitled " NETWORK MAPPER" submitted by , students of final Sarvagya Soni, Shivam Joshi, Sourabh Shrivastava, Shubham Kumawat, Yashil Jijhotiya year B.E. (Computer Engineering) in the year 2015-2016 of this Institute, is a satisfactory account of their Project Phase-I work based on syllabus.

Internal Examiner **External Examiner**

Abstract

The purpose of our project, is to create a network mapping tool that would map the entire LAN infrastructure using active probing technique. The software will generate a network tree dynamically, which will help network administrators to visualize the whole network. The network administrator can add or delete node to create the network infrastructure and then the software will map the network to create a visual.

Some of the needs of network administrator are, to visualize the network topology, to determine the broken links in network, to maintain a log of uplinks and downlinks along with their specifications like timing, associated nodes etc.

The project is based on active probing technique that uses probes to detect the state of a link that may be broken or active. The active links will be denoted by green lines and broken links by red lines. Also the network admin can add/delete a node by providing some information. Also when a link goes down, a notification will be generated for the network admin regarding that link with its specification. The project can be proven beneficial for the network administrators, in large organizations where it is an uphill task to trace the presence of downlinks.

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 Preamble

The aim of this project report is to describe the network mapping tool, for surveillance and monitoring of network infrastructure using active probing technique.

This project proposal is addressed to the network monitoring system and the people associated with it. It is intended to enhance the current network monitoring System, which has umpteen flaws and currently used in various organization. Our network monitoring software which is going to surrogate the current but obsolete system addresses all the problem associated with it.

The rapid growth in size and complexity of distributed systems makes performance management tasks such as problem determination – detecting system problems and isolating their root causes – an increasingly important but also extremely difficult task. For example, in IP network management, we would like to quickly identify which router or link has a problem when a failure or performance degradation occurs in the network. In the e-Commerce context, our objective could be to trace the root-cause of unsuccessful or slow user transactions (e.g. purchase requests sent through a web server) in order to identify whether it is a network problem, a web or back-end database server problem, etc.

1.2 Need of the project

Network mapper is a dynamic discovery and mapping tool that helps network administrators/ Data centre admins to visualize their complete network infrastructure with a live network map.

There's no facility available for the network admin for getting a notification when a link goes down. This network mapper will generate an automatic notification when a link goes down.

There are other tools which are present to perform network mapping. But they are generally based on SNMP (Simple Network Management Protocol). The disadvantages associated with SNMP based approach are:

- It is not compatible with all the devices.
- It has a long header size.
- It has security issues in SNMP v1 and v2 i.e there's no encryption of data.
- In SNMP v3 cryptographic calculations is present but it can increase CPU load approx. By 20 %.

1.3 Problem statement

In a large network setup it is hard to visualize the network topology. The network admin faces several problems, such as broken link. The network administrator is not aware of these problems instantly.

Even when he detects that something is going wrong, it becomes an uphill task to trace those broken links on the network. Also, he doesn't know the uptime and downtime of that link, and no records are stored.

There is a requirement for a tool to dynamically generate and monitor network graph and to notify the admin when a broken link is detected with its location and downtime.

1.4 Objective

- To create a network mapping tool, for surveillance and monitoring of network infrastructure using active probing technique.
- To generate notification message to network admin when a link goes down.

1.5 Solution approach

The network mapper provides a tool to visually monitor the entire LAN network. Whenever a device is added in our network (manually), then it will be shown automatically. A message will be generated whenever a link is down. An entry of every device will be maintained in a log file for network administrator.
We are using active probing technique because it is compatible with all the devices, no such security is needed.

Active probing:

This technique relies on traceroute-like probing on the IP address space. These probes report back IP forwarding paths to the destination address. By combining these paths one can infer router level topology for a given POP. Active probing is advantageous in that the paths returned by probes constitute the actual forwarding path that data takes through networks. It is also more likely to find peering links between ISP's.

Active probing selects and sends probes as needed in response to problems that actually occur. It therefore avoids both the difficulty of constructing probes for all possible problems as well as the waste inherent in using probes for problems that in fact never occur. Furthermore, because probes are selected on-line to obtain further information about particular problems that have occurred, they need not circulate regularly throughout the entire network; instead they can be targeted quickly and directly to the points of interest. Active probing may be a powerful and effective technique for problem determination. Probing Technology In the context of network systems management, a probe is a program that executes on a particular machine (called a probe station) by sending a command or transaction to a server or network element and measuring the response. The PING PROGRAM is probably the most popular probing tool that can be used to detect network availability.

Our proposed approach:

Active probing is an active network monitoring technique that has potential for developing effective solutions for fault localization. In this paper we use active probing to present an approach to develop tools for performing fault localization. We discuss various design issues involved and propose architecture for building such a tool. We describe an algorithm for probe set selection for problem detection and present simulation results to show its effectiveness.

We propose an active probing scheme for the task of problem determination. The key idea is to divide problem determination into two steps: detecting occurrence of a problem and then actively probing in order to localize the problem. The first step is to detect whether there is a problem; once occurrence of a problem is detected, the second step, we have developed it such a way that we are pinging all the systems and then we receive ICMP reply. Now, we analyse the ICMP reply and then we identify the code of ICMP packet and if the ICMP code type is 3, then the destination node is down and if it is type 8, It is then destination node is active.

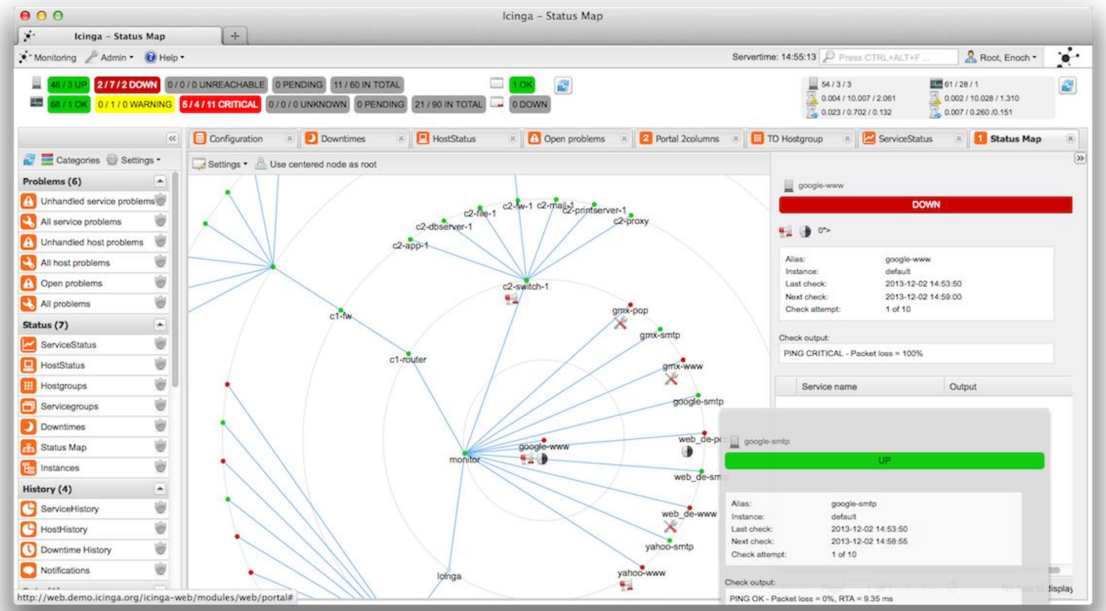
Chapter 2

Background

2.1 Related tools

Icinga

Icinga is an offshoot of Nagios that is currently being rebuilt anew. It offers a thorough monitoring and alerting framework that's designed to be as open and extensible as Nagios is, but with several different Web UI options. Icinga 1 is closely related to Nagios, while Icinga 2 is the rewrite. Both versions are currently supported, and Nagios users can migrate to Icinga 1 very easily.



google-www

DOWN

Alias:	google-www
Instance:	default
Last check:	2013-12-02 14:53:50
Next check:	2013-12-02 14:59:00
Check attempt:	1 of 10

Check output:
PING CRITICAL - Packet loss = 100%

	Service name	Output

google-smtp

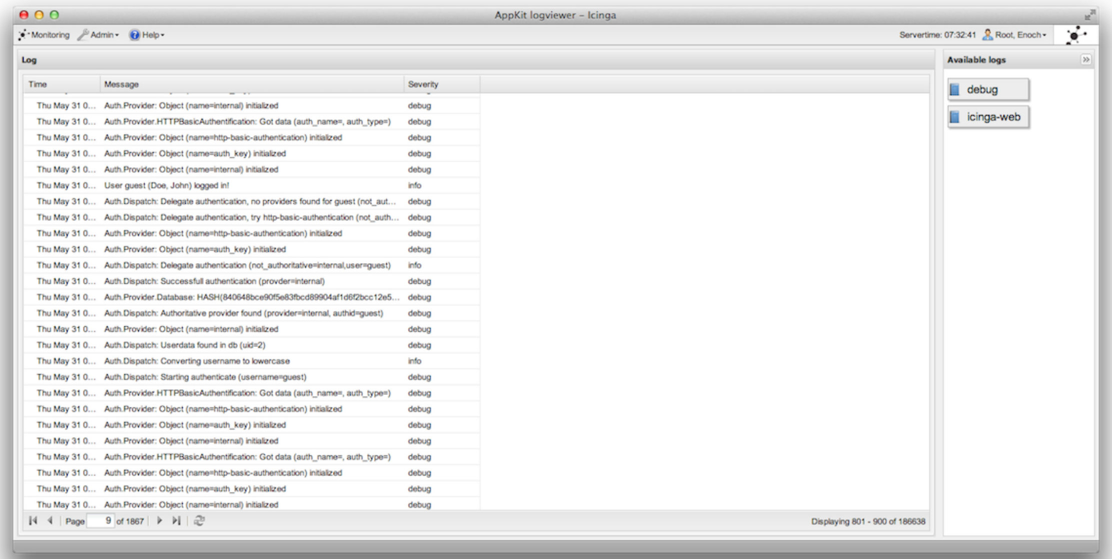
UP

Alias:	google-smtp
Instance:	default
Last check:	2013-12-02 14:53:50
Next check:	2013-12-02 14:58:55
Check attempt:	1 of 10

