AUTOMATED NETWORK FAULT INFERENCE TOOL (ANFIT)

By

Youngsoo Shin

Zuraidah Sulaiman

Nazleeni Samiha Haron @ Baharon

Norhidayah Omar

A group report submitted in partial fulfilment of the requirements for the degree of

Msc Data Communications Networks and Distributed System

Department of Computer Science
University College London
6th September 2004

Supervisor
Professor Mark Handley

GLOSSARY OF TERM

HTTP Hyper Text Transfer Protocol

TCP Transmission Control Protocol

IP Internet Protocol

URL Uniform Resource Locator

URI Uniform Resource Identifier

ARP Address Resolution Protocol

DIG Domain Information Groper

PING Packet INternet Groper

DNS Domain Name System

SSL Secure Sockets Layer

IFCONFIG Interface Configuration

NETSTAT Network Statistics

IPFW Internet Packet Firewall

DOMAIN NAME The domain name of any node in the tree of DNS is the list of labels, starting at that node, working up to the root, using a period ("dot") to separate the labels that might represent sites or a group.

ZONE In a DNS database, a contiguous portion of the domain tree that is administered as a single separate entity by a DNS server. The zone contains resource records for all of the names within the zone.

SOA Start of Authority. Every domain name has an SOA record in its database that indicates basic properties of the domain and the zone that the domain is in.

LDS Local DNS Server

ADS Authoritative DNS server that hosts a primary or secondary copy of zone data. Each zone has at least one authoritative DNS server.

RDS Root DNS Server. Root is the uppermost level in a DNS hierarchy.

TLD Top Level Domain. The next level of DNS hierarchy after root.

GTLD Generic TLD (ex: .com, .org etc)

CTLD Country Code TLD (ex. .my, uk.)

TESTER Prepares testing plans and conducts testing

DEVELOPER Specifies the features for physical design and writes code

TRACKER Keeps track of schedule and all the related tracking documents updated and write reports

COACH Mentors and guides the team members during the implementation stage

ARCHITECT Designs the system architecture

PROJECT MANAGER Oversees the progress of the project

APPLICATION LAYER This layer defines generic available network applications or services that the Internet can support. Below are the widely used network applications and the corresponding network protocols.

Application	**Protocol**
Web	HTTP
Email	SMTP
File Transfer	FTP

TRANSPORT LAYER This layer also known as TCP Layer concerns how data can be reliably transferred over the network. UDP (User Datagram Protocol) is used when speed of data transmission is more important than reliability.

INTERNET LAYER The Internet layer in the TCP/IP reference model is responsible for transferring data between the source and destination computers. The Internet layer accepts data from the Transport layer and passes the data to the Network Interface layer

LINK LAYER This layer also known as Local Network Access Protocol (NAP) is the part of the system that is concerned with how you communicate with your local network, whether is Ethernet or token ring.

PHYSICAL LAYER This is the physical connection whether using a Network Interface Card (NIC) or with a modem to connect to the local network.

WEB SERVICE describes a standardized way of integrating Web-based applications using the XML, SOAP, WSDL and UDDI open standards over an Internet protocol backbone

WEB SERVER A piece of computer software that can respond to a browser's request for a page, and deliver the page to the Web browser through the Internet..

WEB PAGE A document on the World Wide Web. Every Web page is identified by a unique URL.

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

Every beginning has its ending. We are indeed delighted and grateful to the Lord for He has given the chance of a lifetime for us to move one step further in attaining the precious knowledge in University College London. Without His blessing, everything seems impossible.

To materialize this project as a whole, we are indebted to many individuals for the cooperations and contributions, be it directly or indirectly. We wish to express our sincere and ultimate appreciations to Professor Mark Handley for his guidance, unwavering support, mind-opening commentaries, thought-provoking opinions, countless assistance and verily, his tonnes of patience along the journey of this project till the preparation of this manuscript. Significantly, his willingness to shed lights when we are in total darkness and blur, and to expose us to a new programming experience of Python. His friendly gesture and that smiling face he always wears make him approachable. He is truly our idol!

To all the faculty members of the Computer Science Department for their invaluable input in quenching our thirst of knowledge, thank you is just not enough to portray our highest appreciations! A special thanks goes to Dr. Saleem Bhatti, the DCNDS Course Director, for without his assistance, insights and understanding, we would not be able to achieve many things.

We owe a great deal of thanks to our dearest late team member, Youngsoo Shin for without him to coach and manage the group while he was still with us, we will not be able to achieve the point that we are now today thus, this project will not be a success. We wish to express our solemn regret that he is unable to witness this achievement together with us and his presence will be sorely missed. He is truly a remarkable leader and a companion. May he rest in peace.

To all our dearest colleagues in DCNDS 2003/2004 lecture, we wish to express our fervent gratitude for the genuine friendship that has been relayed to us despites the diversity of cultures and backgrounds. We definitely will cherish the hectic yet exciting days with all of them albeit the short moment together. Special acknowledgement to Fahd Shariff for his creative ideas, guides, encouragements and patience along the beautiful friendship.

For Sheng Jiang, a PhD research student in Grid Ipv6 Mobility, we thank him for the practical insights and comments that he has rendered while he acted as the network expert upon the usability testing that has been conducted.

Last but not least, exceptional thanks to the amazing friends residing 109, Camden Road and to our remarkable Malaysian buddies. From the bottom of our hearts, the encouragement and support have indeed maintain ourselves to be strong and in high spirits.

ABSTRACT

The lack of specialized experts in diagnosing network faults, inconsistencies of diagnose results and professional opinions, time-consuming and growing complexity of this task; has motivated the development of our Automated Network Fault Diagnostic System. This system aims to serve as an intelligent diagnostic system that will be able to produce fast, accurate, user-friendly and appropriate suggestions that will assist normal network users and administrators respectively. To ensure the realistic and successful development of the system, we adopt Extreme Programming methodology. Many efforts have been paid to implement a novel and efficient solution to precisely diagnose problems and in timely manner. The methodology has evolved from rule-based systems through case-based systems to more recent model-based systems. Our project is designed upon case-based diagnostic approach as it suggests the use of previously experienced, concrete problem or cases instead of rules or modelling queries evaluation. We propose a system that will provide reactive response on-demand in term of error messages based on inaccessible URL input entered by user. The system will then diagnose the problems based on the formulated inference table that is comprised of pre-defined failure cases and test cases which will be developed via user-defined functions and general network probing tools. From there, we expect the output to be returned in command line error messages. To measure the success of the system, four Key Performance Indicators (KPI) have been identified as evaluation metrics which are coverage, accuracy, time and response. Hence, unit testing, integration testing and usability test will be conducted to obtain the assessment results. We claim that the system could initiate an extensible framework for network services that act as a community support tool. However, at present we narrow down our focus on Web Service application but by all means encouraging and welcoming the extension to other network services or adding in new test cases as future development for the benefit of all network users.

Chapter 1

INTRODUCTION

1.1 PROBLEM STATEMENT AND WORK MOTIVATION

Network problems can be the result of a wide range of issues, from the slight carelessness of unplugged cable, minimal disruptions in service and up to simple configuration problems. The task of diagnosing network related errors can consume undoubtedly an extensive amount of time and thus lead to frustration for users especially those who are not trained as network experts.

Computers on the Internet, as we are all aware, are connected by various networks. The complexity of networking is addressed by dividing the Internet into many layers. Consequently, this layered network approach has shed some light and motivated many designers to beef up their error diagnosing and system reliability mechanisms of tackling the network problems. The computer industry has leaned heavily in the direction of a 7 layer network model also known as ISO/OSI (International Organization for Standardization Open System Interconnection) model long before the Internet has gained its popularity. The 7 layer model has been revised to a 5 layer TCP/IP based Internet Model which has been our basis guideline that motivated us in designing a simple tool yet complete enough to cover the most of commons network errors that may occur at each layer.

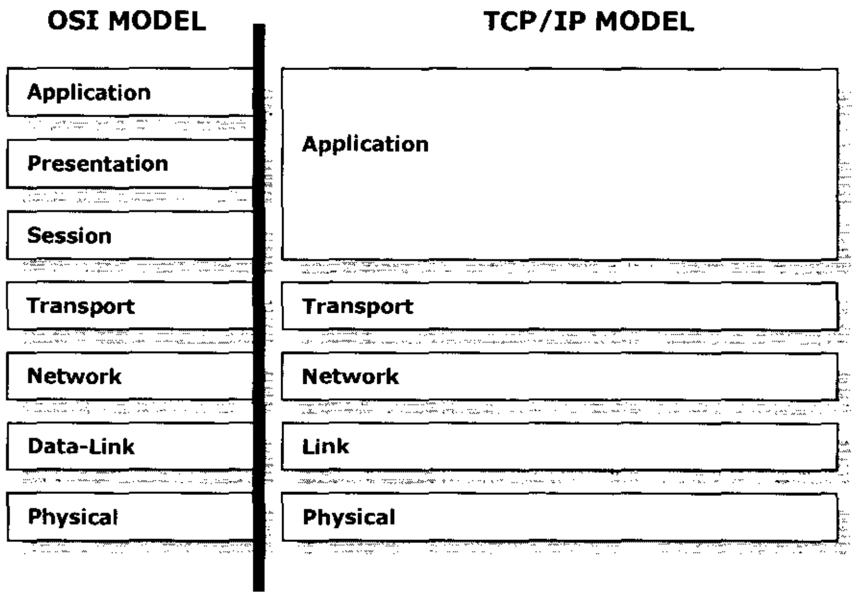

Figure 1.1: OSI Model as compared to the Internet Model

While this is considered an excellent approach to build networks, the layered approach however does have a trade-off in term of error reporting [http://www.microsoft.com]. Applications generally must work independently of the network environment, and lower layers of the network do not generally report meaningful errors to upper layer applications. It is always the case that lower layer network problems can cause upper layer application problems without giving any information about why the errors are occurring. The nature of applications is that they do not possess any sophisticated methods for identifying and correcting network-related errors. Consequently, none of any corrective measures can be taken as the network does not identify any specific problems for the application. This eventually results on the normal network users being confused, annoyed and frustrated that they must then call to the professional support teams, network experts or administrators to help in solving their application problems.

Support professionals, network experts or administrators must then embark on one of two strategies below, depending on accessibility of the machine:

1. Spend time teaching the consumer command line tools or utilities such as PING, Telnet, dig and others. Needless to say, this is of course frequently performed via phone communication which we know having the high potential in leading to a more complications and misunderstandings. A 'faster' alternative way is that users are normally pointed out to some troubleshooting links, online manual or advices on the net. Normal network users may find some technical terms alien to them, so this can even worsen the situation if users could not properly conduct the course of actions as indicated due to misinterpretation.

   or

2. Request permission from the users to allow the support professionals to work at their machine while the users do something else during the troubleshooting process. This hands-on strategy is frequently used by onsite support. It may be an effective way but will definitely consume time and hassle to the support team as they need to shift places and move around from their work spaces to the user's site.

In both cases, fixing network related problems in a timely manner requires methodical troubleshooting techniques. The first critical step is gathering information about the user's machine. The second critical step and most important is identifying what works and what does not. Many of the tools and techniques used in this process only frustrate a normal network user who is not interested in the command line tools and interfaces that are necessary to solve the problem. On the other hand, support professionals generally prefer command line utilities for their speed capabilities.

Therefore, these issues have motivated us in coming up with a simple type of automated and extensible network fault diagnosing tool that is user-friendly to different type of users; be it the normal network users or the advanced users such as the network experts or professional support teams respectively.

1.2 OUR APPROACH

In response to the problem statements above, how does the support professionals or the network experts gather the required information unobtrusively and solve the actual problem in a timely manner, assuring a satisfaction on the user's side? And furthermore, in the case the normal network users are to troubleshoot their own network condition, how can they possibly start off in performing the task with a much comprehensible and user-friendly way?

The answer lies in the emergence of our tool that is purposely designed for flexibility, simplicity and automation. **Automated Network Fault Inference Tool** or shortly pronounced as **ANFIT** is a network diagnostic system that operates on command line, which is an effective and practical for both the normal network users as well as the advanced users. Regardless of which environment the tool run, the normal or advanced, both the users and administrators will find useful information or the immediate resolution to a problem. This tool help eliminate the necessity for these both target users to ever have to use a command line utility on their own as all have been automated, making the troubleshooting experience easier for everyone.

At this particular moment, ANFIT is designed and developed in such a way that it will only cater for the Web Service application which obviously means it could only detect the network problems in respect to the web utilization. In order to come out with this solution, the group embarks on a course of researching and studying on the normal network behaviour on how different things are working together in order to obtain a web page. We then gather and understand how the existing technical implementations, algorithms and protocols that are associated to the network actually operate. Subsequently, we then determine a set of possible failure cases that may occur in the network followed by figuring out the set of diagnostic tests that can be executed in order to reveal the failures. Some of the related and appropriate protocols, tools and technologies are then adopted and customized to our project needs. All these initial preparations then encourage us in a great deal of coming out with the inference table, which is the heart of the project. After all are set, we then automate those set of different diagnostic tests into code where we then sum up everything with the displaying out of different error messages to the different level of users.

1.3 ANFIT OVERVIEW

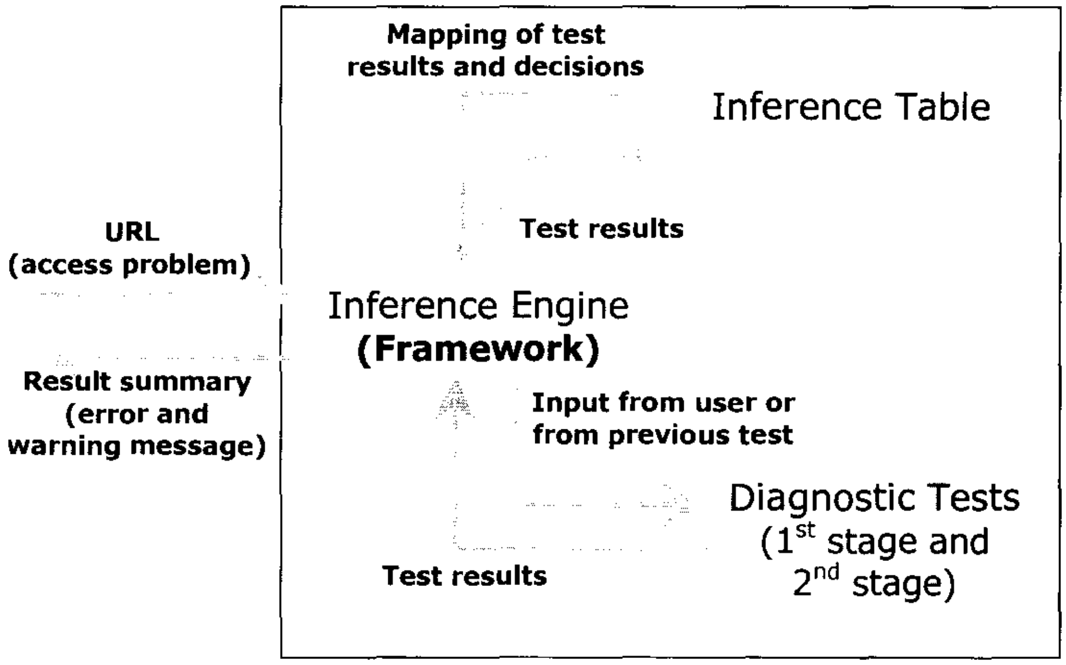
Mapping of test results and decisions
Inference Table
URL (access problem)
Test results
Inference Engine (Framework)
Result summary (error and warning message)
Input from user or from previous test
Diagnostic Tests (1st stage and 2nd stage)
Test results

Figure 1.2: ANFIT overview diagram

As depicted on the diagram above, users will need to enter the URL of the web page that they are having problem with in order to start off with ANFIT. Prior to that, they are required to provide their user level, either a normal Internet user or the advanced user in order for ANFIT to supply the appropriate error messages to them accordingly at the end of the session. This pertinent information will then be passed to Framework which is the heart of ANFIT as here lies the inference engine. The mechanism that involved is that all the relevant information to be processed in this inference engine are based on the inference table that we outlined during the designing stage. This table which comprises a set of diagnostic tests, problem layers and the mappings of diagnostic results and decisions will be explained more on the later section. Besides being the central repository of all the results obtained from executing the diagnostic tests, Framework also functions as the inference engine that decides whether or not to launch the further detailed diagnostic test for the particular problematic layers. The results of these thorough diagnostic tests will then again be sent to Framework as to allow the final mapping of error codes to the corresponding error messages. Eventually, these error messages or warning messages (where necessary) generated by Framework will be displayed to the users as based on their user level which will inform them the cause of failure to obtain a web page that has been detected by ANFIT.