Check output:
PING OK - Packet loss = 0%, RTA = 9.35 ms

Fig 2.1

Status map Icinga

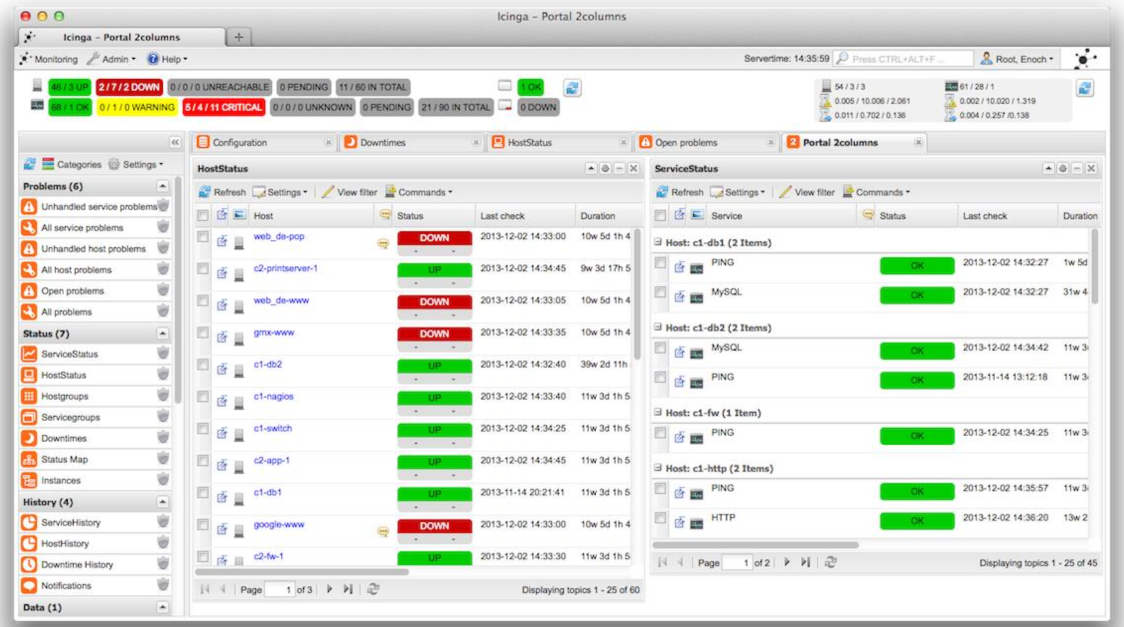


Time	Message	Severity
Thu May 31 0...	Auth.Provider: Object (name=internal) initialized	debug
Thu May 31 0...	Auth.Provider.HTTPBasicAuthentification: Got data (auth_name=, auth_type=)	debug
Thu May 31 0...	Auth.Provider: Object (name=http-basic-authentication) initialized	debug
Thu May 31 0...	Auth.Provider: Object (name=auth_key) initialized	debug
Thu May 31 0...	Auth.Provider: Object (name=internal) initialized	debug
Thu May 31 0...	User guest (Doe, John) logged in!	info
Thu May 31 0...	Auth.Dispatch: Delegate authentication, no providers found for guest (not_aut...	debug
Thu May 31 0...	Auth.Dispatch: Delegate authentication, try http-basic-authentication (not_auth...	debug
Thu May 31 0...	Auth.Provider: Object (name=http-basic-authentication) initialized	debug
Thu May 31 0...	Auth.Provider: Object (name=auth_key) initialized	debug
Thu May 31 0...	Auth.Dispatch: Delegate authentication (not_authoritative=internal,user=guest)	info
Thu May 31 0...	Auth.Dispatch: Successfull authentication (provder=internal)	debug
Thu May 31 0...	Auth.Provider.Database: HASH(840648bce90f5e83fbcd89904af1d6f2bcc12e5...	debug
Thu May 31 0...	Auth.Dispatch: Authoritative provider found (provider=internal, authid=guest)	debug
Thu May 31 0...	Auth.Provider: Object (name=internal) initialized	debug
Thu May 31 0...	Auth.Dispatch: Userdata found in db (uid=2)	debug
Thu May 31 0...	Auth.Dispatch: Converting username to lowercase	info
Thu May 31 0...	Auth.Dispatch: Starting authenticate (username=guest)	debug
Thu May 31 0...	Auth.Provider.HTTPBasicAuthentification: Got data (auth_name=, auth_type=)	debug
Thu May 31 0...	Auth.Provider: Object (name=http-basic-authentication) initialized	debug
Thu May 31 0...	Auth.Provider: Object (name=auth_key) initialized	debug
Thu May 31 0...	Auth.Provider: Object (name=internal) initialized	debug
Thu May 31 0...	Auth.Provider.HTTPBasicAuthentification: Got data (auth_name=, auth_type=)	debug
Thu May 31 0...	Auth.Provider: Object (name=http-basic-authentication) initialized	debug
Thu May 31 0...	Auth.Provider: Object (name=auth_key) initialized	debug
Thu May 31 0...	Auth.Provider: Object (name=internal) initialized	debug

Page 9 of 1867 Displaying 801 - 900 of 186638

Fig 2.2

Log viewer Icinga



Host	Status	Last check	Duration
web_de-pop	DOWN	2013-12-02 14:33:00	10w 5d 1h 4
c2-printserver-1	UP	2013-12-02 14:34:45	9w 3d 17h 5
web_de-www	DOWN	2013-12-02 14:33:05	10w 5d 1h 4
gmx-www	DOWN	2013-12-02 14:33:35	10w 5d 1h 4
c1-db2	UP	2013-12-02 14:32:40	39w 2d 11h
c1-nagios	UP	2013-12-02 14:33:40	11w 3d 1h 5
c1-switch	UP	2013-12-02 14:34:25	11w 3d 1h 5
c2-app-1	UP	2013-12-02 14:34:45	11w 3d 1h 5
c1-db1	UP	2013-11-14 20:21:41	11w 3d 1h 5
google-www	DOWN	2013-12-02 14:33:00	10w 5d 1h 4
c2-fw-1	UP	2013-12-02 14:33:30	11w 3d 1h 5

Page 1 of 3 Displaying topics 1 - 25 of 60

Service	Status	Last check	Duration
Host: c1-db1 (2 Items)			
PING	OK	2013-12-02 14:32:27	1w 5d
MySQL	OK	2013-12-02 14:32:27	31w 4
Host: c1-db2 (2 Items)			
MySQL	OK	2013-12-02 14:34:42	11w 3
PING	OK	2013-11-14 13:12:18	11w 3
Host: c1-fw (1 Item)			
PING	OK	2013-12-02 14:34:25	11w 3
Host: c1-http (2 Items)			
PING	OK	2013-12-02 14:35:57	11w 3
HTTP	OK	2013-12-02 14:36:20	13w 2

Page 1 of 2 Displaying topics 1 - 25 of 45

Fig 2.3

Portal 2 columns Icinga

NeDi

NeDi may not be as well known as some of the others, but it’s a great solution for tracking devices across a network. It continuously walks through a network infrastructure and catalogs devices, keeping track of everything it discovers. It can provide the current location of any device, as well as a history.

NeDi can be used to locate stolen or lost devices by alerting you if they reappear on the network. It can even display all known and discovered connections on a map, showing how every network interconnect is laid out, down to the physical port level.

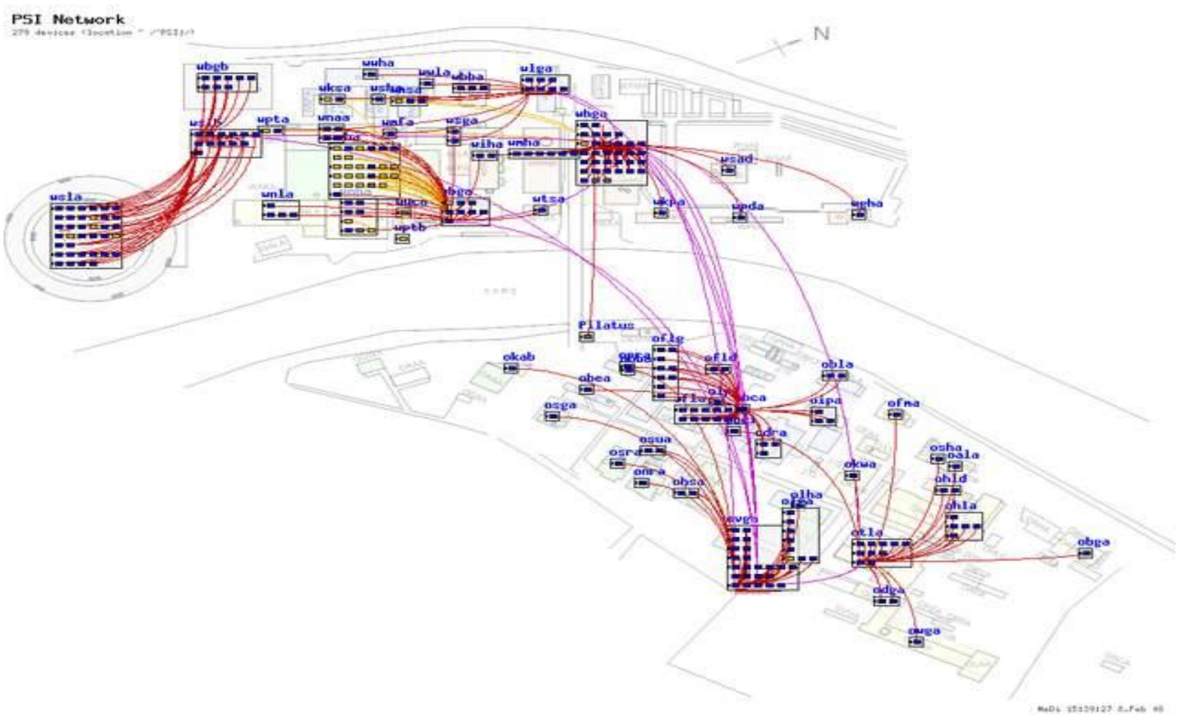
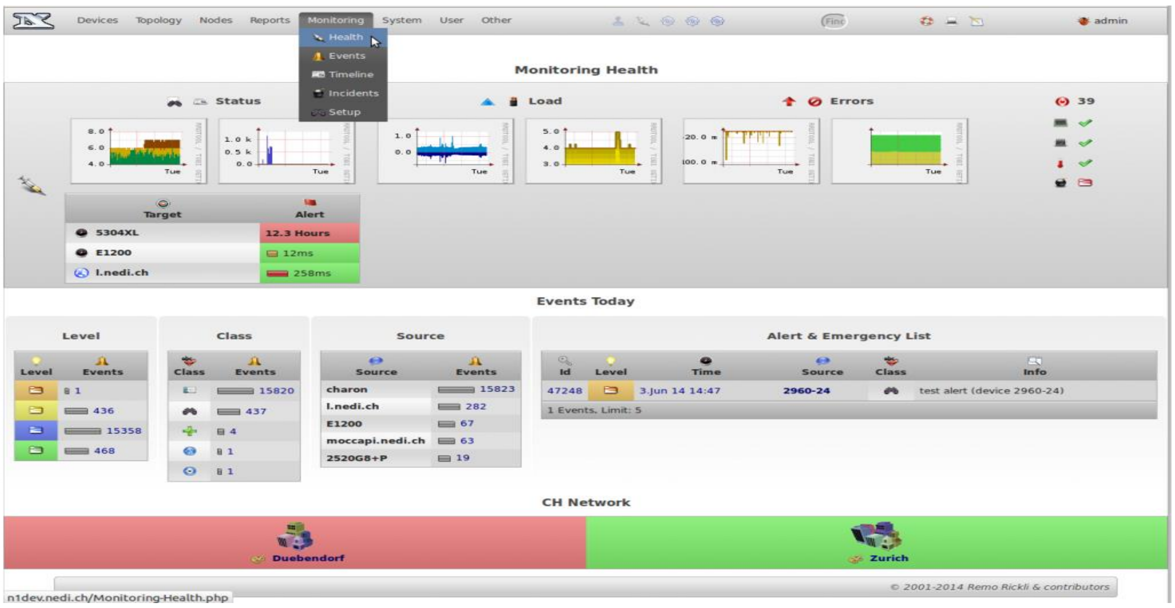


Fig 2.4

Network diagram Nedi



Monitoring Health

Target	Alert
5304XL	12.3 Hours
E1200	12ms
l.nedi.ch	258ms

Events Today

Level

Level	Events
🟧	1
🟨	436
🟦	15358
🟩	468

Class

Class	Events
	15820
	437
	4
	1
	1

Source

Source	Events
charon	15823
l.nedi.ch	282
E1200	67
moccapi.nedi.ch	63
2520G8+P	19

Alert & Emergency List

Id	Level	Time	Source	Class	Info
47248	🟧	3.Jun 14 14:47	2960-24		test alert (device 2960-24)
1 Events, Limit: 5					

CH Network

Duebendorf

Zurich

© 2001-2014 Remo Rickli & contributors

n1dev.nedi.ch/Monitoring-Health.php

Fig 2.5

Network Status Nedi

Zabbix

Zabbix monitors servers and networks with an extensive array of tools. There are Zabbix agents for most operating systems, or you can use passive or external checks, including SNMP to monitor hosts and network devices. You'll also find extensive alerting and notification facilities, and a highly customizable Web UI that can be adapted to a variety of heads-up displays. In addition, Zabbix has specific tools that monitor Web application stacks and virtualization hypervisors.

Zabbix can also produce logical interconnection diagrams detailing how certain monitored objects are interconnected. These maps are customizable, and maps can be created for groups of monitored devices and hosts.