1.4 GOALS

The group has outlines a few goals that motivate us in a great deal upon conducting the project:

- To assist users in detecting problems why they cannot access certain web sites by diagnosing the network and display out the error messages.
- To provide users with simple and practical suggestions where necessary.
- To integrate isolated yet related existing probing tools/applications into one unify Web Service tool.
- To produce a tool that is understandable and workable to users with varying degrees of skills.
- To enhance knowledge gain by comprehending profoundly on how the normal network behavior actually works in term of Web Service application.
- To initiate a tool that can serve as a huge community supportive due to the extensible framework of the system.

1.5 OBJECTIVES GUIDED BY KEY PERFORMANCE INDICATORS (KPI)

The objectives of ANFIT have been associated with Key Performance Indicators (KPI), in order to ensure the objectives outlined are measurable and achievable. KPI are quantifiable measurements that define target performance level of ANFIT and reflect the critical success factors of ANFIT. Enumerated below are the performance metrics chosen and the corresponding objectives, measures and targets established so as to assess ANFIT.

1.5.1 TIME

Objective:
- to prompt fast responses

KPI (measures):
- time taken to diagnose with test cases and prompt out the error messages

Target:
- at most 2 minutes for every case + keep user informed every 20 seconds

1.5.2 RESPONSE

Objective:
- to return understandable and appropriate responses that are user- friendly

KPI (measures):
- usability test

Target:

- at least 90% of target users find tool beneficial

1.5.3 COVERAGE

Objective:

- to be able to detect enough general network failures for the tool to be useful

KPI (measures):

- successful test cases proved by error messages as output

Target:

- at least 20 failure cases that we pre-defined

1.5.4 ACCURACY

Objective:

- to be able to produce responses similar to those produced by human expert when faced with same situation

KPI (measures):

- the results generated by the system are not conflicted with the results from human expert such as network administrator

Target:

- at least 90% similar to human expert responses

Chapter 2

BACKGROUND

To be able to diagnose network faults, some previous knowledge of normal network scenarios should be present. With this knowledge it is then possible, to understand certain failure situations. This section aims to illustrate some research findings about normal and abnormal (failure) scenarios related to the web services, which are suggested to be supported by ANFIT.

2.1 NORMAL NETWORK SCENARIO

In the case of normal scenarios, it is necessary to analyze basic protocols and what kinds of protocols are necessary for supporting specific web services. For the purpose of adding clarity, some examples are given below. In order to being able to support web services, it is assumed say the host running applications like Internet Explorer should not encounter any types of network problem. In addition, the network environment through which information is delivered is also suggested not to come across any problems. Lastly, the web server, which generates and provides particular information shouldn't have any problems either. The user's site can be classified as the local host, the network environment is identified as the network whilst the web server can be categorised as the remote server. Each category is subject to utilising specific protocols, which are necessary to provide the web service. Local host can either utilise DHCP or Configuration. The network environment itself employs a wide variety of protocols such as IP, DNS. The TCP and HTTP protocols are usually utilised by remote servers.

2.1.1 CABLE / HARDWARE

Apart from the network environment itself, the networking equipment such as server, network interface cards (NIC) and the wiring also play a vital role in accessing the net.

2.1.2 DEFAULT GATEWAY

In order to access the remote web servers, we need to configure our TCP/IP connection which we must specify a default gateway to access the remote networks. This information is used to tell your computer where to send packets when they are destined for or from an Internet address. Default gateway is the intermediate network device that is attached to the local network (can be a computer or router in the same subnet as the sending host) that has knowledge of the network IDs of the other networks in the Internet, so it can relay the packets to other gateways until they are delivered to the one connected to the specified destination. The default gateway address specifies the IP address of the host on the local subnet that provides the physical connection and forward traffic to remote networks, and is used by default when TCP/IP needs to communicate with hosts on other subnets.

2.1.3 LOCAL HOST CONFIGURATION

The interface configuration settings for a host require an IP address, subnet mask and default gateway. IP Address will act as unique identifier of the host on the network, subnet mask is used to determine whether a host is on the local subnet or on a remote network and default gateway is the router specified on the host, which links the host to other subnets.

2.1.4 DHCP

The Dynamic Host Configuration Protocol (DHCP) is a mechanism for automatically distributing essential network configuration data to PCs connected to large networks. The settings that are automatically distributed are typically IP address, subnet mask, Default Gateway, DNS server IP address(es), DHCP server IP address. DHCP supports mechanisms through which clients can be assigned a network address for a finite lease (known as *DHCP lease*), allowing for serial reassignment of network addresses to different clients. Once the lease time has expired, the configuration might stop working and it is the responsibility of the PC's operating system to *renew the lease* periodically.

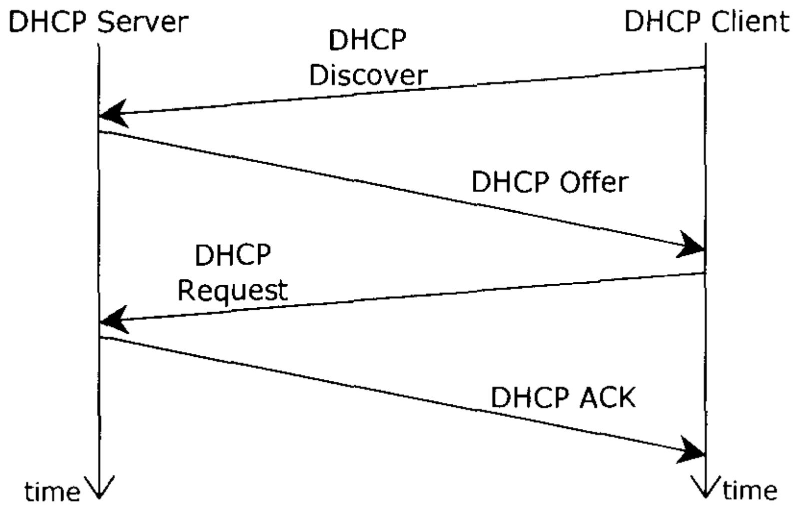

Figure 2.1: DHCP client - server interaction

As depicted in Figure 2.1, the DHCP process begins with *DHCP client* broadcasts a *DHCP Discover* message on its local physical subnet. Each DHCP Server may respond with a *DHCP Offer* message that includes an available network address and other configuration parameters. The message can either be unicast to DHCP Client or broadcast on the client's subnet. The DHCP Client then broadcasts a *DHCP Request* message to the server from which to request configuration parameters. The selected DHCP server will receive the message and replies with *DHCP Ack*, which contains the configuration parameters for the requested clients. Those servers that are not selected will use the *DHCP Request* message as a notification that the client declined the server's offer.

2.1.5 IP

Internet Protocol (IP) is the basic network transmission protocol of the Internet. This particular protocol operates on the network layer of the OSI model and is the primary network layer protocol in the Internet Protocol suite. It provides an addressing scheme and some control information, which is necessary to allow packets to be routed across a network. In addition, IP also specifies the format of packets. Combined with TCP, it provides the functionality of establishing a virtual connection between a destination node and a source node. The role of IP can be compared to the postal system; a parcel can be addressed and dropped into the system for delivery. In other words, IP is responsible for ensuring the packets are sent to the right destination.

2.1.6 TCP

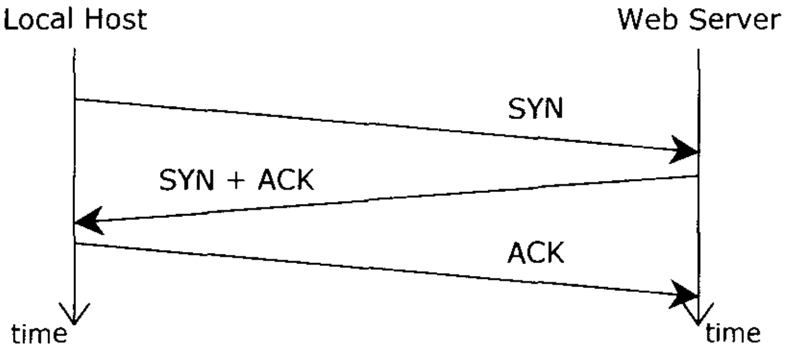

Figure 2.2: 3-way handshake - Connection establishment procedures

TCP is one of the main protocols in TCP/IP networks. TCP enables two hosts to establish a connection and exchange streams of data. TCP guarantees delivery of data and also ensures that packets will be delivered in the same order in which they were sent. As it is illustrated in the diagram above (Figure 2.2), a three-way handshake must first be established between the local host and the web server before application data can be transmitted. During the three-way handshake the exchange of parameters such as packet size and timeout values are accomplished before the exchange of application data can progress.

2.1.7 DNS

Domain name is the unique name that identifies an Internet site Domain Names always has 2 or more parts, separated by dots. The part on the left is the most specific, and the part on the right is the most general. A given machine may have more than one Domain Name but a given Domain Name points to only one machine. Domain Name System (DNS) is the method by which domain names are converted into the equivalent numeric IP address or vice versa. To the user and application process this translation is a service provided either by the local host or from a remote host (name server) via the Internet.

Based on the requests from programs or other DNS Server(s), every DNS Server can do any of the followings:

1. Answer the request with an IP address or any answers pertaining to the query if it has the answer.

2. Contact another DNS server and try to find the answer to the query.

3. Refer the requester to other authoritative DNS servers because it does not have the answer as well as not the authoritative name server for the requested domain.

4. Return error message if the requested domain name is invalid or does not exist or if it is experiencing internal failure.

5. If it is an authoritative name server for the zone and the requested domain does not exist and no data of the requested type exists, returns SOA record so that the response may be cached.

In the example shown in Figure 2.3, a client on Aldgate.cs.ucl.ac.uk needs the IP address of google.com. The following events take place:

1. The client contacts Local DNS Server (LDS) with a recursive query for google.com. The server must now return either the answer or an error message.

2. LDS checks its cache and zones for the answer, but does not find it, so it contacts a server authoritative for the Internet (that is, a *root server*) with an iterative query for google.com.

3. The server at the root of the Internet does not know the answer, so it responds with a referral to a server authoritative for the .com domain.

4. LDS contacts a server authoritative for the .com domain with an iterative query for google.com

5. The server authoritative for the .com domain does not know the exact answer, so it responds with a referral to server(s) authoritative for the google.com domain.

6. LDS contacts either one of the Authoritative DNS Servers (ADS) for the google.com domain with an iterative query for google.com

7. The ADS for the google.com domain does know the answer. It responds with the requested IP address.

8. LDS responds to the client query with the IP address for google.com

9. Client browser contacts google.com web server using IP address given by LDS.

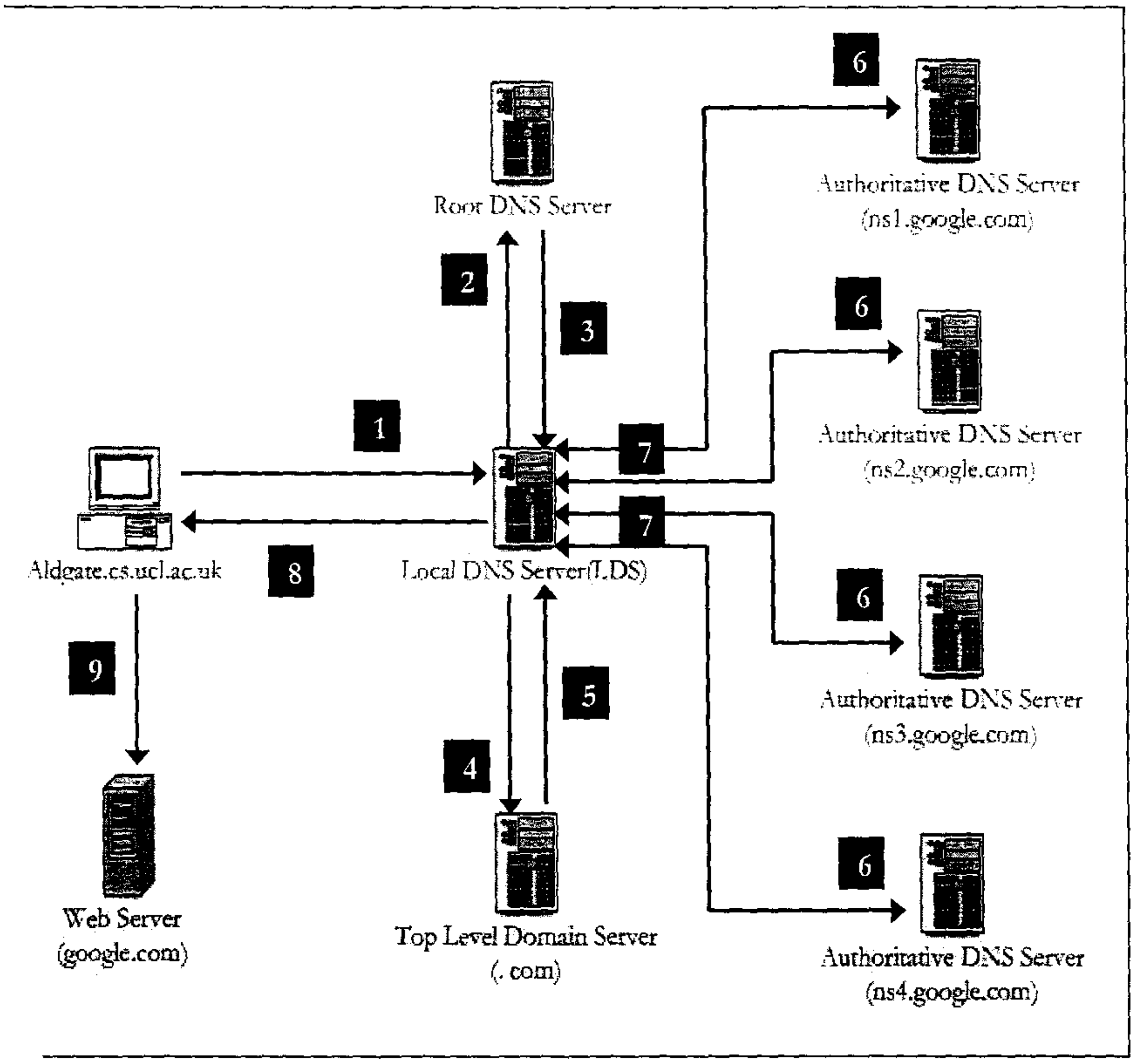
Root DNS Server
Authoritative DNS Server (ns1.google.com)
Authoritative DNS Server (ns2.google.com)
Authoritative DNS Server (ns3.google.com)
Authoritative DNS Server (ns4.google.com)
Aldgate.cs.ucl.ac.uk
Local DNS Server(LDS)
Web Server (google.com)
Top Level Domain Server (.com)
1
2
3
4
5
6
7
8
9

Figure 2.3: DNS Lookup with iterative and recursive queries

2.1.8 HTTP OR HTTPS

HTTP is the web's application layer protocol that realise the operation of obtaining web page from the remote web servers. It is implemented in both a client and a server program. These both programs will execute on different end systems, talk to each other by exchanging HTTP messages. A web browser (that act as user agent on the client host) such as the largely used Internet Explorer, Mozilla or Netscape Communicator employs this protocol to request a web page from a web server as illustrated in figure above as example. The web server, with each addressable by a URL, will implement the server side of HTTP and will then response with either returning the document requested or an error message indicating it fails to fulfil the request for certain reason as based on the standard error status codes as have been outlined in RFC 2616. The HTTP Request and Response procedures are shown pictorially in Figure 2.4.

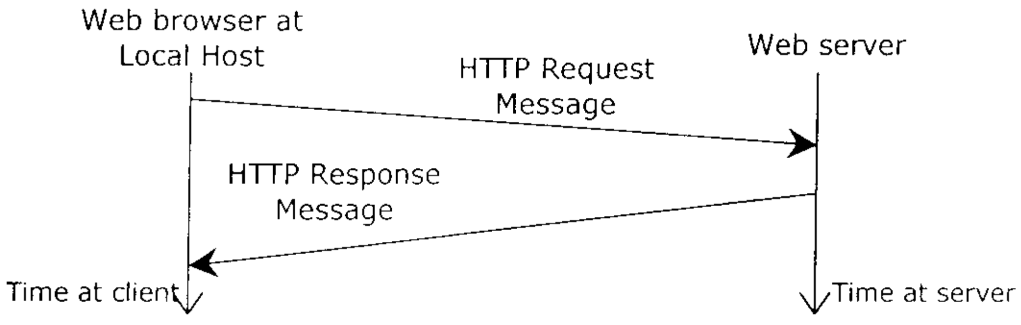

Figure 2.4: HTTP Request and Response procedures

HTTPS is an encrypted form of HTTP used for sending sensitive data like credit card details between the browser and the web server. This is also sometimes called Secure HTTP or SSL (Secure Sockets Layer). This protocol which carried over the bi-directional, encrypted, 'pipe' provided by the SSL protocol, incorporates the use of digital certificates and signatures so that the client can automatically authenticate the server, and establish a highly secure two-way 'pipe' without the user needing to manage cryptographic keys.

Other common protocols (which are not within the boundary of this project) that work in similar ways are FTP and Gopher, but there are also protocols that work in completely different ways. It is worth noting that HTTP and HTTPS only defines what the browser and web server say to each other, not how they communicate. The actual work of moving bits and bytes back and forth across the network is done by TCP and IP.

2.1.9 URL

The first thing the browser has to perform in order to accomplish the process of obtaining any web page is to look at the URL of the new document to find out how to get hold of the new document. Most URLs have this basic form:

"protocol://server:[port]/request-URI"

The protocol part describes how to tell the server which document that you want and how to retrieve it. The server part tells the browser which server to contact. If the server port is empty or not given (this is optional), port 80 is assumed for HTTP and port 443 for HTTPS and the request-URI (the term used by the HTTP standard) is the name used by the web server to identify the document which comprises of the directory and file.

2.2 POSSIBLE CAUSE OF FAILURES

To troubleshoot any network fault, apart from knowing how network operates under normal circumstances, it is also essential to be familiar with the possible cause of failure at every part of

2.2.1 CABLE / HARDWARE

Sometimes, users cannot access certain web pages due to a number of several issues. Some of these issues may not necessarily refer back to network problems but could also be faults of technical nature on the user's side. Most common problems are suggested to be machine misconfigurations or refer to malfunctioning devices. A good example is a cable problem where the cable was accidentally unplugged from the user's machine or a faulty cable. Another problem could be caused by human errors such as faulty or improperly wiring. Issues relating to the networking hardware itself could be faults with the port.

2.2.2 DEFAULT GATEWAY

One of the obvious reasons that a packet cannot be relayed to the remote hosts that reside outside the subnet is that the wrongly configured IP Address of the default gateway. Since default gateway is in the same subnet, it communicates with other host in the subnet via ARP. In the case that any host is sending an ARP request message, the default gateway will not be able to respond with a corresponding ARP reply message if there is something wrong with the default gateway configuration. Thus, we will be able to know that our local connectivity itself is experiencing problem that may be a reason of why we cannot get access to certain web page of the remote web servers.

2.2.3 LOCAL HOST CONFIGURATION

Network problems can be attributable to incorrect configurations of the three main interface settings. A host with incorrect IP addresses will not be able to communicate with some or all computers on their local networks but can talk to hosts on remote networks. A bad subnet mask can be determined when the host can reach other hosts in its local subnet and remote hosts on distant networks, but it cannot reach hosts on other local subnets. A computer configured with an incorrect default gateway will be able to communicate with hosts on its own network segment, but will fail to communicate with hosts on some or all remote networks

2.2.4 DHCP

The DHCP server automatically allocates IP addresses and related TCP/IP configuration settings to DHCP-enabled clients on the network. Therefore, a DHCP failure is one of the factors, that contributes to user unable to access the network and its resources. The possible problems with DHCP are inability of the DHCP client to acquire or renew a lease, disconnection every few hours, DHCP server overload or down, DHCP server is out of IP addresses, wrong configurations of IP Address or Default Gateway at DHCP Server.

2.2.5 IP

When IP packets are routed across a network, there is the potential for problem at every hop between the source and the destination. Potential problems could be corrupted packets, wrong IP addresses for the intended destinations. In addition, a router may not have a route to the source or destination, an MTU mismatch, software problem might have occurred on a router and also a resource problem on one router might be prohibiting proper router operation. This could possibly be caused by lack of memory, lack of buffers, buffer overflows or the lack of processing power of the CPU.

2.2.6 TCP

Potential problems with TCP for example are the non availability of HTTP processes which are usually necessary to handle requests from the sender. Also, certain issues such as the web server being busy and/ or refusing to establish TCP connections are further potential problems with TCP. Finally, the link between a router and a web server could have been disconnected, and a web server may not reply due to having crashed or having encountered a power shortage. A packet filter/firewall issue might have arisen for the specific protocol, data connection, or return traffic.

2.2.7 DNS

DNS look up failure is one of the prevalent causes for Internet connection failures. DNS look up failure can be determined through the information returned by the DNS servers. The most common causes for the failure are DNS server provides stale information, returns inconsistent answers or performs incorrect name resolution. Other problem that may exist are DNS server fails to respond to client which are either due to no data exist pertaining to query or internal server failure.

2.2.8 HTTP OR HTTPS

Another factor that can cause a reason why a user cannot get the page they requested is due to the fact that both the client host and web server may have failed to complete the HTTP request-response connection. As we know, HTTP protocol uses TCP as their underlying transport protocol thus; the client first initiates a TCP connection with the server. Once connection is established, the browser's (client) process and the server's process will access the TCP through their socket interfaces. Although the transport layer connection is successfully established between both parties, the client and server may fail in term of sending and receiving HTTP Request Message or Response Message into its socket interface. In this case, the server or client may have their own complications or caused by the instability of the network that hinders them of doing so.

2.2.9 URL

An important concept here is that to the browser, the server works as a black box which means, the browser requests a specific document and the document is either returned or an error message is returned. How the server produces the document remains unknown to the browser. The status codes, representing either success or error conditions, are all three-digit numbers that are grouped by the first digit into 5 groups (refer to Table 2.1). Server can return any reason phrase they wish that is suitable to each status code. [Please refer to RFC 2616 for a more detailed explanation of each code]. As based on the error codes, there are numerable occasions that involved both the server and client side that result in a failure of obtaining a web page. Errors could be as trivial as punching in a wrong URL at the client side (browser) up to the servers themselves are having certain problems.

Table 2.1: HTTP Error Codes

Error Code	Type of Error Code
1xx	Informational - The request received, showing a continuing process. Rarely used and generally only written to server logs except under experimental conditions.
2xx	Successful - The action successfully received, understood and accepted. Only 200 frequently used and others are generally only written to server logs.
3xx	Redirection - Just a warning as the request may still be satisfiable, provided that further action must be taken.
4xx	Client Error - The request was invalid in some way that it cannot be fulfilled by the server (eg. bad syntax, misspelled)
5xx	Server Error - The server could not fulfil the apparently valid request (eg. Server busy, overloading, service unavailable.

Note: Please refer Appendix C for a list of error codes in each series

2.3 COMPARISON WITH OTHER WORKS

As a normal procedure in every system development, a thorough research in that particular area needs to be conducted beforehand. This section highlights a collective comparisons study that we have embarked on before proceeding with the design and implementation of ANFIT.

2.3.1 COMPARISON WITH HUMAN EXPERTS

Automated diagnostic tool is likely to be as accurate as human diagnosis but much faster because it is capable of rapid identification, analysis of conditions, and diagnosis in real time. Besides, human experts are not systems of rules; they are libraries of experiences which make them sometimes unable to articulate reasoning process sufficiently and precisely. Moreover, the inferences given may vary from one expert to another, though given the same problem scenario. Furthermore, human experts are expensive and short in supply, which make automated tool, a more preferred choice. Human experts are also affected by fatigue, emotional states, forgetfulness and habituation that make them prone to making errors in judgment.

2.3.2 COMPARISON WITH OTHER DIAGNOSTIC APPROACHES

2.3.2.1 MODEL-BASED [de Barros et al.]

The Model-Based approach is based on a model of the structure and behaviour of the device that the system is designed to simulate. In order to illustrate the concept of this approach, an example by de Barros et al is given below. In the literature de Barros and colleagues performed the communication fault diagnosis in the network domain using a Model–Based approach. It is a diagnostic reasoning approach based on a complete representation of the logical and physical levels of the network. A Knowledge Base System is constructed through the specification of two basic structures: Domain Models and Problem Solving Methods. The matching of components and variables among different views of the network domain is achieved through uniform representation of the network models. This diagnostic approach provides a function for exchanging information between the sub-tasks of a complex system. This method used models that represent many aspects of network. Amongst them are configuration models, performance models and others. Problem Solving Methods decomposes the diagnosis task into three sub-tasks which are Symptom Detection, Hypothesis Generation and Hypothesis Discrimination. In symptom detection, the symptoms starting from observations done on the device is detected. A symptom corresponds to some abnormal observation. In hypothesis generation, the possible causes (diagnostic) that explain the presence of the symptoms based on a set of symptoms is determined by this sub-task. In hypothesis discrimination, the set of hypothesis generated by the previous task in order to determine the most probable one that is the best explanation for the observations is analyzed.

2.3.2.2 RULE-BASED [Hargis]

In Rule-Based approach, the knowledge base consists of a set of rules in the form of IF-THEN statements: IF some condition THEN some action. It starts with a rule-base, which contains all of the appropriate knowledge encoded into If-Then rules, and a working memory, which may or may not initially contain any data, assertions or initially known information. The system examines all the rule conditions (IF) and determines a subset, the conflict set, of the rules whose conditions are satisfied based on the working memory. Of this conflict set, one of those rules is triggered (fired). Which one is chosen is based on a conflict resolution strategy. When the rule is fired, any actions specified in its THEN clause are carried out. These actions can modify the working memory, the rule-base itself, or do just about anything else the system programmer decides to include. This loop of firing rules and performing actions continues until one of two conditions is met: there are no more rules whose conditions are satisfied or a rule is fired whose action specifies the program should terminate.

2.3.2.3 CASE-BASED [Aamodt, A., and Plaza, E.]

In Case-Based approach, a new problem is solved by finding a similar past case, and applying it to new problem situations. It is able to utilize the specific knowledge of previously experienced, concrete problem situations (cases). It also is an approach to incremental, sustained learning, since a new experience is retained each time a problem has been solved, making it immediately available for future problems. A case usually denotes a problem situation. A previously experienced situation, which has been captured and learned in a way that it can be reused in the solving of future problems, is referred to as a past case, previous case, stored case, or retained case. Correspondingly, a new case or unsolved case is the description of a new problem to be solved.

ANFIT is based on this Case-Based Approach but there is suggested to be some variation from this approach. Cases here refer to the Problem Categories that are represented in the row of the Inference Table and will be discussed in more detail in Chapter 4. The test cases that are represented in the column of the Inference Table are used to detect where the problems might occur. In the Case-Base approach the group has agreed on using some of the test cases are dependent on the outcome of the previous test results. Therefore the knowledge gained from previous tests is being used in some of the following tests.

2.3.3 COMPARISON WITH EXISTING TYPE OF NETWORK DIAGNOSTIC TOOLS

There are numerous kinds of network diagnostic tool that could become handy these days. Much of the state-of-the-art innovations are software suites that leverage intelligence engines, software that facilitates meaningful data analysis from various sources. These solutions are designed for adaptability, flexibility, automation and are distinguished by various features designed to appeal to specific categories of users. These network diagnostic tools, despite their diversity, are normally divided into two major categories based on their level of activity; the proactive diagnostic tools and the reactive ones [Reference: http://www.gcn.com/archives/gcn/1999/April5/29.htm]. As for the proactive, these tools provide most of their performance monitoring, diagnostics and reporting automatically according to hourly, daily, weekly and annual schedules. Most come with online, telephone or pager alarms in case of network faults. Nearly all provide graphical network maps and highly detailed reports in graphical as well as statistical formats. Most provide end-to-end network overviews, right down to users' desktops and application use. As for the reactive network diagnostic tools, no monitoring or timely reporting has been applied and problems are reacted to only after they occur. In reference to that, ANFIT is designed to suit the reactive response activity which will provide users with a kind of on-demand response in term of error messages as based on the problematic URL entered by user.

2.4 EXPERT SYSTEM

An expert system can best be defined as a computer program that emulates the problem solving process used by human experts. The system is built and embodied with the knowledge elicited from experts and captured from established references and specifications. For a network diagnostic tool, the notable use of expert system is in evaluating results obtained from running tests on the network. The results are then used in making inferences on what and where does the failure occurs. Usable solutions or practical suggestions are imparted to the user as the final conclusion. Another adopted expert system strategy is to estimate the level of certainty for any solution or suggestion given as an indicator for the degree of accuracy of the answer. This is because the user needs to be convinced that the conclusion given is substantially correct and the solution proposed is appropriate to their particular case.

Chapter 3

REQUIREMENT

Diagnosing why a network service does not work is a difficult and time consuming task, even for someone who knows what they are doing. "Is the net down?" is a common query. The cause can range from an unplugged network cable on the client machine, to the server being down, with many other possibilities in between, such as DNS misconfiguration, routing failure, or link failure. The challenge is to design a tool that attempts to self-calibrate what normal network behaviour is. Then, when something is not working, to be able to attempt to answer questions such as "Why can't I get to the BBC's web site?"

3.1 SCOPE

This section of the report aims to illustrate the scope of the project. Due to the complexity of the project and the broad range of application services offered by the Internet, the group had to agree on the scope of the project as otherwise it would have not been possible to undertake the project within the given time constraints. This project only focuses on the Web Service and no other types of Internet applications such as File Transfer Protocol (FTP) or email. The final output, which is the error messages that will be displayed by ANFIT will only provide a result illustrating where the network has gone wrong based on the Problem Categories that are represented in the row of the Inference Table. The Problem Categories are Local Host Problem, Network Problem, DNS Problem, Reachability Problem and Remote Server Problem. This will be discussed in more detail in Chapter 4. ANFIT also gives simple suggestions in command line form, not detailed solutions for troubleshooting.