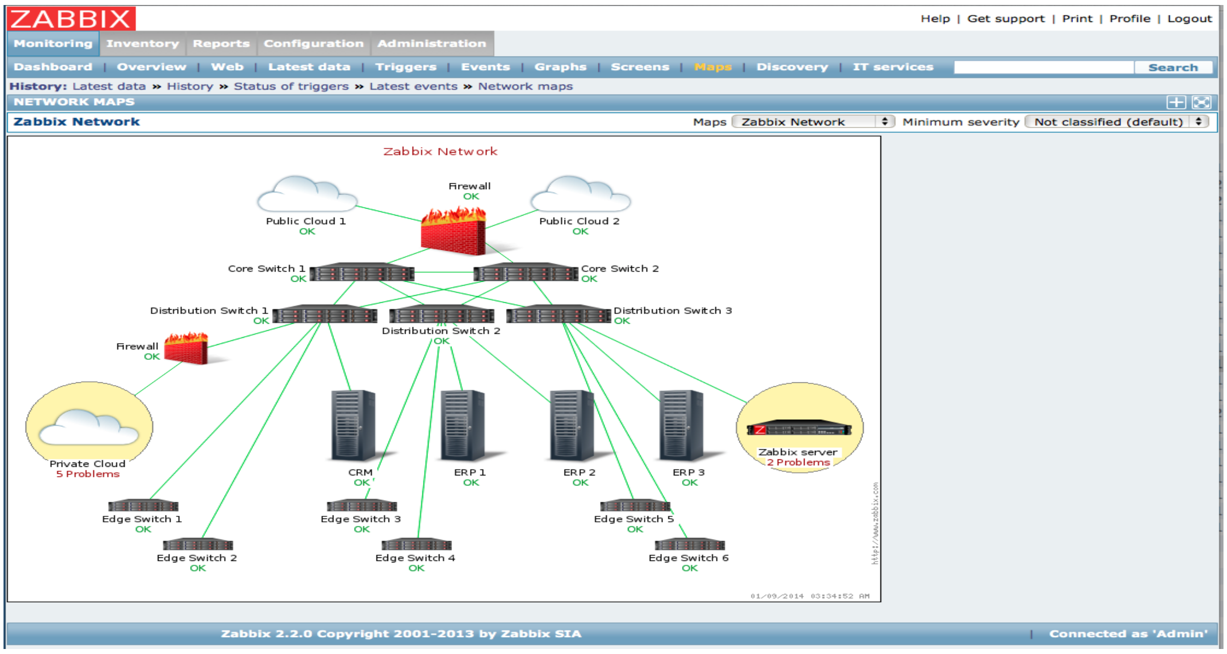
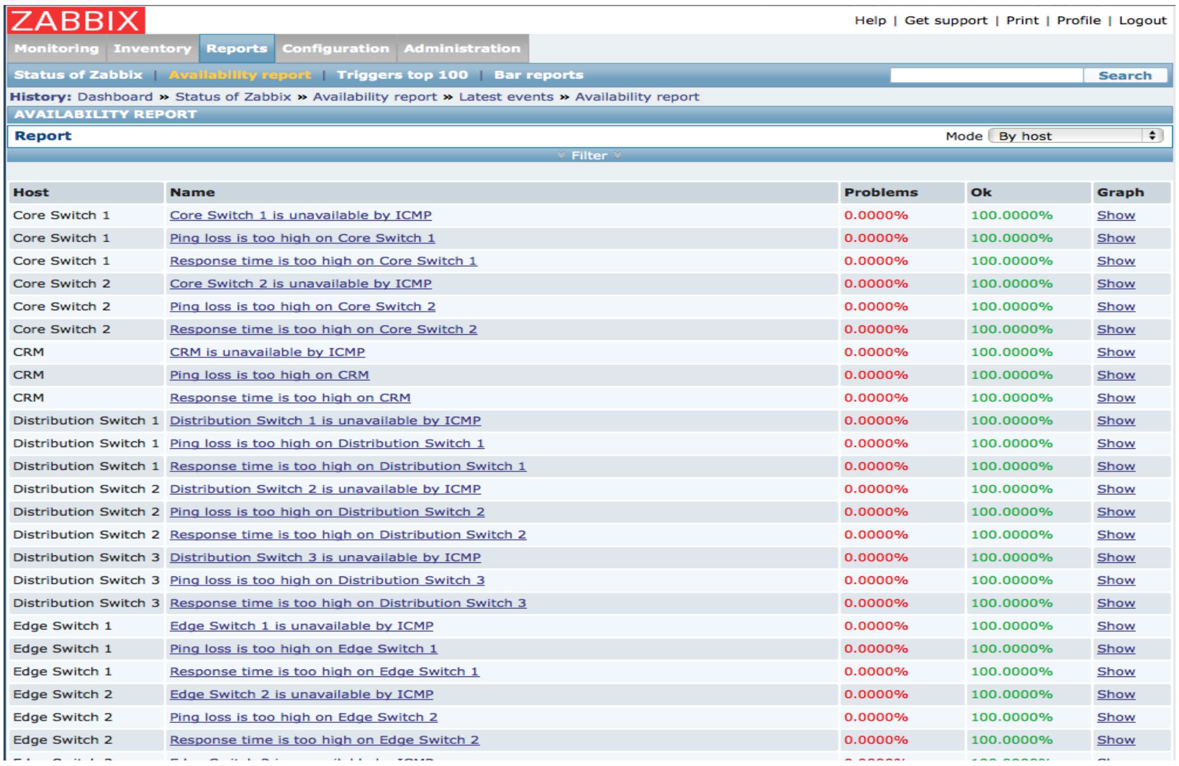


Fig 2.6

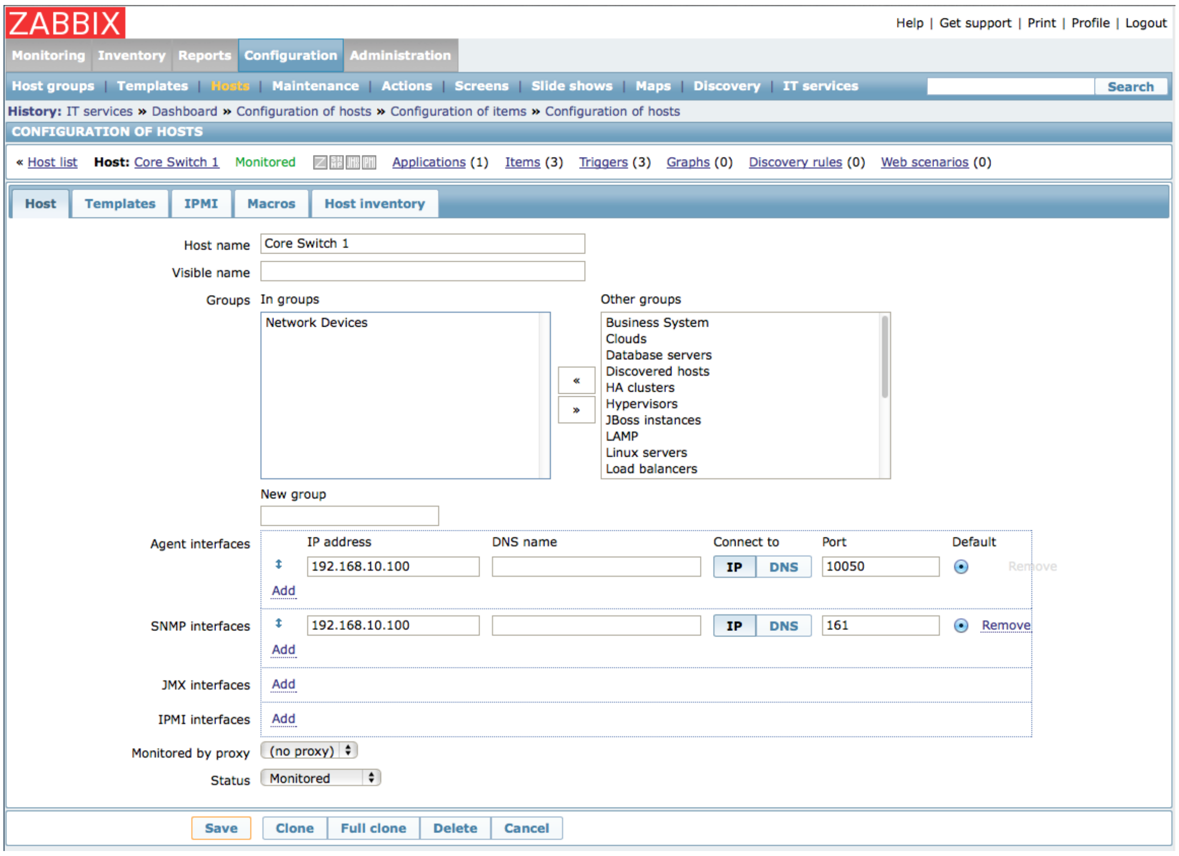
Network diagram Zabbix


ZABBIX
Help | Get support | Print | Profile | Logout
Monitoring Inventory Reports Configuration Administration
Status of Zabbix | Availability report | Triggers top 100 | Bar reports
Search
History: Dashboard » Status of Zabbix » Availability report » Latest events » Availability report
AVAILABILITY REPORT
Report
Mode By host
Filter

Host	Name	Problems	Ok	Graph
Core Switch 1	Core Switch 1 is unavailable by ICMP	0.0000%	100.0000%	Show
Core Switch 1	Ping loss is too high on Core Switch 1	0.0000%	100.0000%	Show
Core Switch 1	Response time is too high on Core Switch 1	0.0000%	100.0000%	Show
Core Switch 2	Core Switch 2 is unavailable by ICMP	0.0000%	100.0000%	Show
Core Switch 2	Ping loss is too high on Core Switch 2	0.0000%	100.0000%	Show
Core Switch 2	Response time is too high on Core Switch 2	0.0000%	100.0000%	Show
CRM	CRM is unavailable by ICMP	0.0000%	100.0000%	Show
CRM	Ping loss is too high on CRM	0.0000%	100.0000%	Show
CRM	Response time is too high on CRM	0.0000%	100.0000%	Show
Distribution Switch 1	Distribution Switch 1 is unavailable by ICMP	0.0000%	100.0000%	Show
Distribution Switch 1	Ping loss is too high on Distribution Switch 1	0.0000%	100.0000%	Show
Distribution Switch 1	Response time is too high on Distribution Switch 1	0.0000%	100.0000%	Show
Distribution Switch 2	Distribution Switch 2 is unavailable by ICMP	0.0000%	100.0000%	Show
Distribution Switch 2	Ping loss is too high on Distribution Switch 2	0.0000%	100.0000%	Show
Distribution Switch 2	Response time is too high on Distribution Switch 2	0.0000%	100.0000%	Show
Distribution Switch 3	Distribution Switch 3 is unavailable by ICMP	0.0000%	100.0000%	Show
Distribution Switch 3	Ping loss is too high on Distribution Switch 3	0.0000%	100.0000%	Show
Distribution Switch 3	Response time is too high on Distribution Switch 3	0.0000%	100.0000%	Show
Edge Switch 1	Edge Switch 1 is unavailable by ICMP	0.0000%	100.0000%	Show
Edge Switch 1	Ping loss is too high on Edge Switch 1	0.0000%	100.0000%	Show
Edge Switch 1	Response time is too high on Edge Switch 1	0.0000%	100.0000%	Show
Edge Switch 2	Edge Switch 2 is unavailable by ICMP	0.0000%	100.0000%	Show
Edge Switch 2	Ping loss is too high on Edge Switch 2	0.0000%	100.0000%	Show
Edge Switch 2	Response time is too high on Edge Switch 2	0.0000%	100.0000%	Show

Fig 2.7

Availability report Zabbix


ZABBIX
Help | Get support | Print | Profile | Logout
Monitoring Inventory Reports Configuration Administration
Host groups | Templates | Hosts | Maintenance | Actions | Screens | Slide shows | Maps | Discovery | IT services
Search
Host Templates IPMI Macros Host inventory
Save Clone Full clone Delete Cancel

History: IT services » Dashboard » Configuration of hosts » Configuration of items » Configuration of hosts

CONFIGURATION OF HOSTS

« Host list **Host:** Core Switch 1 Monitored Applications (1) Items (3) Triggers (3) Graphs (0) Discovery rules (0) Web scenarios (0)

Host name	Core Switch 1
Visible name	
Groups	In groups: Network Devices
	Other groups: Business System, Clouds, Database servers, Discovered hosts, HA clusters, Hypervisors, JBoss instances, LAMP, Linux servers, Load balancers
	New group

	IP address	DNS name	Connect to	Port	Default	
Agent interfaces	192.168.10.100		[✓ IP] DNS	10050	●	Remove
	Add					
SNMP interfaces	192.168.10.100		[✓ IP] DNS	161	●	Remove
	Add					
JMX interfaces	Add					
IPMI interfaces	Add					

Monitored by proxy	(no proxy)
Status	Monitored

Fig 2.8

Configuration of host Zabbix

Cheops

Cheops-ng is a Network management tool for mapping and monitoring your network. It has host/network discovery functionality as well as OS detection of hosts. Cheops-ng has the ability to probe hosts to see what services they are running. On some services, cheops-ng is actually able to see what program is running for a service and the version number of that program.