3.2 INITIAL ASSUMPTION

Prior to starting this project, the group had to come up with two initial assumptions for the users to use ANFIT. The first assumption is that users are just using Web Service applications and not FTP, e-mail or other Internet applications. The second assumption states that in case of more complex errors occurring, ANFIT will inform normal Internet users to ask the network administrator about the specified issue who in turn will provide a more specific solutions for troubleshooting (higher level instructions) During this interval, an intersession message will be displayed to user. This is to inform the user that a further diagnosing stage is taking place and in order to retain the users' interest. The group is aware that in a real network environment, a host can has multiple interfaces. However, for the simplicity in the design and development of this project, it has been decided to assume that each local host will only have one interface.

3.3 TYPES OF USER

There are two types of ANFIT users. They are advanced network user and normal Internet users. The group has decided to differentiate them in the hope that ANFIT will benefit each of them accordingly. As for the advanced network users, we hope that ANFIT will be able to provide them with an automated tool that can detect actual problems with web service without manual (human) interference. Apart from that, we also hope that ANFIT will become a powerful command line tool that can provide quick results of the problems. In return, this can assist them in solving or troubleshooting network problems (precisely web service) efficiently. Nonetheless, we hope that by using ANFIT, it will save them the hassle to entertain trivia tasks. As for the normal Internet users, we aim to provide them with a friendly diagnostic tool that can impart the results in an understandable, low-level and appropriate language. We also hope that ANFIT will provide them with a tool that can determine the actual problems that occur.

- **Normal Internet User:**

Those who use Internet regularly with little knowledge or experience in networking despites their technical background.
Example: computer science student and non-computer science student

- **Advanced Network User:**

Those who possess vast knowledge and experiences in the area of computer networking.
Example: network administrator, support teams and network expert

3.4 FAILURE SCENARIOS

This section contains descriptions of various failure scenarios that serve as guidelines for the group to construct the Inference Table as well to clarify our thoughts prior to designing ANFIT. Enumerated below are the failure scenarios that we have collated and summarized as based on the common possible network faults in the hope that these will reveal themselves accordingly once ANFIT is executed.

Reachability and Remote Web Server Problem

Failure Scenario 1:

- Web page requested not in Web server.
- So Web server will return status code 404[Not Found] since it cannot find anything that matching the request.

Failure Scenario 2:

- Web page exists but service is unavailable due to temporary overloading or maintenance.
- HTTP Request message is OK
- Server will send HTTP Response = 503 to indicate service is unavailable.

Failure Scenario 3:

- IP level connectivity is already established and TCP connection is already set up.
- User sends GET message but Web server does not respond at all.
- Timeout is needed in order to re-attempt sending request.

Failure Scenario 4:
- Web page exists but has been redirected (either temporary or permanent) to a different URL.

Failure Scenario 5:
- IP level connectivity is already established.
- User sends SYN to Web server but Web server refuses due to no HTTP process to handle request.
- Web server sends ICMP message to user as 'Reset' which means nobody is listening

Failure Scenario 6:
- IP level connectivity is already established.
- User sends SYN to Web server but Web server refuses the TCP connection because it is busy or overload.
- Server will not respond at all which indicates that it is busy.

Failure Scenario 7:
- User cannot send SYN to Web server in order to initiate TCP connection due to broken link between router and Web server.

Failure Scenario 8:
- Last hop router wants to forward TCP SYN to the Web server but the server crashes or powers off.
- If ithe router has no cache of the destination host (Web server) in the ARP table, it will send ICMP message to the user as "Destination Host Unreachable".

Failure Scenario 9:
- Intermediate routers want to forward TCP SYN to the Web server but the server crashes or powers off.
- If the router has no cache of the destination host (Web server) in the ARP table and it cannot find the network, it will send ICMP message to the user as "Network Unreachable".

Failure Scenario 10:
- Intermediate router wants to send TCP SYN to the Web server but the server crashes or powers off.
- If intermediate router has cached the address mapping of the Web server in the ARP table, it will send out the packet but eventually the packet will be silently discarded because nobody will claim the packet.

Failure Scenario 11:

- Sends request to DHCP Server but it does not respond at all.
- Maybe due to crashes, power off or internal server failure, which makes it unable to process the DHCP request.

Local Connectivity Problem

Failure Scenario 12:

- Faulty, unplugged cable of the local host
- Malfunctioning of devices

Failure Scenario 13:

- Local host misconfigurations

Failure Scenario 14:

- IP conflict due to same IP address map to different MAC addresses.

DNS Server Unreachable

Failure Scenario 15:

- Cannot reach any Local DNS Server(LDS)
- Maybe due to link between LDS and router is broken or LDS crashes or powers off.

Failure Scenario 16:

- Cannot reach any of the 13 Root DNS Servers(RDS)
- Maybe due to link between RDS and router is broken or RDS crashes or powers off.

Failure Scenario 17:

- Cannot reach Top Level Domain(TLD) server
- Maybe due to link between TLD server and router is broken or TLD crashes or powers off.

Failure Scenario 18:

- Cannot reach any Authoritative DNS Server(ADS)
- Maybe due to link between ADS and router is broken or ADS crashes or powers off.

Internal Server Failures /SERVFAIL

Failure Scenario 19:

- Connection to RDS is ok.
- Internal LDS Failure
- LDS was unable to process the query due to a problem with the name server (ex: OS problem, etc)
- LDS sends response code = 2 [SERVFAIL] to host

Failure Scenario 20:

- Connection to RDS is ok.
- Internal ADS Failure
- AS was unable to process the query due to a problem with the name server (ex: OS problem, etc)
- AS sends response code = 2[SERVFAIL] to LDS

Operation Timed Out

Failure Scenario 21:

- IP level connectivity between user and LDS is OK but LDS does not give any respond at all until timeout expires.
- Failure may due to LDS unable to handle the query or unable to generate the reply.

Failure Scenario 22:

- ADS does not give any respond until timeout expires.
- Failure may due to ADS unable to handle the query or unable to generate the reply.

Domain does not exist/NXDOMAIN

Failure Scenario 23:

- If input from user = g.com and query made to Local DNS Server(LDS)
- LDS will then send response code = 3[Name Error] or NXDOMAIN and SOA record of the .com zone to requester to indicate the domain does not exist.

Failure Scenario 24:

- If input from user = g.com and query made from Level Above
- Top Level Domain (TLD) Server will send response code = 3[Name Error] or NXDOMAIN and SOA record of the .com zone to requester to indicate the domain does not exist.

Failure Scenario 25:

- If input from user = 1.google.com and query Local DNS Server(LDS)
- LDS will then send return code = 3[Name Error] or NXDOMAIN and SOA record of the google.com zone to requester to indicate the domain does not exist.

Failure Scenario 26:

- If input from user = 1.google.com and query from Level Above
- TLD server returns referrals on google.com's ADS
- ADS send return code = 3[Name Error] or NXDOMAIN and SOA record of the google.com zone to the requester to indicate the domain name requested does not exist.

BAD CACHE

Failure Scenario 27:

- Bad cache – mapping given by LDS is not the latest one and conflicted with the one stored in ADS.
- IP List of LDS is not subset of ADS

INCONSISTENT DATA

Failure Scenario 28:

- Different TLD name servers return different NS records

Failure Scenario 29:

- Different Authoritative DNS Servers return different NS records

NO DATA

Failure Scenario 30:

- The query is valid and the status of the query is NOERROR but no answer given by the Top Level Domain(TLD) Server

Failure Scenario 31:

- The query is valid and the status of the query is NOERROR but no answer given by the Authoritative DNS Server (ADS).
- ADS should have returned SOA record if no data of the requested type exists.

Failure Scenario 32:

- Query made to the zone's primary name server, using zone name as the query, asking for NS record should returned Authoritative DNS Server (ADS) for that zone.
- If the zone fails to return ADS list, this means that the zone's name server is wrongly configured.

3.5 TOOLS AND TECHNOLOGY

This section comprises of the tools and technologies deployed in developing ANFIT.

3.5.1 JAVA

JAVA is an object-oriented programming language incorporated to construct the framework of ANFIT

3.5.2 XML

XML is a format used to store data pertaining to inference table, execution command and arguments for each diagnostic test.

3.5.3 PYTHON

Python is a scripting language utilized to develop all the diagnostic tests.

3.5.4 IFCONFIG

ifconfig (Interface Configuration) is used to assign an address to a network interface or to configure network interface. It is a handy command for troubleshooting purposes as it can display and quickly reveal the status of all configured interfaces.

3.5.5 ARP

arp (Address Resolution Protocol) command is used to analyze problems with IP to Ethernet address translation. The command has three practical options for troubleshooting and they are for displaying all ARP entries in the table, deleting an ARP entry and adding a new entry.

3.5.6 NETSTAT

netstat provides a variety of information. It is a diagnostic command commonly used to display detailed statistics about each network interface, network sockets and the network routing table. This command is available only if the TCP/IP protocol has been installed

3.5.7 TCPDUMP

tcpdump is an utility that listens to and records received and transmitted packets on a network segment. This utility is particularly useful in troubleshooting and monitoring the network activity. Expression can be specified in the command as to help in narrowing the focus of the diagnostic area as well as pinpointing the possible problems. It can be utilized in diagnosing TCP, DNS or other network problems.

3.5.8 PING

ping indicates whether a remote host can be reached. It also displays statistics about packet loss and delivery time. The ping tool uses the IP ICMP echo request and echo reply messages to test reachability to a remote system. In its simplest form, ping simply confirms that an IP packet is capable of getting to and getting back from a destination IP address.

3.5.9 DIG

dig (Domain Information Groper) is a flexible and comprehensive tool that sends DNS queries to DNS servers and display the replies. The replies consist of header fields, flags, question, answer, authority records and additional records sections and printed in a human-readable format. Summary of how long the exchange required is also printed. The replies given can be used as information to troubleshoot DNS problems.

3.5.10 TELNET

telnet is a TCP/IP terminal emulation protocol which enables remote login(s) - the ability to login to another computer (the remote computer) from your local computer. It is also a UNIX command which starts the telnet program on a UNIX computer. A telnet program is the software (based on the telnet protocol) which enables the user to login to a remote computer and use its resources. It does not utilise any security protocols such as secure sockets layer (SSL) and does therefore not provide any encryption.

3.5.11 WGET

wget is a network utility to retrieve files from the World Wide Web, using HTTP (Hyper Text Transfer Protocol) and FTP (File Transfer Protocol). It has been designed for robustness over slow or unstable network connections; that if a download fails due to a network problem, it will keep trying until the whole file has been retrieved

3.5.12 ECHO

echo is a utility that writes its arguments to the standard output. It can be invoked in order to repeat the previous command.

3.5.13 DUMMYNET (IPFW)

Dummynet is a flexible tool for testing networking protocols in the test environment of the project. It is implemented in FreeBSD but is easily portable to other protocol stacks. Dummynet is heavily based on the ipfw package for packet selection.
ipfw allows the selection of IP packets based on a combination of source and destination addresses and ports, protocol types (eg. UDP, TCP, ICMP, etc), interface and direction (incoming and outgoing)

Design Drawback:

- This strategy has appeared to be lacking of failure coverage and accuracy. Since this strategy requires the diagnostic method to stop at the faulty section, it may overlook errors that might occur at other sections of the network. Thus, the error message that will be imparted to the user may not be accurate and precise.

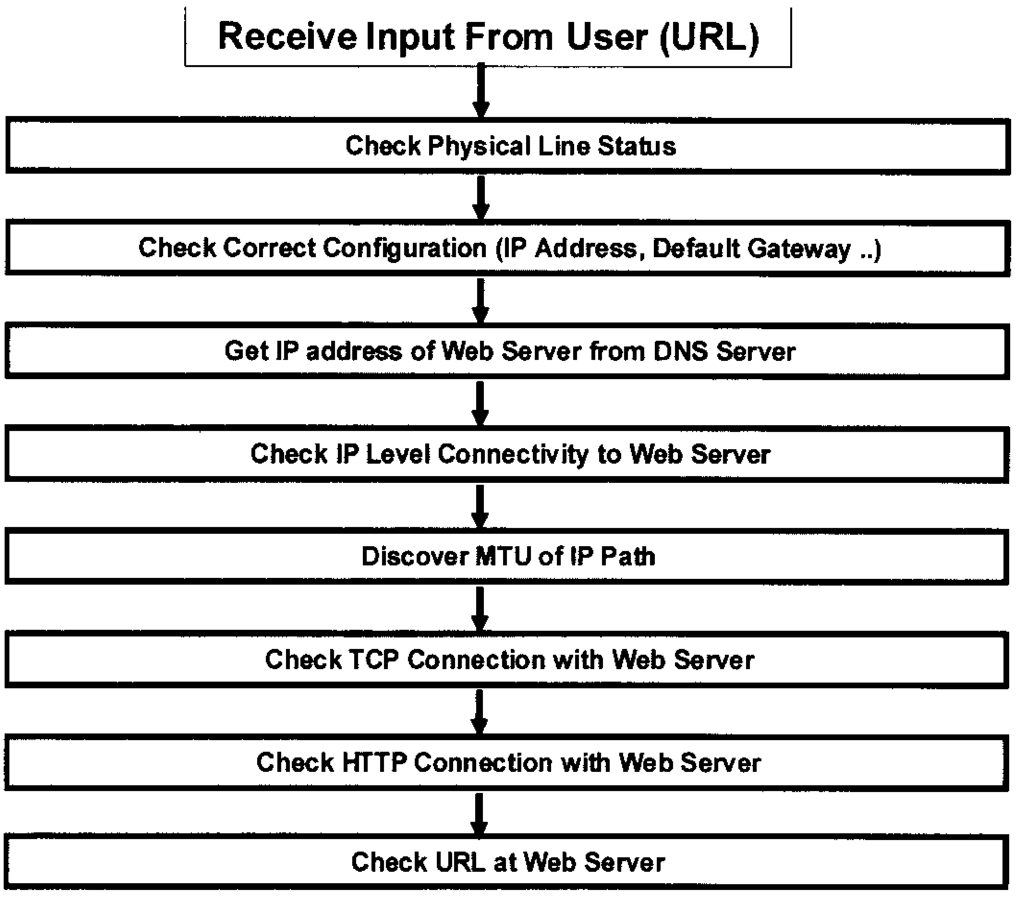
Receive Input From User (URL)
Check Physical Line Status
Check Correct Configuration (IP Address, Default Gateway ..)
Get IP address of Web Server from DNS Server
Check IP Level Connectivity to Web Server
Discover MTU of IP Path
Check TCP Connection with Web Server
Check HTTP Connection with Web Server
Check URL at Web Server

Figure 4.1: Diagnostic Steps

4.2.2 DIAGNOSTIC FLOWS

Design Strategy:

- This design is a slight improvement from the previous design as it groups the sections into different categories that perform different roles and functions. Each category is designed to cater for local connectivity, remote connectivity and performance issues respectively. As illustrated by the Figure XXX, it is assumed that the local connectivity is not the source of failure if the tool managed to obtain the IP(s) of the web server. Otherwise, the tool will carry out each diagnostic test that belongs to each section under that category (local connectivity). If the tool failed to get the web page, the cause of the failure may be attributable to the remote connectivity problem. As for that, the tool will troubleshoot the upper layers of the Internet. If no failures detected in both categories, then the tool will check the overall network performance. Error messages are shown to the user once the failure is detected.