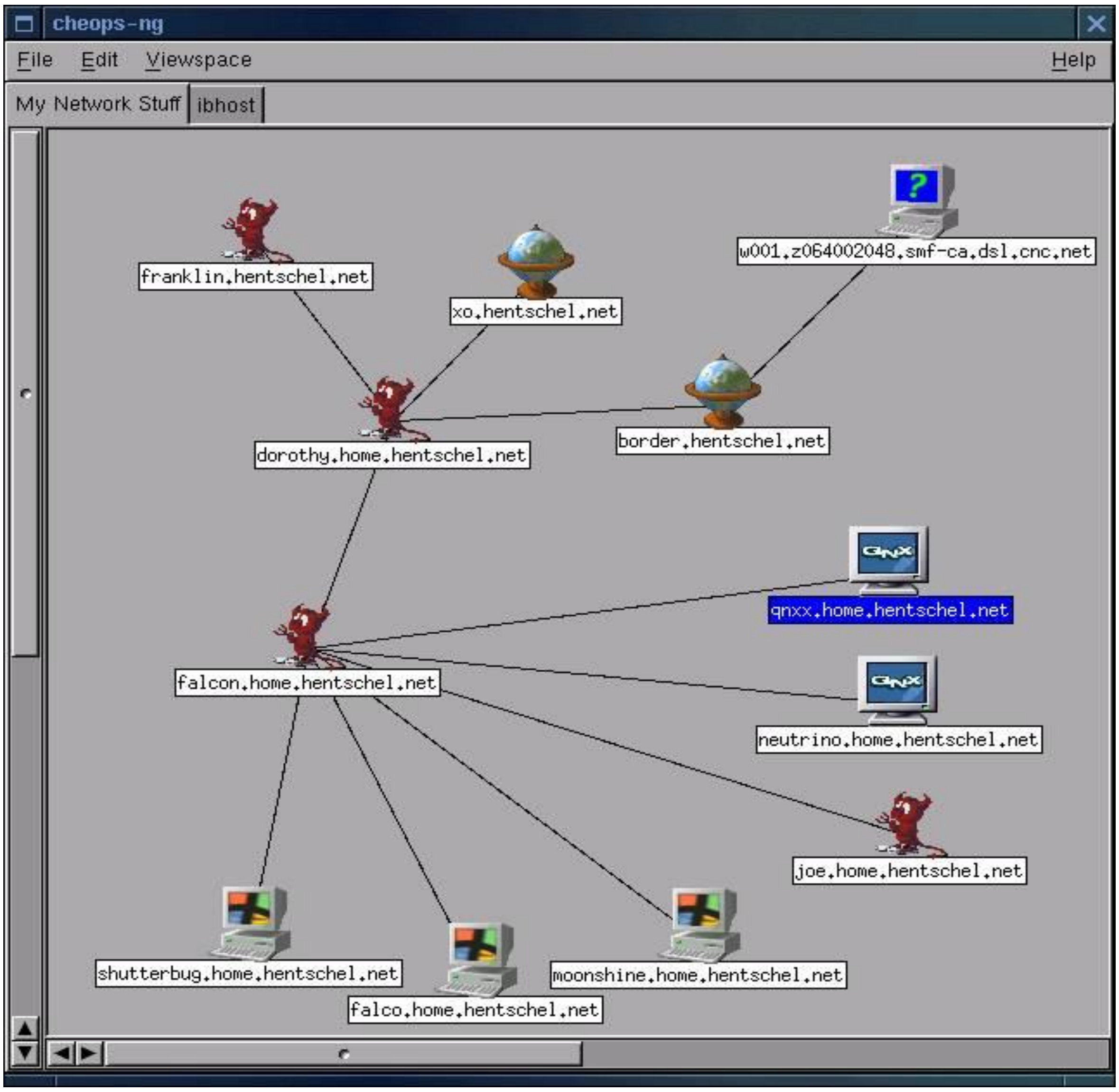


Fig 2.9

Network diagram Cheops

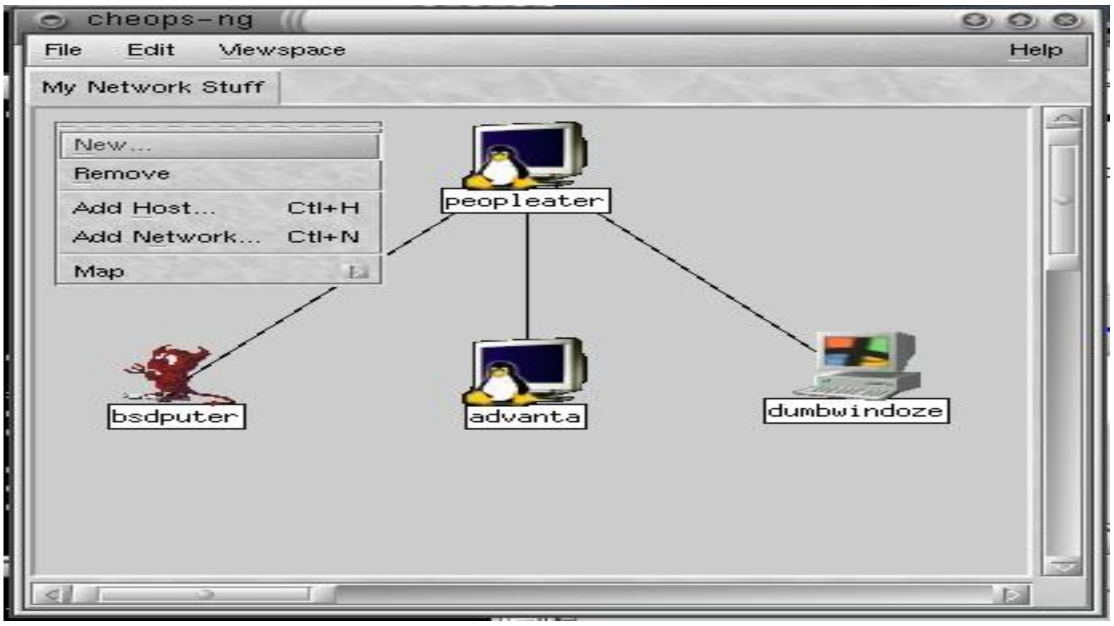


Fig. 2.10 Editing network Cheops

2.2 Literature review

Problem determination is one of the most important tasks in managing networking systems. Probing (both at the transaction and network levels) has been widely used for assessing compliance with Service Level Agreements and locating problems in networking systems. However a probing scheme which uses a fixed setoff regularly scheduled probes can be expensive in terms of the number of synthetic transactions needed, especially for the task of problem determination. This paper introduces an active probing scheme to reduce the number of probes needed. Our key idea is to divide the problem determination task into two steps. We first use relatively small number of fixed, regularly scheduled, probes for detecting that problem has occurred. In the second phase, once occurrence of a problem is detected, additional probes are issued on-the-fly to acquire additional information until the problem is localized. We develop algorithms for selecting an optimal setoff probes for problem detection and choosing which probes to send next based on what is currently known. We demonstrate through both analysis and simulation that the active probing scheme can greatly reduce the number of probes and the time needed for localizing the problem when compared with a non-active probing scheme.

Introduction :

The rapid growth in size and complexity of distributed systems makes performance management tasks such as problem determination – detecting system problems and isolating their root causes – an increasingly important but also extremely difficult task. For example, in IP network management, we would like to quickly identify which router or link has a problem when a failure or performance degradation occurs in the network. Ianthe e-Commerce context, our objective could be to trace the root-cause of unsuccessful or slow user transactions (e.g. purchase requests sent through a web server) in order to identify whether it is a network problem, a web or back-end database server problem, etc. Another example is real-time monitoring, diagnosis and prediction of the "health" of a large cluster system containing hundreds or thousands of workstations performing distributed computations (e.g., Linux clusters or GRID-computing systems).Two general approaches are commonly used for problem determination. The first is event correlation, in which every managed device is instrumented to1emit an alarm when its status changes. By correlating the received alarms a centralized manager is able to identify the problem. However, this approach usually requires heavy instrumentation, since each device needs to have the ability to send out the appropriate alarms.

Also it may be difficult to ensure that alarms are sent out, e.g. by advice that is down. To avoid these problems, which arise from using a fixed, "passive "data-gathering approach, a more active probing technology has been developed, which allows one to test network and system components in order to provide more accurate and cost efficient problem determination. Aerobe is a test transaction whose outcome depends on some of the system's components; accurate diagnosis can be achieved by appropriately selecting the probes and analyzing the probe outcomes. Previous work has focused on selecting probes in an off-line, pre-planned fashion. Prior information about network and system dependencies, which problems need to be detected most urgently (perhaps because they are more important or more likely to occur), and other forms of prior knowledge are used to construct set of probes that is small (thereby reducing probing costs such as data storage requirements and network load) yet provides extensive coverage so that problems can be diagnosed. These probes are scheduled to run periodically to provide information about what problems may be occurring. A typical example is IBM’s EPP technology

Using pre-planned probe sets suffers from considerable limitations. Because the probe set is computed off-line, it needs to be able to diagnose all possible problems which might occur. However in practice constructing such a probe set can be quite difficult, because one must envisage all problems one would like to be able to diagnose and construct probes for them. A pre-planned probe set may also be very wasteful, because many problems that might occur do not in fact ever happen. Probes to detect such problems enlarge the probe set unnecessarily, but knowing which probes can safely omitted can usually only be determined on-line by monitoring which problems in fact occur.

Another disadvantage of pre-planned probe sets is that because the probes run periodically at regularly scheduled intervals, there may be a considerable delay in obtaining information when a problem occurs. It is clearly desirable to detect the occurrence of problem as quickly as possible. Furthermore, once the occurrence of a problem has been detected, additional information may be needed to diagnose the problem precisely. This information may not be obtainable from the results of the pre-planned probes - additional probes may need to be sent to obtain it. These probes should be appropriately selected “on-demand”, based on the results of the previous probes. Our works develops a methodology called active probing that addresses these limitations. This involves probing in an interactive mode, where probe results are analyzed to determine the most likely diagnosis, and then additional probes are selected and sent in order to gain further information. This process may repeat - once additional probe results are obtained, the diagnosis is refined, and, if necessary, more probes are selected, and so on, until the problem is completely determined. The idea of this approach is to “ask the right questions at the right time”.

Active probing selects and sends probes as needed in response to problems that actually occur. It therefore avoids both the difficulty of constructing probes for all possible problems as well as the waste inherent in using probes for problems that infect never occur. Furthermore, because probes are selected on-line to obtain further information about particular problems that have occurred, they need not circulate regularly throughout the entire network; instead they can be targeted quickly and directly to the points of interest.

Thus fewer probes are needed than in a pre-planned approach, allowing for a considerable reduction in probing costs.

Implementing active probing requires developing solutions for the following issues:

1. A small probe set must be pre-selected, so that when a problem occurs we can detect that something has gone wrong.

2. The probe results must be integrated and analyzed, to determine the most-likely network state.

3. The “most-informative” probes to send next must be selected, based on the analysis

of previous probe results.

4. This process must be repeated until the problem diagnosis task is complete.

In this paper we describe efficient solutions for each of these issues and integrate them into a practical system for active probing. We also analyze the costs and benefits of active probing and show experimentally that active probing substantially reduces the number of probes needed to diagnose problems when compared with pre-planned probing. Our preliminary results suggest that active probing may be a powerful and effective technique for problem determination.

3.1.2 Existing discovery systems:

Initial topology discovery systems, such as Mercator, based on hop-limited probing methods sent probes out from a single location. Probing in such a way is likely to result in a tree-like topology and will tend to miss out “cross-links”, i.e. traversal branches. Subsequent work therefore adopted multiple source approaches, in which multiple probing sources are used. Quite like looking glasses, a large number of public traceroute servers have been made available and can be found at. Initially their use was mainly intended for debugging purposes but they can also be used by topology discovery tools, although their web interfaces sometimes lack convenience.