Design Rationale:

- The motivation of this design is to improvise the previous design in terms of failure coverage. This is achieved by carrying out all the diagnostic tests under the category that is assumed to have failure. Due to that, this will maintain the fast performance aspect that we should gain from the previous method.

Design Drawback:

- Since the method stops once the failure is detected at certain category, hence, it will not execute further checking at other categories. Therefore, if more than one categories are having failures, this strategy will unfortunately fail to deduce an accurate decision that should be derived from the combination of all categories.

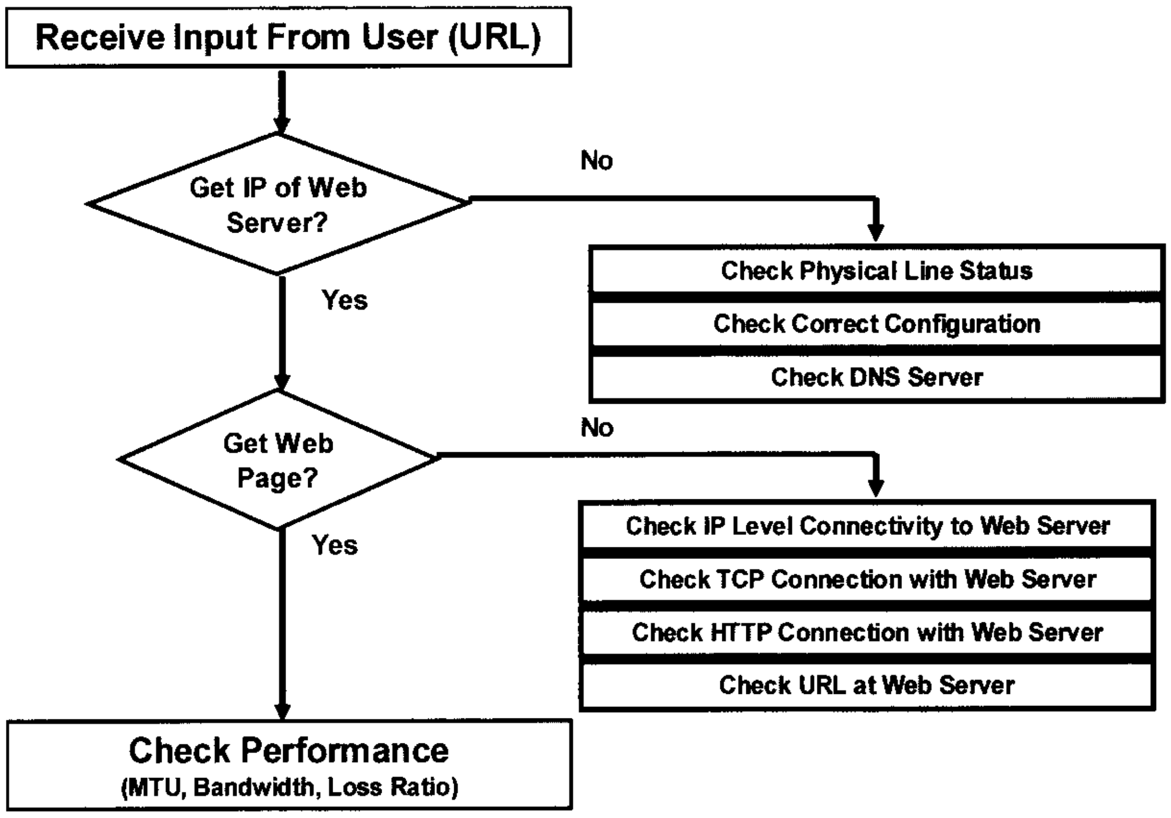
Receive Input From User (URL)
Get IP of Web Server?
No
Yes
Check Physical Line Status
Check Correct Configuration
Check DNS Server
Get Web Page?
No
Yes
Check IP Level Connectivity to Web Server
Check TCP Connection with Web Server
Check HTTP Connection with Web Server
Check URL at Web Server
Check Performance
(MTU, Bandwidth, Loss Ratio)

Figure 4.2: Diagnostic Flows

4.2.3 2-DIMENSIONAL INFERENCE TABLE

Design Strategy:

- This design is a major leap from the previous designs as we transform the algorithmic way of diagnosing into a tabular form. The table is structured in such a way that the diagnostic tests fill the column of the table and failure cases form the row. All the seven diagnostic tests are the same as in the previous designs but a new feature has been added in the strategy, which are the failure cases. These failure cases are derived from the failure scenarios as enumerated in Chapter 3 previously (Section 3.4). This is illustrated in Table 4.1. As in the previous, the tool still runs all the diagnostic tests sequentially and the failure will reveal itself based on the combination of all the test results. Eventually, the tool will conclude the specific failure as the cause of the problem in accessing the web page.

Design Rationale:

- This strategy is clearly better in terms of failure coverage, which is achieved through running all the diagnostic tests. This ensures a higher certainty level of the cause of failure since the tool will only infer the final decision after evaluating all the results from each diagnostic test.

Design Drawback:

- The downside of this strategy is that the approach in deciding the failure is too decisive. This means the tool can only impart the result within the scope of the failure scenarios' list. Although this method has a higher level of certainty than the previous designs, but it is still not sufficient as the error message is bound to be based by the specific failure case. Furthermore, there is lack of flexibility to the design as the result can only be due to one failure whereas in reality, the cause of failure can be rooted from many sources.

Table 4.1: 2-Dimensional Inference Table

Test Cases / Failure Cases	Line	Configu-ration	DNS	IP	TCP	HTTP	Web Page
Faulty/unplugged cable	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
DHCP server is not working	[illegible]	✗	[illegible]	[illegible]	[illegible]		✗
DNS Server is not working	[illegible]		✗	[illegible]	[illegible]	[illegible]	[illegible]
No IP path to Web Server			[illegible]	[illegible]		[illegible]	✗
Cannot open TCP connection with Web Server		[illegible]		[illegible]	[illegible]	[illegible]	✗
No response from HTTP server		[illegible]	[illegible]	[illegible]		[illegible]	[illegible]
Web page does not exist	[illegible]		[illegible]	[illegible]		[illegible]	[illegible]

Legend:

✗ **Fail**

✓ **Success**

4.2.4 REFINED INFERENCE TABLE (I)

Design Strategy:

- This method deploys the same strategy as the previous design but it is improvised for betterment in terms of diagnosing approach. The group has decided to embark on two stages tests approach. The first stage tests will determine where the network has gone wrong and the second stage tests decide what has caused it to fail. The test cases that form the column are still the same but with refined terms. This approach still runs the diagnostic tests in sequential manner but pre-conditions are added to each of the diagnostic test. This acts as indicators whether or not to execute that current diagnostic test. We made enhancement by merging the TCP and IP diagnostic tests into one test, Remote Server TCP/IP Level Connectivity Test but producing two different results. The same goes to the Remote Server Application Connectivity Test where we combine the HTTP and web existence (URL) diagnostic tests. However, the row has been filled with totally different items. Apparently, the failure cases have been generalized into problem layers where the potential network problems may occur. This is depicted in Table 4.2. Another major modification to the design is the introduction of decision pairs, which hold the mappings between the result of the each diagnostic test and the corresponding decision. Based on the decision of each diagnostic test, a final decision will be inferred based on the rules of preferences that the group has decided. This final decision will determine whether the respective layer is experiencing any problem. This layer then will determine the launching of second stage tests which will be discussed in detail in section 4.9.2 later in this document.

Design Rationale:

- By running a single test that produces two different results, it helps to mitigate the time taken to complete the whole diagnosing process. The generalization of the failure cases into problem layers is to solve the drawback of the previous design which in the case of errors occur are not within the scope of failure scenarios' list. As we go along, the group has decided to use the decision pair, as we are aware that the diagnostic result cannot be too decisive. The result is not as simple as yes or no situation but lies in these probabilities: 'Cannot Be Tested', 'OK', 'Maybe OK', 'Neutral', 'Maybe Not OK' and 'Not OK'. The decision that is based on the result will also has its own probabilities: 'Likely Problem', 'Unlikely Problem', 'Problem', 'No Problem' and 'Not Related'. This will be discussed in more details below.

Design Drawback:

- We realize that a fast and efficient inference tool does not need all diagnostic tests' results to determine that certain layers are having problems. Apparently, the allocation of decision pairs in all the blocks of the Inference Table seems too strict and unnecessary.

Table 4.2: Refined Inference Table (I)

Basic Tests Problem Categories	Configuration Test	Local Connectivity Test	Root DNS Servers Connectivity Test	DNS Consistency Test	Remote Server TCP/IP Level Connectivity Test		Remote Server Application Level Connectivity Test	
					TCP	IP	URL	HTTP
Local Host Problem (Physical Layer or Configuration)	CN - LP OK - UP NO - PR MO - UP MN - NR NE - NR	CN - LP OK - UP NO - LP MO - UP MN - LP NE - LP	CN - LP OK - UP NO - LP MO - UP MN - LP NE - LP	CN - LP OK - UP NO - LP MO - UP MN - LP NE - NR	CN - LP OK - UP NO - LP MO - UP MN - LP NE - NR	CN - LP OK - UP NO - LP MO - UP MN - LP NE - NR	CN - LP OK - UP NO - LP MO - UP MN - LP NE - NR	CN - LP OK - UP NO - LP MO - UP MN - LP NE - NR
Network Problem (IP Level Connectivity)	NR	CN - NR OK - UP NO - LP MO - UP MN - LP NE - LP	CN - NR OK - UP NO - LP MO - UP MN - LP NE - LP	CN - NR OK - UP NO - LP MO - UP MN - LP NE - NR	CN - NR OK - NP NO - NR MO - UP MN - NR NE - NR	CN - NR OK - NP NO - PR MO - UP MN - LP NE - LP	CN - NR OK - NP NO - NR MO - UP MN - NR NE - NR	CN - NR OK - NP NO - NR MO - UP MN - NR NE - NR
DNS Problem	NR	NR	CN - NR OK - NP NO - LP MO - NR MN - LP NE - NR	CN - NR OK - NR NO - PR MO - UP MN - LP NE - LP	NR	NR	NR	NR
Remote Server Problem (Transport or Application Layer)	NR	NR	NR	NR	CN - NR OK - NR NO - LP MO - NR MN - LP NE - LP	NR	CN - NR OK - NP NO - LP MO - UP MN - LP NE - LP	CN - NR OK - UP NO - LP MO - UP MN - LP NE - LP
Others (Performance)	NR	CN - NR OK - UP NO - LP MO - LP MN - PR NE - LP	CN - NR OK - UP NO - LP MO - UP MN - LP NE - LP	CN - NR OK - UP NO - LP MO - UP MN - LP NE - LP	CN - NR OK - UP NO - LP MO - UP MN - LP NE - LP	CN - NR OK - UP NO - LP MO - UP MN - LP NE - LP	CN - NR OK - UP NO - LP MO - UP MN - LP NE - LP	CN - NR OK - UP NO - LP MO - UP MN - LP NE - LP

Legend:

Test Results
Each diagnostic test will return either of the followings as the result:

Table 4.3: Result of Diagnostic Test

Abbrev.	Meaning	Description
CN	**Cannot be tested**	**The diagnostic test cannot be carried out due to not enough information from the previous test(s) or does not meet the pre-conditions**
OK	**OK**	**Explicit indicator of success from that diagnostic test**
NO	**Not OK**	**Explicit indicator of error from that diagnostic test**
MO	**Maybe OK**	**Implicit indicator of success from that diagnostic test**
MN	**Maybe Not OK**	**Implicit indicator of error from that diagnostic test**
NE	**Neutral**	**Balanced indicator of error and success from that diagnostic test**

Decision based on Test Results
Each diagnostic test's result will be mapped to either of the following decisions:

Table 4.4: Mapping of the result of Diagnostic Test to its corresponding decision

Abbrev.	Meaning	Description
NR	**Not Related**	**The diagnostic test's result is not associated to the corresponding layer.**
PR	**Problem**	**The corresponding layer is having problem.**
NP	**No Problem**	**The corresponding layer has no problem.**
LP	**Likely Problem**	**The corresponding layer may have problem.**
UP	**Unlikely Problem**	**The corresponding layer may not have problem.**

Preferences of Final Decision

PR >> NP >> LP >> UP >> NR

Description:
Final decision will determine which layer to be launched in the second stage test. The decision of each diagnostic test will be evaluated to get the final decision. The final decision is derived from the rules as set above. The leftmost which is the Problem(PR) will have highest weight(priority) while the rightmost, which is Not Related(NR) carries the least priority.

4.2.5 FINAL REFINED INFERENCE TABLE (II)

Design Strategy:

- This method has been chosen as our final design choice to diagnose network-related problems. It refers exactly to the previous design but a minor improvement has been imposed in order to make this design the ideal one. The modifications include eliminating the unnecessary decision pairs as a solution to the mentioned drawback in the previous design. Another amendments that have taken place are a new entry to the problem layers which is the reachability problem layer and removing others(performance) entry from the table. In this stage, the group has also decided to conduct the diagnostic tests of Remote Server TCP/IP Level Connectivity Test and Remote Server Application Connectivity Test per IP address of the web server.

Design Rationale:

- The idea of having reachability problem layer in the Inference Table is because to distinguish between the general network conditions (the local and external connectivity) and network reachability from host to the remote web servers. We found that the other issues (such as network performance, MTU mismatch, network stability and etc) are negligible. Thus, the entry is removed from the table. Moreover, the removing of unnecessary decision pairs from the Inference Table will eventually make the whole process of diagnosing the network problem a more flexible task.