An example of such a platform is Planet Lab which has currently 694 machines hosted on 335 different sites which provides a network testbed for many active research projects of all kinds, covering file sharing, content distribution networks, QoS overlays, anomaly detection mechanisms, and network measurement tools. An implementation making use of Planet Lab is script route aimed at providing traceroute servers' accessibility but with the flexibility of running a variety of measurement tools. Other approaches have deployed their own measurement infrastructure, like the well-known Skitter project of CAIDA which has between 20 and 30 available monitors' world-wide providing traffic measurements services for researchers. The Active Measurement Project (AMP) of the National Laboratory for Applied Network Research (NLANR), although recently terminated in July 2006, took a slightly different approach by letting 150 monitors, mainly deployed in the United States, probe each other in a full mesh in order to obtain dense coverage of the underlying network. Another project, Rocket fuel, concentrates on discovering the internal topologies of ISPs. A more recent trend has been to move towards a more distributed system as demonstrated by a project like DIMES from the Tel-Aviv University. DIMES relies on the distribution of small portable agents to the community, in a similar spirit to ones such as SETI@home, which perform typical traceroute and ping probes around the globe.

Chapter 3

Requirement Analysis

3.1 Functional requirements

In a large network setup it is hard to visualize the network topology. The network admin faces several problems, such as broken link. The network administrator is not aware of these problems instantly.

The network admin should know about anomalies present in his network instantly and should be able to solve them in quick succession of time. Also the admin should know the status of a link whether it is up or down but without a proper tool it is very tedious job for the network admin.

What he requires is a tool which can generate a graph for the given network so that he can visualize his network easily and can find the anomalies present in it instantly.

Functional requirements -

1. **Visually monitor entire LAN network.**

Every device in a network will be shown as an icon, the live link connecting two nodes in a green connector and the dead link in red connector. For every type of device we have a defined icon, such as for routers, desktops and servers.

2. **Dynamic identification.**

If a device is added in our network (manually), then it will be shown automatically. The system will refresh itself after defined time period.

3. **Automatic message generation**

Whenever any link is down, (which will be shown in red on map) an automatic message/SMS will be generated to the network administrator.

4. **Maintain log**

The system will maintain a log of links, indicating the uptime and down time of the network device. It will also show the current status of link.

Additional functionalities (for CCTV monitoring)

Identify if the CCTV camera is blocked or not using image processing in matlab.