Table 4.5: Final Refined Inference Table (II)

First Stage Tests / Problem Categories	Configuration Test	Default Gateway Connectivity Test	Root DNS Servers Connectivity Test	DNS Consistency Test	Remote Server IP/TCP Connectivity Test		Remote Server Application Test	
					IP	TCP	HTTP	URL
Local Host Problem	CN - LP OK - UP NO - PR MO - UP MN - LP NE - LP	CN - LP OK - UP NO - LP MO - UP MN - LP NE - LP	CN - LP OK - UP NO - LP MO - UP MN - LP NE - LP	NR	NR	NR	NR	NR
Network Problem	NR	CN - NR OK - UP NO - LP MO - UP MN - LP NE - LP	CN - NR OK - UP NO - LP MO - UP MN - LP NE - LP	NR	NR	NR	NR	NR
DNS Problem	NR	NR	NR	CN - NR OK - NP NO - PR MO - UP MN - LP NE - LP	CN - NR OK - NR NO - LP MO - NR MN - NR NE - NR	CN - NR OK - NR NO - LP MO - NR MN - NR NE - NR	CN - NR OK - NR NO - LP MO - NR MN - NR NE - NR	CN - NR OK - NR NO - LP MO - NR MN - NR NE - NR

Reachability Problem	NR	NR	NR	NR	CN - NR OK - NP NO - PR MO - UP MN - LP NE - LP	CN - NR OK - NP NO - NR MO - UP MN - NR NE - NR	NR	NR
Remote Server Problem	NR	NR	NR	NR	NR	CN - NR OK - NR NO - LP MO - NR MN - LP NE - LP	CN - NR OK - NR NO - LP MO - NR MN - LP NE - LP	CN - NR OK - NP NO - LP MO - UP MN - LP NE - LP

4.3 DIAGRAM OF ANFIT MECHANISM

As a conclusion, the mechanism of ANFIT can be illustrated as below . (Refer to Figure 4.3)

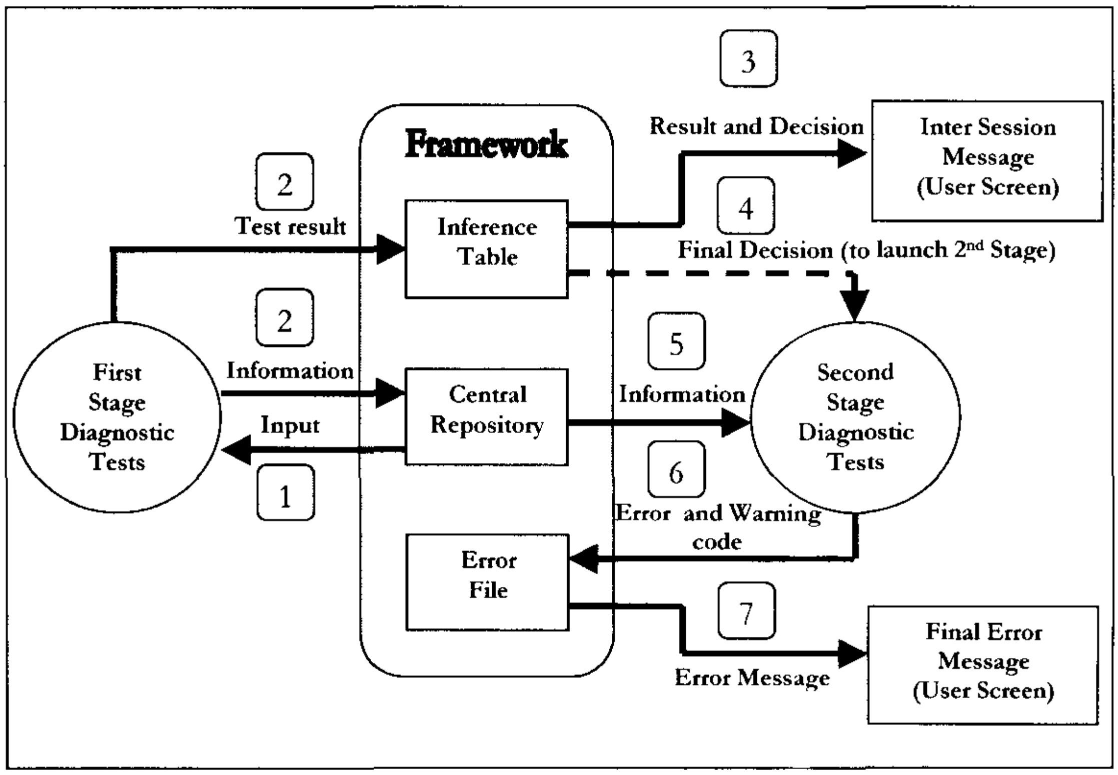
3
Result and Decision
Inter Session Message (User Screen)
Framework
2
Test result
Inference Table
4
Final Decision (to launch 2nd Stage)
2
Information
5
Information
First Stage Diagnostic Tests
Central Repository
Second Stage Diagnostic Tests
Input
1
6
Error and Warning code
Error File
7
Error Message
Final Error Message (User Screen)

Figure 4.3: Mechanism of ANFIT

4.4 PROGRAMMING LANGUAGES

It was decided during the early stage of the project that two different programming languages were needed to cater for two distinct parts of the project which are the framework and diagnostic tests (First and second stage). Framework and diagnostic tests are of disparate nature that Java has been picked as the language of choice for developing framework while python was deployed in implementing diagnostic tests.

Java Programming Language

Framework development requires a system programming language like Java, in order to optimize the performance of ANFIT. Java is known to have faster run time than other programming languages like C or C++. Furthermore, Java facilitates a good data structure for the framework with its object-oriented feature. As a consequence, it provides ease of integration or interaction among objects such as diagnostic tests objects, problem objects and etc. Java also has support for other languages, so python commands (in order to execute diagnostic tests) can be run through it. Java is also platform independent and this significant trait can ensure the extensibility and portability of the project in the future.

Moreover, the main feature of framework is the inference table of which is portrayed using XML file structure. XML documents tend to have a very explicit structure that is easily addressed by a language like Java. Although there are available implementations of standard XML parsers in many languages, including C, C++, Tcl, Perl and Python but the XML parser we are using which is SAX (Simple API for XML) is designed in and for Java. SAX which is an event-based interface parser reads the document and tells the program about the symbol it finds, as it finds them. For example, it will notify the application when it finds a start tag, when it finds character data and when it finds the end tag. Thus, the implementation of Inference Table is made feasible by using this approach, hence making Java as the ideal language to use.

Python Programming Language

Scripting language was decided as the ultimate type of language for developing diagnostic tests as it possesses such traits as providing fast build-cycle turnaround(no compilation needed), facilitating dynamic typing(no declaring of variables needed) , and offering interactive environment where we can create, view or change objects at run-time. Scripting language is also better for rapid development and reusing code which are apt for test automation. All these significant features are very useful to ensure the efficiency and flexibility of the diagnostic tests.

Based on the discussion with our supervisor, two scripting languages are taken into considerations which are Perl and Python. However, none of the group members has any experience in dealing with any of the languages, thus side by side comparisons were performed in order for us to select the most suitable language. From the research, we found that Perl' s regular expressions and quoting mechanisms are more powerful than Python, which are good for common application-oriented tasks but its lack of readability and object orientation have made us turn to Python. Furthermore, Python's easy, clear and precise syntax makes not only programming easier but also causes the debugging and testing processes more simple and efficient, hence making it a very convenient programming language for beginners like us. Although, most of the load was put into learning Python but due to the ease-of-use nature of the language, increased productivity has complemented the time spent. Moreover, the Python's enforcement on indentation style has been an aid for code maintainability as well as fostering us for a good programming practice. Apart from that, Python is also available on an incredibly wide range of hardware and software platforms. This factor will allow this project to be extensible in future due to the portability nature of the language. All the aforementioned reasons made Python the language of choice in developing the diagnostics tests.

4.5 TWO-STAGES STRUCTURE

ANFIT consists of two stage tests in order to reduce time in executing all the diagnostic tests. In the first stage, ANFIT can infer the location of network fault meaning, which layer is having problem. As in the second stage, ANFIT will produce detail information about the possible cause of failure. This kind of structure is appropriate because to detect and to analyze are two different tasks that require different kinds of approach respectively. Apart from that, the varied time needed to accomplish each task since the latter consumes more time. Intersession message will be displayed in between the two stages.

4.5.1 FIRST STAGE: COARSE-GRAIN TEST

Design Strategy:

- This stage adopts a breadth style of diagnosing which will execute all the diagnostic tests sequentially to gather all the results. Each diagnostic test is designed in such way that it is sufficient enough to detect where the error lies, as no aspect of thoroughness is needed yet at this point. It is also at this stage that inference table holds a significant role by facilitating us in mapping the diagnostic test results and decisions, determining the final decision that eventually concludes which problem layer should be launched in the second stage.

Design Rationale:

- The breadth style will ensure ANFIT can accomplish running all diagnostic tests in short time yet possess enough and useful information. Essentially, this information is needed to generate the intersession message and to launch the second stage. In the case of limited time to conduct the second stage, this approach will also ensure all diagnostic tests produce enough coverage of practical information to the users.

4.5.2 SECOND STAGE: FINE-GRAIN TEST

Design Strategy:

- This stage adopts a more comprehensive mechanism for each detailed diagnostic test since ANFIT will only execute these tests on the problem layers that has been detected having error as determined by the first stage. These detailed diagnostic tests which will run in parallel, assimilate a more exhaustive analysis and more combination of tools and technologies. It is at this stage that failure scenarios are referred as guideline to design the detailed diagnostic tests.

Design Rationale:

- Failure scenarios have been identified to consist of general and specific cases that each should reveal one or more errors. Realizing that, detailed diagnostic tests need to be crafted in order to ensure a higher level of certainties in determining the what is the cause of failure and why it occurs.

4.6 USER INPUT AND OUTPUT

This section illustrates the input requires from the users and also the output that they will benefit from ANFIT. The inputs are as enumerated below:

- User Level: [a] Advanced or [n] Normal
- Valid URL

The outputs are as follow:

- Intersession Message
- Error and Warning Message

4.7 INTERSESSION MESSAGE

Design Strategy:

- This part will display out to the users a summary which problem layer has failure hence; informing the users ANFIT is carrying out second stage test(s) for that specific problem layer(s).

Design Rationale:

- The purpose of this part is to notify user that ANFIT has already detected where the problem occurs as well as to retain user's interest while we are pursuing with further detailed diagnostic test(s).

4.8 ERROR AND WARNING MESSAGE

Design Strategy:

- This part will display the summary of condition for each problem layer. If the problem layer has error, ANFIT will give precise error message indicating what the failures are. Apart from that, any unusual conditions detected by ANFIT but not classified as error will be returned as warning message. Different level of users will receive different kind of error messages that suits their level of technical background. Simple suggestions will be provided to the user wherever possible.

Design Rationale:

- This part will act as the final interaction between ANFIT and user hence the information given must be appropriate and useful to both types of user. The simple suggestions are also included though it is not a primary objective of ANFIT, in the hope it will aid user in troubleshooting the problem.

4.9 INDIVIDUAL DIAGNOSTIC TEST DESIGN DETAILS

This section illustrates the design details of each individual diagnostic tests.

4.9.1 FIRST STAGE TESTS

4.9.1.1 DIAGNOSTIC TEST 1: CONFIGURATION TEST

Design Objective:

- To check the configuration of local sending host.

Input:

- None

Output:

- IP address of the local host
- IP address of the default gateway
- Test Result

Tools/Technologies:

- ifconfig
- netstat

Pre-Conditions:

- None

Design Strategy:

- Get the interface(s) of the local host. This does not include the loopback interface and IPv6 interface.
- If the interface is more than one, the Test Result is CN. DONE.
- Get IP address, Netmask and Default Gateway for the interface.
- If anything cannot be obtained, the Test Result is NO. DONE.
- Check the validity of IP address. The IP address must be in a valid format, and not zero. It must also not the loopback IP address and not a private IP address.
- If anything fails, the Test Result is NO. DONE.
- Check the validity of Netmask by checking the prefix length
- If the Netmask is not valid, the Test Result is NO. DONE.
- Check whether the Default Gateway is in the same IP subnet.

- If the Default Gateway is not in the same IP subnet, the Test Result is NO. DONE.
- Others, the Test Result is OK.

Design Rationale:

- In this diagnostic test, ifconfig is used to extract and check the local host's IP address and subnet mask. ifconfig is used because it is useful tool for detecting bad IP address(s), incorrect subnet mask and also improper broadcast address(s). It is also a utility that provides a good way of getting a snapshot of the IP configuration of the local host. Meanwhile, netstat is used to get the IP address of the default gateway. Since netstat gives a quick list of all incoming and outgoing connections and a summary of protocol statistics, this is the tool chosen to serve the purpose mentioned before.

4.9.1.2 DIAGNOSTIC TEST 2: DEFAULT GATEWAY CONNECTIVITY TEST

Design Objective:

- To detect whether or not local connectivity contributes to the network failures

Input: *(from Diagnostic Test 1)*

- IP address of local host
- IP address of default gateway

Output:

- Test Result

Tools/Technologies:

- arp
- ping
- tcpdump

Pre-Conditions:

- The Test Result of Configuration Test is OK. Pertain

Design Strategy:

- Launch the tcpdump in advance to monitor the outgoing and incoming of ARP packets
- Delete the mapping translation of default gateway IP Address and its MAC Address entry that is cached in the routing table
- Send packets to default gateway IP address using ping to force ARP request and ARP response process
- Result is determined through ARP response received from default gateway to indicate local connectivity is not having any problem
- All the steps above are repeated for ten times to get different uncertainty level of the result as based on the table below

Test Results:

- Based on number of successive response from default gateway

Table 4.6: Mapping of result and condition for diagnostic test 2

Result	Response from default gateway
NO	0 or Mismatched Results
MN	1- 5
NE	6 – 7
MO	8 – 9
OK	10

Design Rationale:

- Since the mapping translation of the default gateway IP Address has been deleted from the ARP table entry (a table where the responses to previous ARP requests are cached), the source host is apparently unable to obtain the MAC Address which is essential in sending the packets out. The source host will then broadcast to all the hosts in the subnet asking on who has the MAC Address of the default gateway. By doing so, the default gateway is now 'forced' through ping to elicit an ARP Response as to reply the ARP Request as broadcast by the source host. Since the default gateway is now responding, we could deduce that the local connectivity is working well and may not be the caused of failure while obtaining the web page.
  Due to the erratic behaviour of the network, the ARP Request is broadcast 10 times to all the hosts in the subnet to observe the corresponding ARP Response that will come back. This will benefit in term of drawing out the test results so that we will have the different level of certainties on the local connection status as outlined on the table above.

4.9.1.3 DIAGNOSTIC TEST 3: ROOT DNS SERVERS CONNECTIVITY TEST

Design Objective:

- To detect whether or not external connectivity contributes to the network failures

Input:

- None

Output:

- Test Result

Tools/Technologies:

- dig

Pre-Conditions:
- The Test Result of Configuration Test is OK.
- The Test Result of Local Connectivity Test is NE, MO or OK.

Design Strategy:
- Contact 13 Root DNS Server by using "dig" without retransmission

Test Results:
- Number of success

Table 4.7: Mapping of result and condition for diagnostic test 3

Result	Number of Success
NO	0
MN	1-3
NE	4 - 6
MO	7 - 9
OK	10 - 13

Design Rationale:
- Root DNS Servers are chosen as qualified parameters as they have been identified as highly reliable and robust servers. Therefore, this diagnostic test can simply confirm that no failure exist in external connectivity if the root DNS servers replies the query. Dig is the tool of choice as it can return information that can easily be parsed automatically. Pinging root DNS servers can also be used as the alternative approach since it can return information on whether the source can reach the destination (and vice versa). However, it takes longer than dig and the lengthy results returned are not entirely essential. Thus, dig is a more preferred choice.

4.9.1.4 DIAGNOSTIC TEST 4 : DNS CONSISTENCY TEST

Design Objective:
- To detect failures from local and remote DNS server(s)
- To detect consistency, completeness and correctness of the information return by the DNS server(s)

Input:
- Fully qualified domain name (FQDN)

Output:
- Current LDS IP Address
- List of Web server's IP Address(s) from LDS
- List of Web server's IP Addresses from ADS

- ADS List from Merge List
- Test Result

Tools/Technologies:
- dig

Pre-Conditions:
- The Test Result of Configuration Test is OK.
- The Test Result of Local Connectivity Test is NE, MO or OK.
- The Test Result of Root DNS Server Connectivity Test is NE, MO or OK.

Design Strategy:
1. Get Information from LDS
- to check availability of All LDSs
- to decide Current LDS
- to get Resolved Name from Current LDS – (i)
- to get IP List in Current LDS with Resolved Name
- to get ADS List in Current LDS with Resolved Name

2. Get Information from ADS
- to get Resolved Name from ADS – (ii)
- to get ADS List from ADS(s) with Resolved Name

3. Get Information from Level Above
- to get Resolved Name and ADSs' List from level above DNS Servers (iii)
- to get TLD servers from Root server
- to get ADS List from TLD server

4. Perform Consistency Check
- to check consistency of Resolved Name in i, ii, iii
- to check consistency of answers return by all DNS Servers. The answers can be of any of the followings:
  - Name Server(s)
  - SOA record

5. Merge List
  - to merge all ADS List (from LDS,ADS and Level Above) – if all DNS Servers return consistent answers.
  - to get IP List from ADS in the Merge List

7. Perform Caching Check
- to check whether the IP Address(s) return by LDS is subset of IP List from Merge List

Note:
Resolved Name is the final name resolved for the domain in query. If the domain name has any canonical name (CNAME) or alias exists, so the Resolved Name is that canonical name. If the domain name has no alias, the Resolved Name will take the domain name as the value.