3.2 Diagrams:

1) Use case Diagram

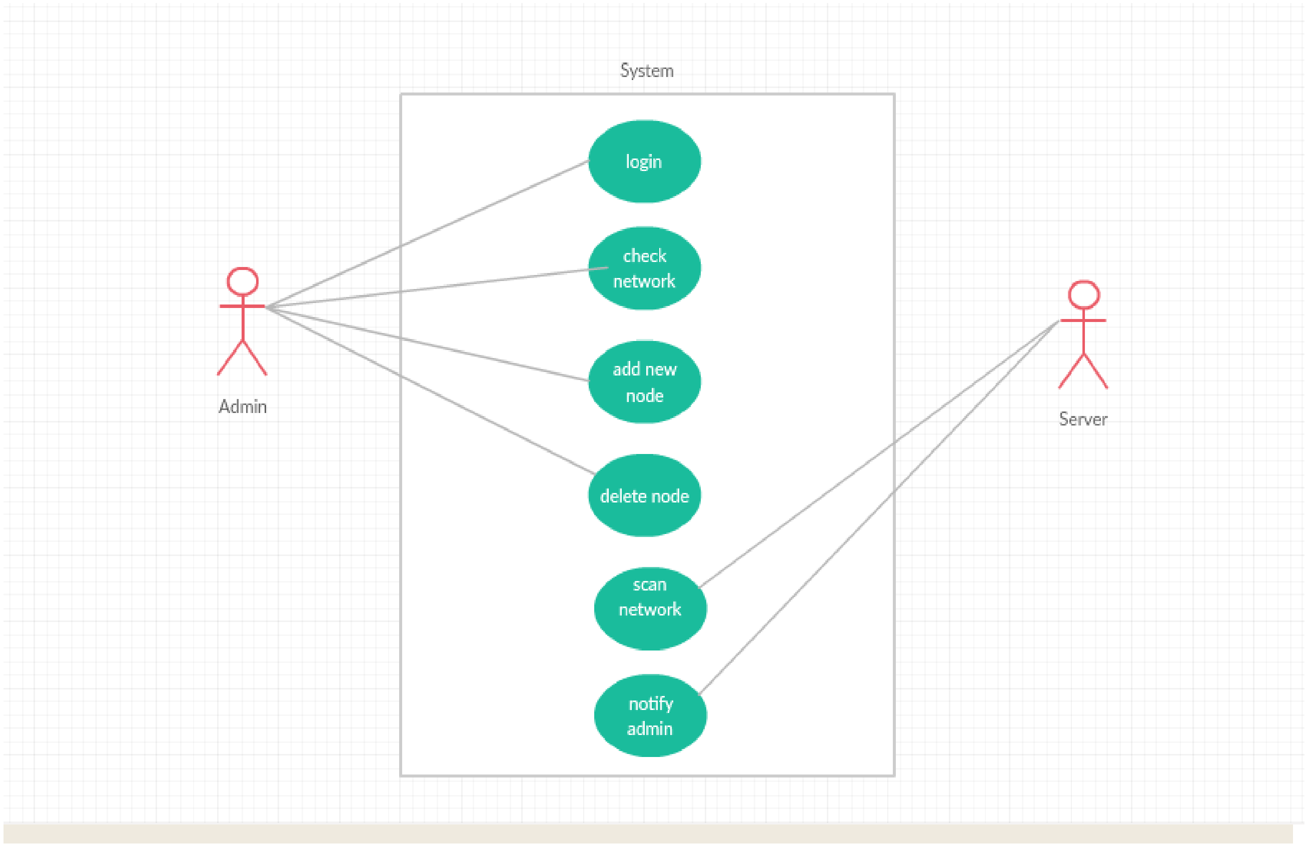


Fig 3.1

Use case diagram

2) Activity Diagrams

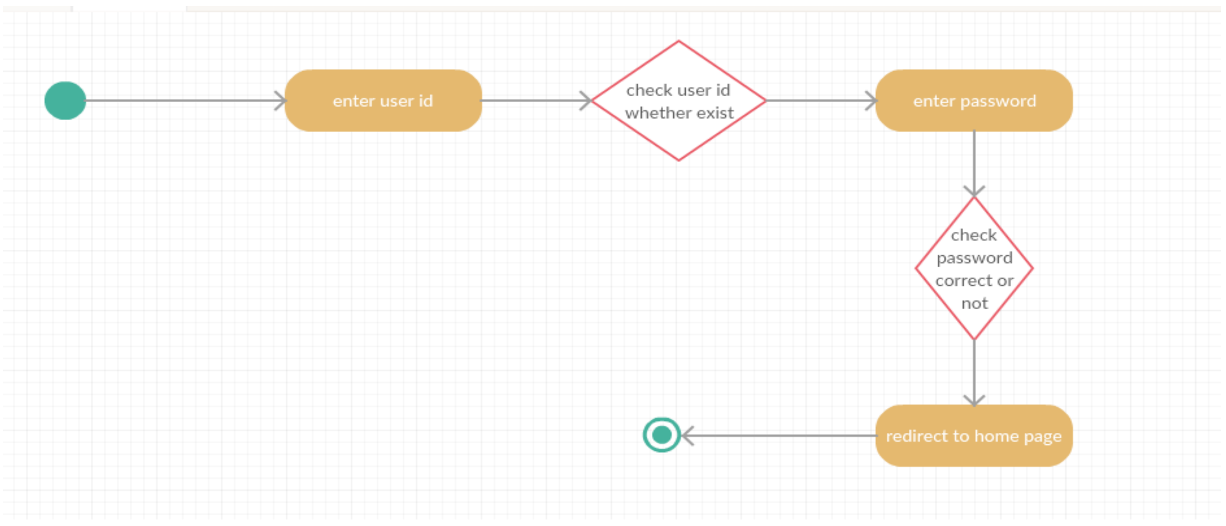


Fig 3.2

Activity diagram – login

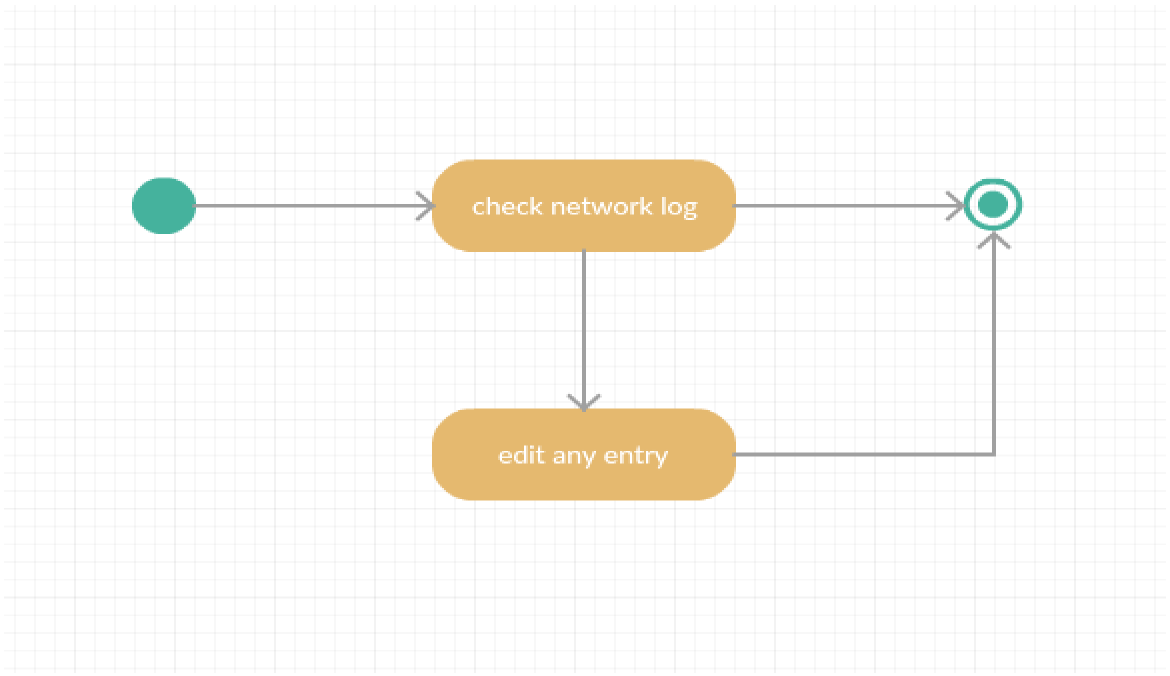


Fig 3.3

Activity diagram – check network

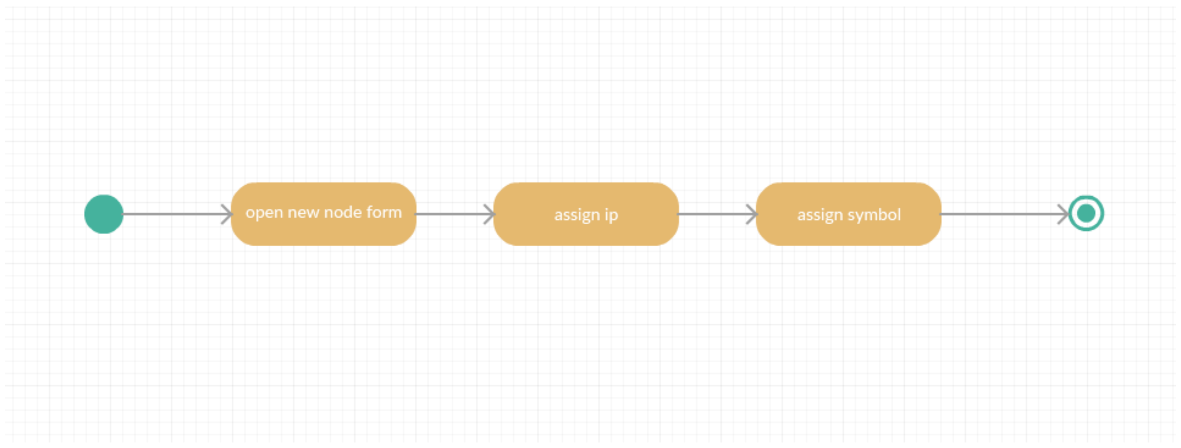


Fig 3.4

Activity diagram – addition of nodes

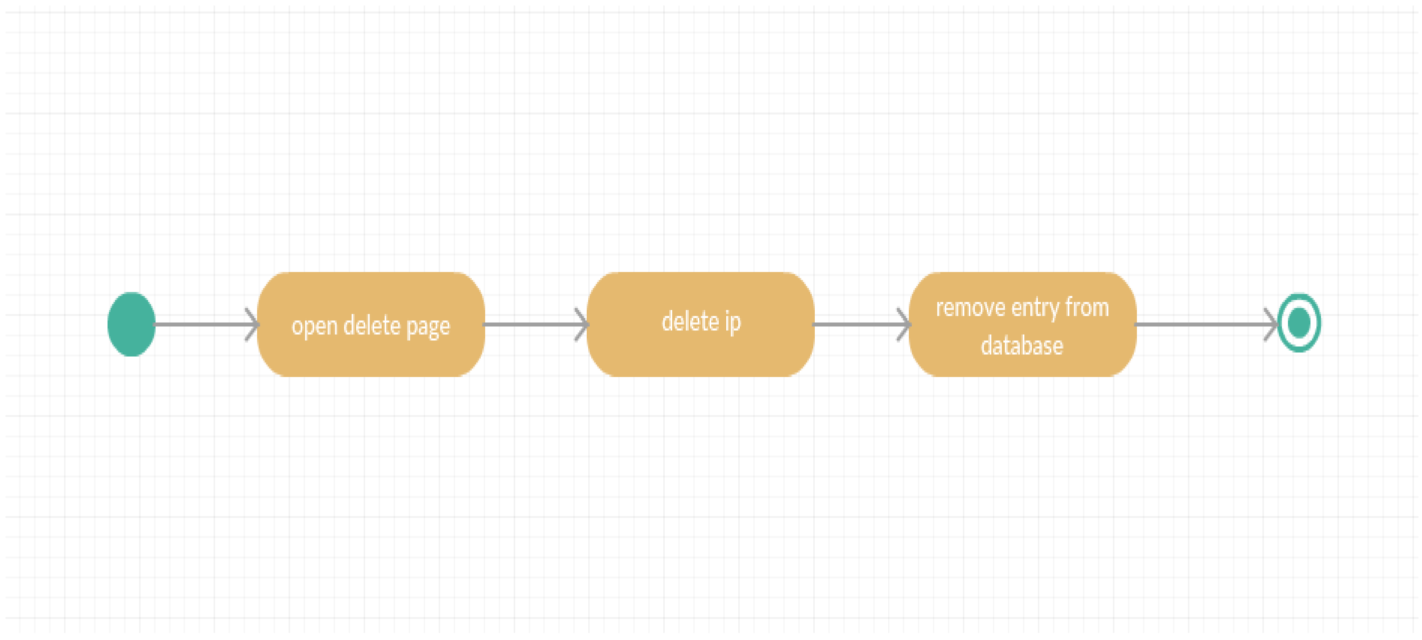


Fig 3.5

Activity diagram – deletion of nodes

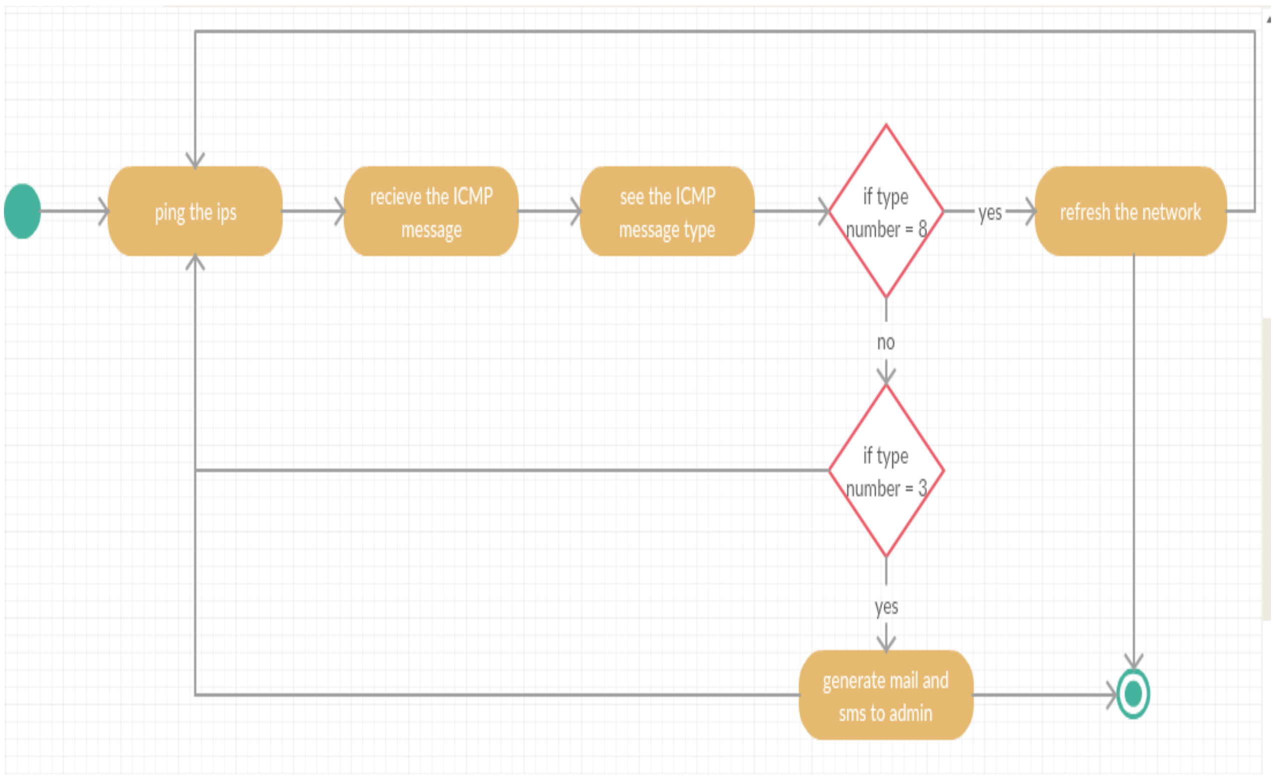


Fig 3.6

Activity diagram – Scan network

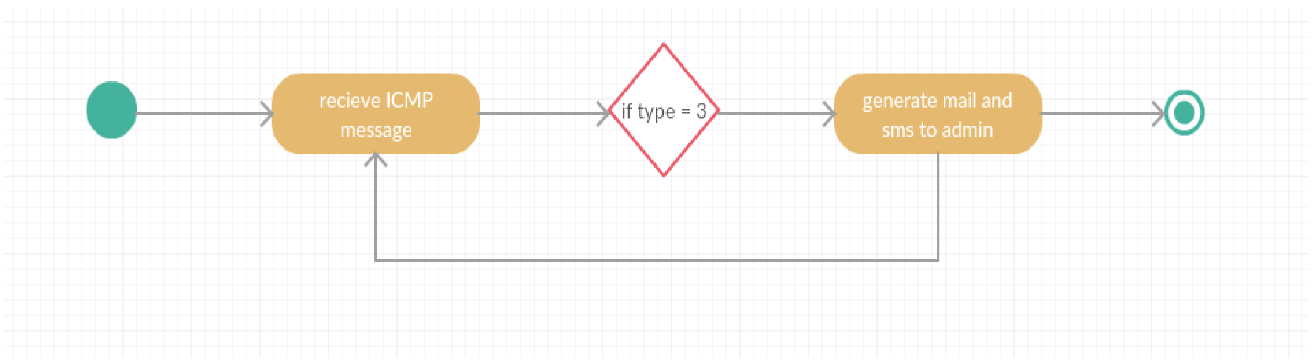


Fig 3.7

Activity diagram – Notify admin

3.3 Feasibility analysis

- Technical feasibility :

Technical feasibility access the current resources and technology, which are required to accomplish user requirements in the software within the allocation time and for this, the software development team ascertains whether the current resources and technology can be upgraded or added in the software to accomplish specified user requirements.

Technical feasibility of the product has been studied under following heads :
Hardware availability: - The application needs a server, camera and computer system(PC).
Software availability: - Brackets, My-SQL.

- Economic Feasibility :

This system requires server, camera for surveillance and computer system.
(I) Hardware cost: - As such no separate hardware is required. We need a server and cameras.
(II) Software cost :- Software needed for development of this system are open source software(Front hand : Html ,Java script and CSS ; Back hand : C++ and SQL,PL-SQL) hence, there is no cost associated with them.

- Operational Feasibility :

Operational feasibility is an measure of how well a proposed system solves the problem. The existing systems have a problem that they are based on SNMP protocol which is not quite secured and also their header size is too large. So the proposed system removes these problem as:-

Here we are using Active Probing to solve the issues involved in previous systems. This proposal is secured such that It does not require any data fetching from the tables. It has nothing to do with Header size problem because it uses ping command, so it will not create any data congestion.

In current system, there is no provision for detect to notify the Admin about the down links. But in our proposed system, if any link is down so it will notify the admin about down links.

- Schedule Feasibility :

A project will fail if it takes too long to be completed before it is useful. Typically this means estimating how long the system will take to develop and if it can be completed in a given time period. Schedule feasibility is a measure of how reasonable the project timetable is.

The system is scheduled accordingly to the needs of the project. It is being divided into two phase I and II.

(i) Phase I - This phase needs completion till design phase as per the predefined schedule, Phase- I is completed as per the expected schedule.

(ii) Phase II - This system has independent front end and back end modules. All identified modules have been clearly defined, divided to team members and the construction has been completed successfully.

CHAPTER 4

DESIGN

4.1. Design Strategy:

Active probing technique is used for implementing our proposed Network Mapper tool. We propose an active probing scheme for the task of problem determination. We have developed it in such a way that firstly all the systems are pinged and then we receive ICMP reply. Now these replies are examined and we grep the destination unreachable .

A graph data structure used to store the network topology in the software. The nodes are added to the graph by the network admin. The non-leaf nodes are either routers or servers. Breadth first search technique is used to traverse the whole tree. We are using BFS technique because it has lower algorithmic constant factor, than DFS technique.

According to the actual graph data stored, the software will detect the active and broken links by using active probing. The software will send probes to detect these links and show them as either green or red line. We have used grey colour lines for undiscovered links.

4.2. Requirements:

The objective of this tool is not just to create a network topology. Most importantly, we want to create and maintain a user friendly tool and compatible for all layer three devices. It should have a GUI through which the Admin can execute each task. The interface should be simple, systematic and clear. It should allow the Admin to actually make easier to work such as generating the graph dynamically, implementing coloured nodes for links for easy recognition and we have assigned different IPs for respective nodes for easy understanding. Each program should be straightforward and should not contain functions that overlap. It should display a clear and systematic nodes with connected links with green colour and broken links with red colour which makes easy for Admin to recognize easily. It should display the front end process such as login, etc. It should display recognition results so that we are able to evaluate and analyse. Internally, Network Mapper have the following processes.

1) Log in process

2) Addition of Nodes

3) Generation of graph.

4) Automatic Alert generation.

5) Dynamic status of graph.

4.3. GUI Environment :

Our GUI Environment is highly-integrated and professional GUI environment in which a user can easily create GUI for any type of projects or dissertations work. It is a high-level programming language that includes two main functions:

1) JSON function.

2) TICK function.