Test Results:

Table 4.8: Mapping of result and condition for diagnostic test 4

OK	• Current LDS is Primary LDS • All non-primary LDSs are available • IPList in Current LDS is not empty • IPList in ADSs is not empty • IPList in Current LDS is subset of IPList in ADSs
MO	• Current LDS is Primary LDS • Not all of non-primary LDSs are available • IPList in Current LDS is not empty • IPList in ADSs is not empty • IPList in Current LDS is subset of IPList in ADSs
NE	• Current LDS is Non-primary LDS • IPList in Current LDS is not empty • IPList in ADSs is not empty • IPList in Current LDS is subset of IPList in ADSs
MN	(from Resolved Name or ADSs's inconsistencty) • IPList in Current LDS is not empty AND • ADSs' List is empty
NOK	• IPList in Current LDS is empty OR • (ADSs' List is not empty AND IPList in Current LDS is not subset of IPList in ADSs)

Design Rationale:

- One of the potential causes of not getting to a web site is the DNS name resolution fails to work. The failure may occur due to such problems as DNS server is unable to perform its task due to internal failure or unable to handle the process or the DNS records are poorly maintained. The problems may also arise if the answers returned by the name server are incomplete, inconsistent and incorrect. In order for that, this diagnostic test is designed in such way that will detect these errors. Dig is chosen as the tool for interrogating the DNS servers since its consistent output format is easy to parse automatically. Apart from that, the tool can also return comprehensive information that mimics DNS packet contents in one single command query. These significant traits of dig are very helpful in diagnosing the DNS errors.

4.9.1.5 DIAGNOSTIC TEST 5: REMOTE SERVER IP/TCP CONNECTIVITY TEST

Design Objective:

- To check IP level connectivity and TCP level connectivity between the host and specific web servers

Input:

- List of Web server's IP Address(es) – from Diagnostic Test 4
- Port number

Output:

- List of non-error Web server's IP Address(es)
- IP Connectivity Test Result
- TCP Connectivity Test Result

Tools/Technologies:

- telnet

Pre-Conditions:

- List of Web server's IP Address(es) – from Diagnostic Test 4 should not be empty

Design Strategy:

- The diagnostic test will be carried out per IP address of web server. In this diagnostic test, telnet is used to make sure that the remote system (web server) is listening on port 80 or 443 depending on the protocol (http or https). Once the connection is established, it is known that the remote web server is listening on port 80 or 443 and there is no firewall between the two hosts that is filtering that specific port. The steps taken are shown below:

  – Send TCP SYN packet to open connection with Remote Server (port number 80 is used if the protocol is http, port number 443 is used if the protocol is https)
  – Receive packet from Remote Server (SYN/ACK)
  – Close connection by sending FIN

- This test will be conducted 10 times for each IP address

Test Results:

- The results will be based on the reply given by the remote host during the telnet session between the two hosts, which will be conducted 10 times. There are 3 types of reply, which are Connected, Connection Refused or Timeout (explicit or implicit). Based on the reply, it will then be decided either IP or TCP fails, or both. To determine the test result whether it is NO, MN, NE, MO or OK, number of fails is counted as shown in the tables below:

IP Level Connectivity

Table 4.9: Mapping of result and condition for diagnostic test 5 (IP level)

Result	Number of successful connection
NO	0
MN	1-3
NE	4-6
MO	7-9
OK	10

TCP Level Connectivity

Table 4.10: Mapping of result and condition for diagnostic test 5 (TCP level)

Result	Number of successful connection
NO	0
MN	1-3
NE	4-6
MO	7-9
OK	10

Design Rationale:

- Telnet is an excellent utility for determining if a TCP connection can be established between two hosts (local and remote) using a specific TCP port. Apart from that, telnet is used due to its simplicity.

4.9.1.6 DIAGNOSTIC TEST 6: REMOTE SERVER APPLICATION CONNECTIVITY TEST

Design Objective:

- To check whether the HTTP Request-Response connectivity is succeed between the host and web servers
- To check whether the web page exists or not and to conclude the reason behind the failure of obtaining the page as per indicated by the error code in HTTP Response Message

Input:

- List of Web server's IP Address(es) – From Diagnostic Test 4
- Web page (trimmed from the full URL)

Output:

- Error code
- New URL (in case of redirection)
- Test Result

Tool/Technologies:

- wget

Pre-Conditions:

- List of non-error Web server's IP Address(es)

Design Strategy:

- The diagnostic test will be carried out per IP address of web server. Owing to the fact that HTTP protocol operates in a bidirectional way, it can be inferred from the test that the HTTP connection fails if the host is unable to send the HTTP Response Message out to the specific web server's IP address due to failure at the TCP level previously or the web server is incapable of replying any HTTP Response Message (either success code or error code).
- Subsequently, this diagnostic test will regard that the page exists and the request is fulfilled if the success status code (2xx series) is replied in the HTTP Response Message. The steps to be implemented are as below:

  - Open TCP Connection to the remote web server
  - Send HTTP Request Message to the web servers
  - Receive HTTP Response Message from the web servers

HTTP Request Message will be sent for 10 times and the response received will be analysed as the table outlined below to determine the result certainty level of this test.

Test Results:

- *HTTP Level Connectivity*

Table 4.11: Mapping of result and condition for diagnostic test 6 (HTTP level)

RESULT	Number of any Response Message (either success or error status code)
NO	0
MN	1- 3
NE	4 – 6
MO	7 – 9
OK	10

- *Web Page Existence*

Table 4.12: Mapping of result and condition for diagnostic test 6 (Web Page Existence)

RESULT	Number of 200 Response Message (success status code)
NO	0
MN	1- 3
NE	4 – 6
MO	7 – 9
OK	10

Design Rationale:

- The HTTP Request Message is sent 10 times to draw a set of different certainty levels in order to determine whether the cause of failure in obtaining a web page is really have been caused by the failure of HTTP request-response protocol or caused by the client does not get the successful status code (200 status) as the HTTP Response back. HEAD is largely used if user wants to see the headers returned by the server for a particular document, without actually downloading the document. This is exactly what the HEAD request method provides. HEAD looks and works exactly like GET, only with the difference that the server only returns the headers and not the document content. This is very useful for programs like link checkers, people who want to see the response headers (to see what server is used or to verify that they are correct) and many other kinds of uses. Nevertheless by default, Wget is very simple to invoke. The basic syntax is will simply download all the URLs specified on the comman line. When invoked with a --spider option specified, Wget will behave as a Web "spider", which means that it will not download the pages, but just check that they are there. This simple usage is what makes Wget appropriate to be chosen for this diagnostic test.

4.9.2 SECOND STAGE TESTS

4.9.2.1 DETAILED DIAGNOSTIC TEST 1: LOCAL HOST PROBLEM TEST

Design Objective:

- To check any specific problem associated with local host

Problems to detect :

- Hardware problem
  - Faulty Cable
  - Faulty Network Card
  - Others

- Configuration Problem

- No IP Address
- Invalid IP Address
- Invalid Network Mask
- Invalid Default Gateway
- IP Conflict (Duplicate IP Address for Different Hosts)
- Others

• DHCP Problem
  - No Response from DHCP server
  - Invalid Response from DHCP Server
  - Lease Time Expiration

Design Strategy:

• In order to detect abovementioned problems, these are the approaches that might be carried out:
  - Passive Listening
  - DHCP Server Check
  - Configuration Check
  - Ping to Default Gateway
  - ARP to Default Gateway
  - IP Address Conflict Check
  - IP Subnet Validity Check

4.9.2.2 DETAILED DIAGNOSTIC TEST 2: NETWORK PROBLEM TEST

Design Objective:

• To check any specific problem associated with general network condition

Problems to detect :

• LAN Problem
• Internet Problem

• Others
  - Firewall Problems
  - NAT Problems

Design Strategy:

• Dig to Root DNS Server

4.9.2.3 DETAILED DIAGNOSTIC TEST 3: DNS PROBLEM TEST

Design Objective:

- To check any specific problem associated with DNS servers.

Problems to detect:

- Local DNS Server Problem
    - No Response
    - Error
    - No IP level connectivity

- Intermediate DNS Server (g-TLD or cc-TLD) problem
    - No Response
    - Error
    - No IP level connectivity

- Authoritative DNS Server Problem
    - No Response
    - Error
    - No IP level connectivity

- Inconsistency Problem
    - Conflict

Design Strategy:

- Dig to Local DNS Servers
- Ping to Local DNS Servers
- Traceroute to Local DNS Servers
- Dig to Intermediate DNS Servers
- Ping to Intermediate DNS Servers
- Traceroute to Intermediate DNS Servers
- Dig to Authoritative DNS Servers
- Ping to Authoritative DNS Servers
- Traceroute to Authoritative DNS Servers
- Check consistency between DNS Servers

4.9.2.4 DETAILED DIAGNOSTIC TEST 4: REACHABILITY PROBLEM TEST

Design Objective:

- To check any specific problem associated with path between local host and web server

Problems to Detect:

- Remote Server Connectivity Problem:
  - Web server is down
  - MTU Mismatch
  - Unidirectional Connectivity

- Others
  - Firewall problems

Design Strategy:

- The detailed diagnostic test will be carried out per IP address of web server. Since the purpose of this detailed diagnostic test is to check the path between the local host and the remote server, the group had decided to do the test as below:

  - Telnet to the remote server
    If telnet is successful in forming a connection, then it means that the web server is listening and there is no firewall between the two hosts that is filtering the specific port. Thus, it means that there is no problem with remote server hence the reachability from local host to the remote server is ok.

  - Traceroute to Remote Server
    TraceRT utility is closely related to PING. It sends a series of PINGs with varying TTLs. It traces packets from the local host to a remote host and prints the name (if it can be determined) and IP address of each gateway along the route to the remote host. Thus, this can be used to locate where exactly along the path that is having problem(s) and therefore can determine the cause of failure.

4.9.2.5 DETAILED DIAGNOSTIC TEST 5: REMOTE SERVER PROBLEM TEST

Design Objective:

- To check any specific problem associated with the remote web server(s)

Problems to Detect:

- Transport Layer Problem
  - Port 80 doest not exist
  - TCP Connection Setup Error
  - Others

- Application Layer Problem
  - Page does not exist (400 - Bad Request, 404 - Not Found)
  - Unauthorized Access
  - HTML errors
  - Incompatible browser
  - Code errors
  - File contains no data

- Others
  - Web server is busy
  - Web Proxy Server Problem

Design Strategy:

- The diagnostic test will be carried out per IP address of web server.
  - Send TCP SYN
  - Setup TCP Connection
  - Send HTTP GET for URL
  - Send HTTP GET for Web server Default Page
  - telnet
  - fetch

Chapter 5

FRAMEWORK ARCHITECTURE

5.1 DATA STRUCTURE

The inference table is made up of a series of tests and the problems that can be identified from the results of the tests. The table must be stored in some kind of data structure so that when a particular test is run, its result can be looked up in the data structure and the corresponding problem identified.

An object oriented data structure is used which consists of the following objects:

- **Test:** A test object represents a single test. It consists of a name, which identifies the test, a command used to run the test (since the test may be written in a language other than Java) and a set of mappings called ResultBlocks. A test might also have a list of arguments, which are appended to the end of the test command before the test is run. Every test has a run method, which executes the command and returns the result of the test. A helper class called TestRunner is used to do this.

- **ResultBlock:** Every result block is made up of a problem name and a set of result-decision pairs. (This would correspond to a single cell in the inference table.)

- **Problem:** This class represents a problem object, which consists of the name of the problem and a set of decision pairs.

- **DecisionPair:** A decision pair holds the mapping between the result of a test and the decision.

- **InferTable:** This object represents the whole table. It is made up of a collection of test and problem objects.

- **CentralRepository:** This is a central hash table, which stores values of interest including the results of all the tests run so far. Sometimes a test might need information that has already been computed by a previous test. In such cases, the test can consult this hash table to lookup what it requires. By default, everything output by a test is stored in the hash table as a "name-value" pair.

5.2 POPULATING THE DATA STRUCTURE

Now that the data structure has been designed, the next step is filling the data structure with tests and problems as specified in the inference table. There are a number of different ways in which this can be done:

- **Hard coding**: The data is written within the code itself. This method is not flexible because if the data changes, the code would need to be edited and recompiled.

- **Flat File Structure**: The data is written in a file with fields separated by delimiter characters such as "tab". This method is not suitable because of the complexity of the data. It would also mean that the programmer would have to write parsing routines in order to extract the data from the file.

- **XML File Structure**: The data is written in XML format using meaningful tags. The main advantage of this approach is that it is extensible i.e. in the future, if more tests need to be added or existing tests need to be removed or altered, this can be done by simply editing the XML data. Also, XML parser libraries are widely available and can be used to parse the XML data and fill the data structure without any hassle.

After careful consideration of the pros and cons of each method, it was decided to store the inference table data in an XML file.

An excerpt of the XML file is shown below:

```
<inferenceTable>
        <test>
                <testname>Configuration Test</testname>
                <command>python Test1.py</command>
                 <args></args>
                  <problemlist>
                        <problem>
                                <problemname>DNS Problem</problemname>
                                <resultset>
                                        <result>CN-LP</result>
                                        <result>OK-UP</result>
                                        <result>NO-PR</result>
                                        <result>MO-UP</result>
                                        <result>MN-LP</result>
                                        <result>NE-LP</result>
                                </resultset>
                        </problem>
                </problemlist>
        </test>
```

An XML parser was downloaded which is able to extract data from the XML file based on tag names. In this way objects can be created and the data structure can be populated.

5.3 WRITING THE DIAGNOSTIC TESTS

Rules:

- The diagnostic tests can be written in any language.
- The command used to run the diagnostic test and any additional arguments must be specified in the XML file. The arguments are appended to the command at run-time, by looking up the specified keyword in the configuration map.
- All diagnostic tests must have a single result (OK, MO etc). This result must be preceded by the keyword "RESULT". All results are stored in the hash table as the test name followed by result. For example, the result of MyTest would be stored as MyTest_Result.
- The output of all diagnostic tests goes into the hash table. If an entry for a particular keyword already exists in the hash table, it is overwritten. Therefore care must be taken when deciding what a diagnostic test has to output.

5.4 STORING ERRORS AND WARNINGS

Error messages for the entire system are stored in a flat file data structure. This is suitable since the data is not complex; it is just a series of identifiers and messages. An extract from the errors file is shown below.

ERR1: This is error 1.
ERR2: This is error 2.

- There are two error files; one containing error messages for normal users and the other for more advanced users.
- Error messages can have the keyword $DATA which is replaced by actual data at run-time.
- Based on the level of the user an error file is read and a hash table is set up mapping identifiers and messages

5.5 DISPLAYING ERRORS AND WARNINGS

- Errors and warnings are identified during the diagnostic tests and must begin with ERR or WARN respectively. The framework recognises this kind of output and instead of storing it in the central hash table it is stored in lists called errorsFound and warningsFound.