4.4. Architecture Diagram:

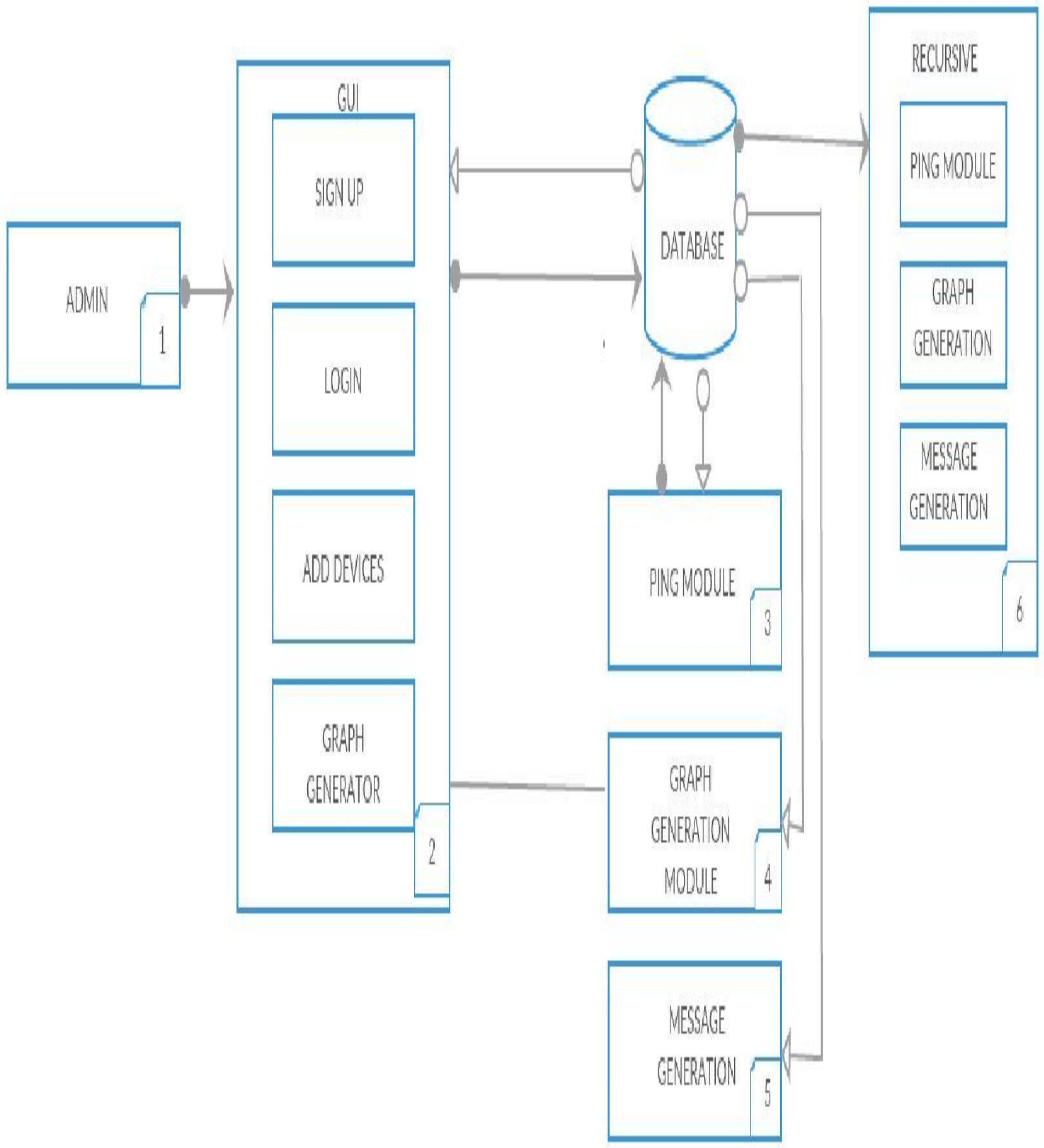


Fig. 4.1. Architecture Diagram

4.5. Ping Module:

In this module we have done parsing of JSON file into JSON objects. The value from these JSON objects are used to fill the array of IP Addresses and the adjacency list. There is a "isAlive" function which returns the Boolean value 0/1 depending on the link status. The output from PING goes to grep such that it scans the destination unreachable keyword and this output is fed to WC utility which counts the number of lines. If the number of lines is 0 then it returns 1 that indicates that the link is up. Initially all link state values are 2 in GraphConnection.json.

4.6. SMS Module:

We have used a SMS gateway "ViaNetSMS" which takes Username, password, message sender, destination address and the message. It takes input from linkstate.json and decodes into Jason objects

It observes all the links of json objects and whose value is 0 or any of the down links and the target of that downed link is appended to the message and the SMS is send to the Admin.

4.7. Front-End Module:

For the purpose of LOG in, we have created a PHP session. After log in, it directs to the Home page.

Our home page has three options ADD node, Generate graph and Send SMS.

4.8. Algorithm Description:

Graph Traversal Algorithm (Breadth First Search):

We have implemented BFS algorithm for graph traversal. The adjacency list which we have created in GraphConnection.JSON is used in BFS Traversal. It checks the status of the devices using Ping utility. The output of PING command shows destination unreachable when the ICMP packet reply from the device is of type 3 or else if it shows destination reachable when the ICMP packet reply from the device is of type 8. The output of PING command is analyzed and based on this output we change the value or status of the link ‘0’as link is down, ‘1’ as link is up and ‘2’ as link is unreachable.

CHAPTER 5
IMPLEMENTATION

This chapter describes how the design has been understood and properly translated into desired system. Implementation phase consists of the coding of the project. Firstly, the information is gathered from the various initial phase of our project. Then this information is used to develop the actual system in the implementation phase. Active Probing is used for the implementation of our project. We created three main module in our project which are Front-End Module, PING Module and SMS Module. In Front End Module we write a program in which it contains options for Add Node, Generate graph, etc. In PING Module we write a program in which it contains C++ code that pings the corresponding IP Address and records its status. In SMS Module we write a program in which all the down links are checked and sends the SMS to Network Admin about the broken links.

5.1. TOOLS DESCRIPTION:

1) Front End:

- HTML:
  We have implemented HTML for Front End which include Log-in page and Admin home page. We have used HTML for the convenience in using Web based applications.
- CSS:
  We have implemented CSS for Web Page designing.
- JavaScript:
  We have implemented D3JS(Data Driven Document) for plotting the Network graph on a web page.

2) Server Side:

- PHP Scripting:
  We have implemented PHP for taking input from JSON files and using it for modifying the link state files. It is also used for executing the C++ executable file.

- C++:

  It is used for Input/Output from JSON files. The main purpose is for Breadth first search for the corresponding graph. It is used for creating graph adjacency list. Our source code uses ping utility to detect the link's state. We have used Json-cpp Parser and source file under our source code.

3) File Format:

- JSON:

  JSON or JavaScript Object Notation is a lightweight text-based open standard designed for human-readable data interchange
  For the purpose of data storage, we have used JSON(JavaScript Object Notation) file format.

4) Server:

- Apache:

  Apache is a freely available Web **server** that is distributed under an "open source" license. Version 2.0 runs on most UNIX-based operating systems (such as Linux, Solaris, Digital UNIX, and AIX)
  The server is used for hosting the Web pages for the front end.

5.2. Model Description:

We have used Active Probing technique for the Network Mapper.Active probing is an active network monitoring technique that has potential for developing effective solutions for fault localization. In this paper we use active probing to present an approach to develop tools for performing fault localization. We discuss various design issues involved and propose architecture for building such a tool. We describe an algorithm for probe set selection for problem detection and present simulation results to show its effectiveness.

An Algorithm for Network Mapper using Active Probing is as follows:
1) Initially, The links are added using GUI, given that the details such as IP Address, ID, Parent ID, device name.

2) Using these graph connections, the nodes are traversed using BSF and the link states are detected by analyzing the output of PING utility.

3) The link states are then used to plot the NETWORK GRAPH.

4) Finally, the SMS is send to the Network Admin regarding the information about the down links.

5.3. Working Diagram:

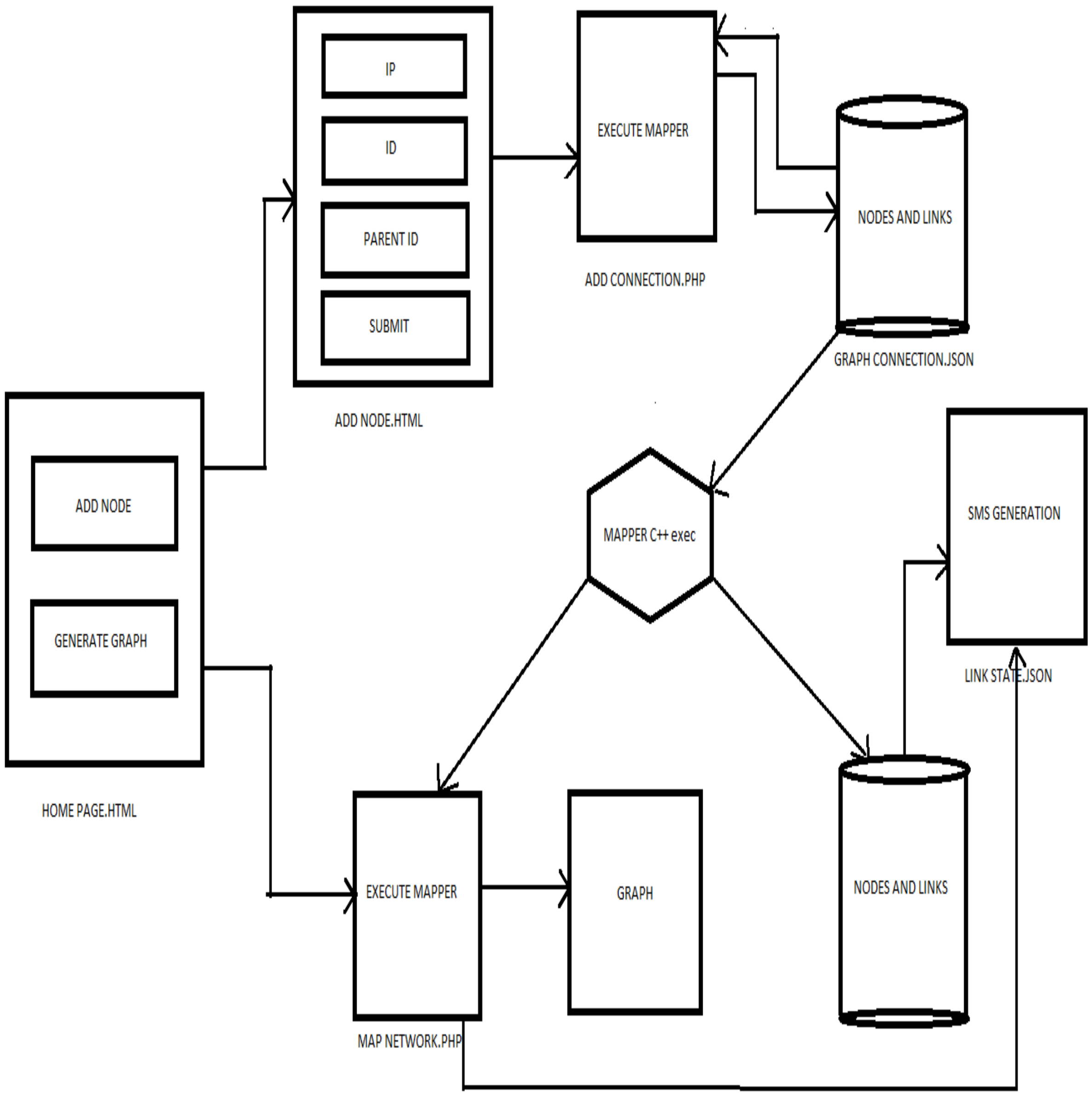


Fig. 5.1. Working diagram of proposed Network Mapper Tool

CHAPTER 6

TESTING AND RESULTS

6.1. Testing strategy:

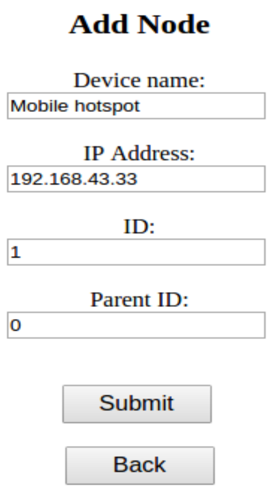
Initially, we have performed unit testing in which single components are tested.

The Unit Testing is performed under three modules:

1) Add node/ Delete Node Testing:

Under this testing, we have just assigned an IP of the devices and stored this IP to a file. We have created a HTML page for taking the inputs from the Admin. We took IP and ID of the device, Parent ID and name of the device. These inputs are then directed to a JSON file such that we randomly take IP of the devices through our HTML page and then we finally check that the particular entered IPs are stored within the proposed JSON file.

Similarly, in the case of deleting a node, we have created a HTML page for the entry of the ID to be deleted. All the entries related to this ID will be deleted from the JSON file. Now, the test was complete when we found the particular ID was removed from the file.


Add Node

Device name: Mobile hotspot

IP Address: 192.168.43.33

ID: 1

Parent ID: 0

Fig. 6.1. Add node implementation

1) PING module testing:

In this testing, we have assigned static IPs to a file, then our program in take these IPs and the program pings all the defined static IPs. Now, we receive PING reply coming from that IPs. Now, we store these reply's status in another file.

2) Front End Testing:

The file consisting of the PING reply's status is reviewed. This status is then used by The D3JS to determine the broken and up links. Our D3JS determines such that if the link to the particular IP is down then it shows that link to that node by a red line. Similarly, it determines the up-link as of green line. Our main objective is to make the interface so user friendly and easy to understand for the Admin and for this purpose D3JS fulfills our requirements.

The Log-in page of our GUI implementation, the URL is not accessible directly. We have tested this as for maintaining security.

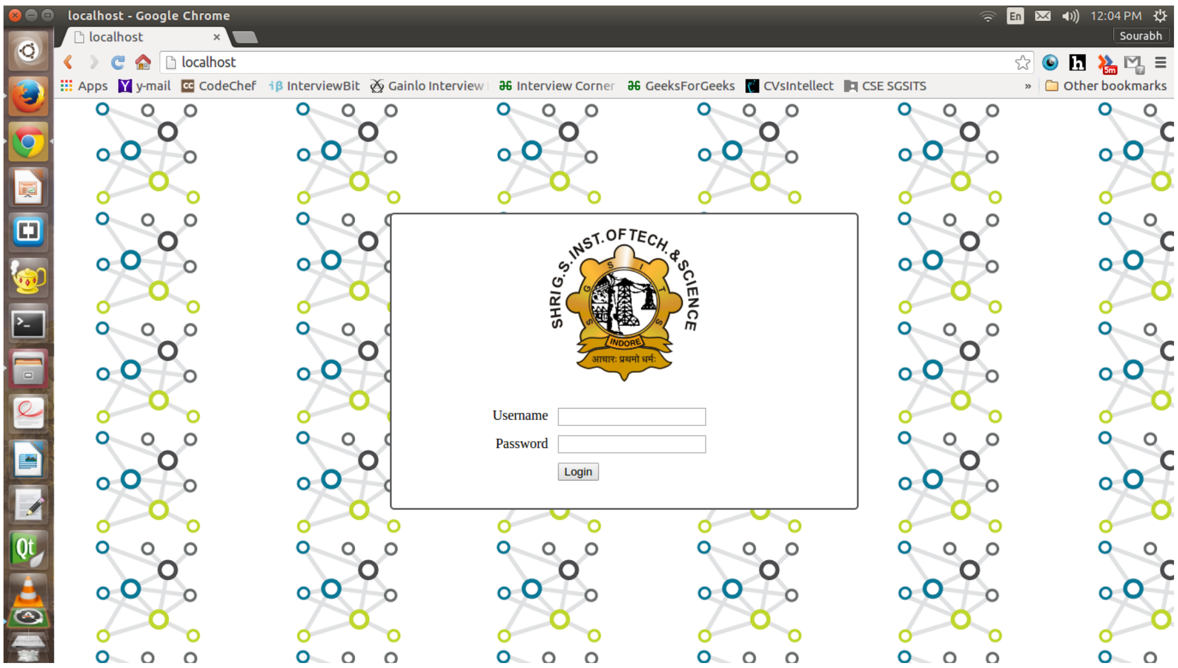


Fig. 6.2. Log-In page

3) SMS Module Testing:

In this testing, we have passed username, password, message, sender's number and receiver's number as parameters into our PHP script for checking whether the SMS is generated or not. Here we put our number as receiver's number and we analyzed the amount of time taken by the PHP script to generate a SMS. After multiple times of testing, the amount of time taken for the same is approximately One minute. This time varies accordingly to the traffic present at the gateway.

6.2. Integrated Testing:

We have implemented this testing on a small scale.

1) Small Scale:

At smaller scale, we have implemented this testing in such a way that the respective IPs of four to five mobile cell phones as well as laptops are analyzed under our testing. Firstly the IPs assigned to these devices are recorded. The graph is generated accordingly with all the links active and live, showing green colored links to respective nodes (Devices and IPs). The main objective was accomplished with the graph generation, showing all the connected devices with their IPs. Now, the testing was done to determine what happens when a particular link to a device is disconnected. So, accordingly we have disconnected a device and we found that the respective disconnected device is displayed as a Red colored link to that particular node. This small scale testing gave us the required confidentiality into our proposed testing and cleared all paths for our large scale testing.

6.3. Results:

After testing stage, the outcome of our proposed Network Mapper tool and its working has been accomplished on a small scale working model. We have connected three different IPs: 10.42.0.65, 10.42.0.83 and 10.42.0.28. The proposed approach to make these IPs visible as connected by green line was achieved. Similarly, to identify the broken link or down link, we have disconnected IP 10.42.0.28 to determine this broken link. Therefore, the link to node of IP 10.42.0.28 is shown by a red line which determines the broken link and SMS is send to Admin regarding the broken link.

STEP 1: 10.42.0.65, 10.42.0.83, 10.42.0.28 - Green links (Connected)

STEP 2: 10.42.0.65, 10.42.0.83 – Green links (Connected) and 10.42.0.28 – Red link (Disconnected)

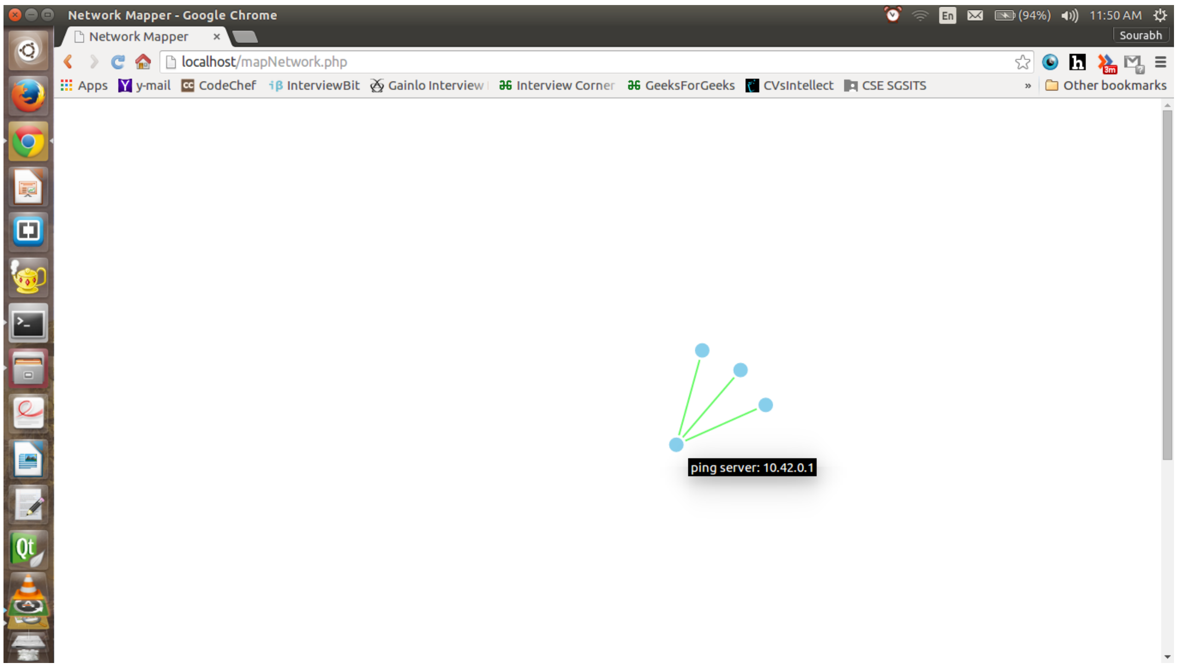


Fig. 6.3. The three IPs connected with three green/active links.

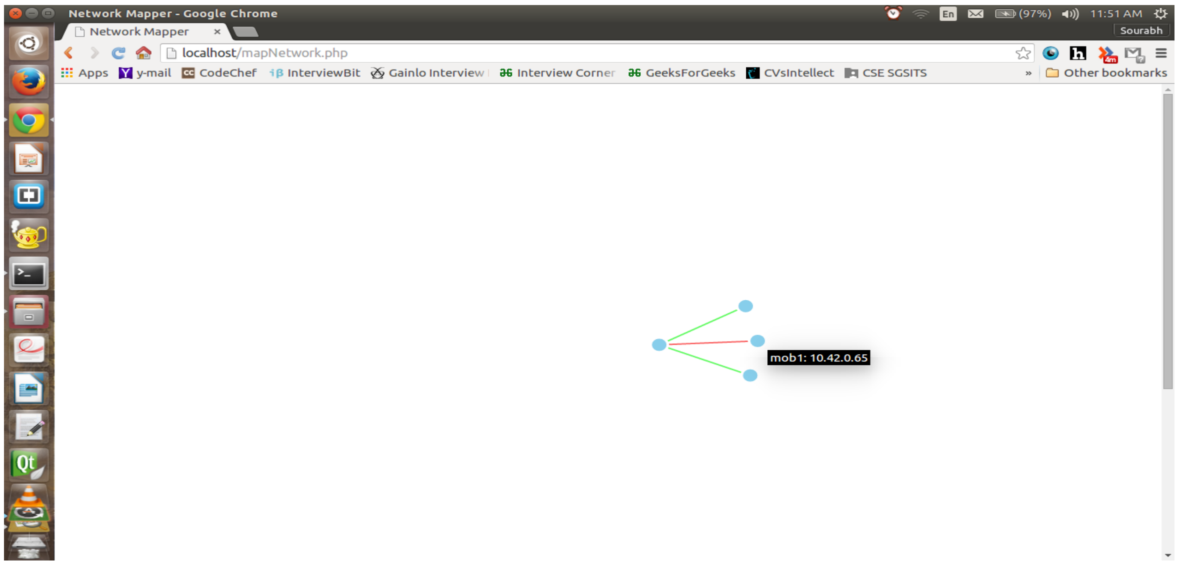


Fig. 6.4. The red link of IP 10.42.0.28 showing disconnection

Chapter 8
Conclusion

In **Phase 1** we have done the study of need of network mapping in the industry. In this study we have gone through current development done in the field of network mapping and what basic technologies used in there.

We studied tools various open source tools like ICINGA, ZABBIX, NEDI,CHEOPS. These all tools are based on SNMP based approach. All these tools are deployed in various industries. But the major disadvantage with the SNMP based approach is that all the hardware doesn’t support SNMP. Various security issues have been listed with SNMP based approach. Also the header size of the probe packet is very large which causes congestion in network and takes time to resolve these large headers at receivers end.

To overcome these drawbacks we have adopted Active Probing based approach. In this approach we continuously ping the system and analyse the ICMP packets. If the ICMP code received is 3 then it points to “Destination Unreachable” which means that the node is down. If the code is 8 then it means “Echo Request and Reply” which means that the node is active.

We have done the feasibility study of this project and reached to a conclusion that this project can be implemented. We have designed Use-Case, Activity and Architecture diagram for the project.

In **Phase 2** our main focus is on development of project using various technologies and methods. We will be working with HTML, CSS, JAVA SCRIPT, C++ and PHP to develop our project. Our back end will be entirely based on C++. All the ping process packet capture will be done by script written in C++ and tcp dump. After analysing the packet code type the entries in the database will be automatically updated and based on that automatic tree generation will be done. A form will be filled by the admin to add a new node in the network and the corresponding entry will be done in the database. And that node will be automatically included in the network tree. Also the whole system will automatically refresh in every 5 Seconds. This proposed approach is successfully achieved. Thus, concluding that our Active probing technology based Network Mapper Tool have surpassed the current prevailing drawbacks of SNMP and its current prevailing tools.

References

1) Andrew S. Tanenbaum, “Computer Networks” Prentice Hall, Hardcover, 3rd edition, March 1996.

2) Douglas E. Comer, “Internetworking with TCP/IP Volume 1: Principles protocols and Architecture”, 6th Edition, 2013.

3) Behrouz A. Forouzan, “Data Communication and Networking”, Published by The McGraw-Hill, 4th Edition, 2007.

4) Larry L. Peterson and Bruce S. Davie, “Computer Networks: A system approach”, Published by Morgan Kaufmann, 5th Edition, 2012.

5) James F. Kurose and Keith W. Ross, “Computer Networking: A top down approach”, Pearson Edition, 6th Edition, 2013.

6) Douglas E. Comer, “Probing TCP implementation”, Department of Computer Sciences, Purdue University, 2002.