- If the error output by a test contains the character '_', all text after this character will be inserted into $DATA in the error message. For example if the error output is ERR1_xxx, the message for ERR1 is looked up in the error table and the message is extracted. If this message contains $DATA it is replaced by xxx.

- In the end, when errors need to be displayed to the user, every element in errorsFound and warningsFound is looked up in the hash table and the corresponding message is output.

5.6 RUNNING THE PROGRAM

- The main class is used to execute the program.

- The program initialises by first reading and parsing the XML file and storing all the data in the data structure.

- The user is prompted to enter their user level (normal or advanced). This is used to decide what kind of error messages to display. The errors file is read and the error table initialised.

- The user is then asked to enter a URL. The format of the URL must be validated first. IP addresses are not permitted. The program also removes any leading *http://* if it exists and extracts the domain name, port number (default 80) and html page if specified. All this information is stored in the central hash table so that it can be looked up by other diagnostic tests if required.

- The program then begins to execute each diagnostic test. Each test is run using the Java Runtime class using its specified command and arguments.

- Once all the diagnostic tests have been run, the program moves onto the second stage.

- The program then prints out a summary of the results and the errors identified.

Chapter 6

IMPLEMENTATION

The strategies that we adopted are exactly based on the design that we performed earlier but there are a few modification as enumerated below to suit the project as the need arise along the development.

6.1 FIRST STAGE DEVELOPMENT

6.1.1 DIAGNOSTIC TEST 1: CONFIGURATION TEST

Test Execution Command:

- python Test1.py

Automated Command(s):

- ifconfig -a
  - the -a flag instructs ifconfig to display information about all the interfaces in the system
- netstat -nr
  - the –nr flag instructs netstat to show the network address(s) in the routimg table as numbers not names

Design Modification:

- **Technical Limitation / Problem:**
  This test can only cater for a local host that has only one interface but in a normal network environment, it may have multiple interfaces. This problem was witnessed clearly during running ANFIT at the testing environment.
- ***Status / Solution:***
  No modification was made to cater for the problem which means ANFIT will treat the situation as CN under the circumstances of more than one interfaces.

6.1.2 DIAGNOSTIC TEST 2: DEFAULT GATEWAY CONNECTIVITY TEST

Test Execution Command:

- python Test2.py Pre-con1 IPAddress DefaultGateway
  - Pre-con1 is the result from Diagnostic Test 1
  - IPAddress means the IP Address of the source host
  - DefaultGateway means the IP Address of the default gateway

Automated Command(s):

- exec tcpdump –n –l arp
  - exec is used in order to have only one background process
  - the flag -n will not convert addresses (i.e., host addresses, port numbers, etc.) to names.
  - the flag -l will ensure the stdout line to be buffered. This is useful in order to see the data while capturing it. E.g., ``tcpdump -l | tee dat'' or ``tcpdump -l > dat & tail -f dat''
- arp –d DefaultGateway > /dev/null
  - the flag –d is to delete an entry for the host called hostname. Only the super-user may use this option.
  - /dev/null is from the Unix null device, used as a data sink
- ping –c l DefaultGateway > /dev/null
  - the –c flag is specified to stop ping after sending (and receiving) ECHO_RESPONSE packets. If this option is not specified, ping will operate until interrupted.
  - the l flag preload is specified so that ping sends that many packets as fast as possible before falling into its normal mode of behaviour. Only the super-user may use this option.

Design Modification:

- **Technical Limitation / Problem:**
  Tcpdump need to be killed once sufficient information is obtained because it cannot terminate by itself. However, if we have more than one processes, we will only kill the shell process which is the parent and not tcpdump process.
- **Status / Solution:**
  The problem is resolved by using exec command in tcpdump, resulting in the shell process being replaced with tcpdump process. Since tcpdump has now having the process ID of the shell, so we have no problem of terminating it.

- **Technical limitation / Problem:**
  Not enough time to monitor incoming and outgoing packet in tcpdump after pinging default gateway.
- **Status / Solution:**
  The problem is resolved by suspending the execution of tcpdump for 100ms. (time.sleep(0.1))

6.1.3 DIAGNOSTIC TEST 3: ROOT DNS SERVERS CONNECTIVITY TEST

Test Execution Command:

- python Test3.py Pre-con1 Pre-con2
  - Pre-con1 is the result from Diagnostic Test 1
  - Pre-con2 is the result from Diagnostic Test 2

Automated Command(s):

- dig @rootDNSserver +retry=1
  - retry = 1 means no retransmission

Design Modification:

- **Technical Limitation / Problem:**
  The IP addresses of root DNS servers may obsolete after some time.
- **Status / Solution:**
  This problem is negligible at the moment since the IP addresses of root DNS servers will undergo changes once in three years.

6.1.4 DIAGNOSTIC TEST 4: DNS CONSISTENCY TEST

Test Execution Command:

- python Test4.py Pre-Con1 Pre-Con2 Pre-Con3 domainName
  - Pre-con1 is the result from Diagnostic Test 1
  - Pre-con2 is the result from Diagnostic Test 2
  - Pre-con3 is the result from Diagnostic Test 3
  - domainName is the domain name retrieved from the URL entered by user

Automated Command(s):

- cat /etc/resolv.conf | grep nameserver
  - to get the list of local DNS server(LDS) IP addresses

- dig @currentLDS domainName +retry=3 +time=1 +noad
  - to get information from LDS
  - retry value is set to 3 which means at most 2 retransmission will occur
  - time out value is set to 1 which means the successive queries will take place between 1 sec
  - noad means no ADDITIONAL SECTION of the record will be displayed as the information is not required.

- dig @remoteDNSservers domainName +retry=7 +time=3 +norec +aa +noad
  - to get information from remote DNS servers such as root DNS Server, Top Level Domain Servers(TLD) and Authoritative DNS Servers

- retry value is set to 7 which means at most 6 retransmission will occur. Higher value is chosen than the LDS's value as to cater for remote reason
- time out value is set to 3 which means the successive queries will take place between 3 sec. Same reason as above applies on why the time out value for remote DNS servers is increased.
- norec means no recursive query will need to be performed by the queried DNS server.
- aa means the only the ADS of the queried zone will reply the query
- noad means no ADDITIONAL SECTION of the record will be displayed as the information is not required

Note:
If SOA record (containing zone name) , or CNAME (canonical name or alias of the domain name) were found during the look up, these names will then replace domainName in the further look up.

Design Modification:

- **Technical Limitation / Problem:**
  Inconsistency of the records returned by DNS servers may due to load balancing or poorly maintained records. Inconsistency due to load balancing is not an error while the latter should be treated as otherwise. However, ANFIT is unable to distinguish between these two(2) scenarios.
- **Status / Solution:**
  Since consistency of information returned by DNS servers play vital role in ensuring the DNS reliability so this limitation is taken seriously. Therefore, warning will be flagged to the user whenever any inconsistencies are detected by ANFIT.

- **Technical Limitation / Problem:**
  One of the sources for deriving ADS list comes from digging all level above ADS. However, a glitch was found in the program where it was apparent that query made to different TLD servers on the same query returned different answers. This has been detected due to the existence of two different types of TLDs which are generic-TLD (.com, .net,.org etc) and countrycode-TLD(.uk,.my,.de etc). Query made to root DNS servers on domain name such as 'tm.net.my' will return referrals on both cc-TLD and g-TLD servers. Thus requesting records from all the TLD servers will result in different answers as cc-TLD server will return g-TLD servers and g-TLD will return ADS list.
- **Status / Solution:**
  This issue is resolved by querying all the DNS servers above ADS using the zone name of one level below them. For example for the domain name 'tm.net.my', the root DNS servers will be queried for '.my' name servers' records , while '.my' DNS servers will be queried for '.net.my' name servers record and so on. This will alleviate the problem mentioned above.

- **Technical limitation / Problem:**
  Answers from DNS servers (be it authoritative or not) should be returned in ANSWER SECTION and referrals from DNS servers should be returned in

AUTHORITY SECTION of the record. However, ANFIT is incapable of classifying the category of records returned by the DNS Servers.

- **Status / Solution:**
  Although it is easy to parse the record returned by dig, in order to find the appropriate sections or to determine whether the DNS servers are authoritative or not, yet it is hard to determine whether the records sent by the non-authoritative DNS server are supposed to be the answer or referral. In spite of this, knowing that this issue is less significant than others, it is reserved for future development.

6.1.5 DIAGNOSTIC TEST 5: REMOTE SERVER IP/TCP CONNECTIVITY TEST

Test Execution Command:

- python Test5.py portnumber IPList
  - portnumber is either 80 or 443 depending on the protocols used (http or https).If the user does not enter the protocol, then by default the port number is 80
  - IPList is the list of web server's IP Address(es) obtained from Diagnostic Test 4

Automated Command(s):

- exec telnet WebServerIPAddress port
  - exec is used to have only one background process (optional in this test)
  - WebServerIPAddress is IP address of each web server in the list of Web server's IP Address(es) obtained from Diagnostic Test 4
  - port is the port number retrieved from framework

Design Modification:

- **Technical Limitation / Problem:**
  This diagnostic test performed 10 telnet sessions for each IP address in the list of web server's IP Address(es). This is considered as slightly hostile as it can be regarded as Denial-Of-Service attack by the web server.
- **Status / Solution:**
  Due to unstable network condition, we need to ensure that the remote server is listening as to confirm TCP/IP connectivity is not experiencing any problem.

6.1.6 DIAGNOSTIC TEST 6: REMOTE SERVER APPLICATION TEST

Test Execution Command:

- python Test6.py page OKIPList
  - page is the web page that is obtained from framework which is previously trimmed from the user URL input
  - OKIPList is the list of Web server that has succeeded on the TCP connection, obtained from Diagnostic Test 5

Automated Command(s):

- exec wget --spider -T 30 -t 3 'WebServerIPAddress'/'page'
  - the option –spider is used so that Wget will behave as a Web "spider", which means that it will not download the pages, just check that the page exists or not with the inclusive return of status code.
  - the flag –T = 30 is used to set the timeout interval to 30 seconds. This option will only commit after TCP connection is established.
  - the flag –t = 3 is used to set the retransmission value to at most 3 in case the web page is failed to be contacted
  - WebServerIPAddress is IP address of each web server in the list of Web server's non-error IP Address(es) obtained from Diagnostic Test 5

Design Modification:

- **Technical Limitation / Problem:**
  The sending of HTTP Request Message for 10 times to each specific web servers will cause unnecessary load to the network and this may be treated by the servers as a Denial-of-Service attack.
- **Status / Solution:**
  The design has been slightly modified to accommodate the above problem by sending HTTP Request Message only once to the intended server while the retransmission value is set to 3 retries if in case the server fails to send any reply.

- **Technical Limitation / Problem:**
  For the web page existence test, the cause of failure in obtaining the web page is still vague as ANFIT will only analyze the result based on the 200 success status code that is coming back to the host.
- **Status / Solution:**
  ANFIT will currently able to cater for the definite reason of failure in obtaining each web page as based on the error codes outlined by the RFC standards. The reply of 3xx series error codes will simply mean a warning of redirection yet not exactly an error while the 4xx series will indicate that there is error from the client (browser) side which user for instance, may need to perform modification to a wrongly spelled URL. The error code of 5xx series will imply that the web page requested is unable to be fulfilled due to the error at the remote web server itself caused by server being temporarily overloaded, experiencing internal error, under maintenance or the service is unavailable.

- **Technical Limitation / Problem:**
  At the initial stage of the design, it is agreed by the group to use the source of Web server's IP Address(es) that is obtained from the Diagnostic Test 4 which the domain name to IP address conversion is performed at the local DNS server. This has resulted in redundancy of diagnosing process since in this stage, ANFIT need to check again on the web server availability at the Transport Layer although it should have been validated at the previous Diagnostic Test 5. Hence, this test will consume more time which can be avoided by careful design and implementation.

- Status / Solution:
  To keep the above problem at bay, the input of Web server IP Address for this diagnostic test has instead being changed from Test 4 to Test 5. ANFIT will now receive only the non-error IP list from the Diagnostic Test 5; which signifies all the IP Addresses in the list have no problem with the TCP connection at the time it was tested previously. This strategy is hoped to expedite the time required to execute this remote server application diagnostic test.

6.2 SECOND STAGE DEVELOPMENT

Second stage development comprised of more detailed diagnostic tests that infer what cause network to fail. This section is supposed to depict in concise the implementation or modification that should take place in respect to the design (Please refer Section 4.9.2). Although this stage is reckoned as the epitome of ANFIT, we unfortunately unable to embark on the development due to the unforeseen circumstances. This stage should be assigned a high priority in the case of further development to be carried out on ANFIT.

6.3 INTERSESSION MESSAGES

As stated in the design (section 4), intersession messages is used to notify the user where the problem lies in the network. The information displayed is a summary of results and decisions collated from the first stage tests. Therefore, it is generated by the Framework with the utilization of Java language.

6.4 ERROR AND WARNING MESSAGES

According to the design, the information to generate error and warning messages should be obtained from second stage tests. However, as we aforementioned, the second stage's call off has caused impact to the implementation of this part. Nevertheless, we still attempted to create error and warning messages for the sake of ensuring the usefulness of ANFIT as well as to fulfil our goals as outlined prior to starting the project. As a consequence, the error and warning messages are contributed from first stage tests.

6.5 INTERESTING AND CHALLENGING ELEMENTS

This section consists of a number of interesting and challenging elements that we encountered as we go along with the project.

6.5.1 FLEXIBLE AND SCALABLE ARCHITECTURE

6.5.1.1 Runtime (How python and java communicate with each other)
A tough task was triggered during the implementation stage which was on how to let Python (diagnostic tests) communicate with Java in order to send the results and information produced by the tests. After a few considerations, it was acknowledged that the only way how Python can communicate with Java is by producing its output and passing it to Java. However, there are two possible ways on how information or output can be passed from Python to Java.

The first method is python can send its output to an intermediate text file. Java will then read the information from the text file. This de-coupling approach is advantageous because there is no direct connection between Python and Java, thus if something happens in Python, it will not have an effect on Java and vice versa. On the contrary, this de-coupling method is certainly not scalable as all Python and Java files will require file reading and writing methods respectively. If more files are involved, more methods will be required and resulting in a messy file handling.

Another alternative is by using Java Runtime class to execute the Python tests. Java will attain the output through obtaining an input stream and read the output directly. The output is parsed by Java and is added to Central Repository for future use. This approach is more convenient as no intermediate files needed, ensuing in no hassle in files handling. Moreover, this approach is very scalable and flexible as all output are parsed and stored in central repository where all other objects can have accessed to it, in order to use the information.

After weighing the reasons, we finally chose the second approach for implementing the interaction between Java and Python.

```
public void run() throws Exception
{

        Process process = Runtime.getRuntime().exec(command);

        BufferedReader br = new BufferedReader(new
        InputStreamReader(process.getInputStream()));
```

6.5.1.2 Central Repository

This is one of the main elements that resides in Framework. It is simple yet powerful component due to its ability to facilitate repository of information produced or obtained throughout the diagnosing process. All the information is readily available hence; whenever it is needed, it can be accessed at once throughout the process. To illustrate this, an example is given below together with the explanations.

Example: `http://www.google.com:80/index.html`

Above is an example of URL that a user can enter when using ANFIT. Framework will trim the URL to suit the tests' needs. Framework trims the input into 3 categories and stores them into Central Repository. The first category is www.google.com, which is stored as the domain name and will be used later in diagnostic test 4. Next category is 80, which is the port number and this is one of the arguments needed by diagnostic test 5 afterwards. Finally, index.html is deposited as the page. This is needed in conducting the diagnostic test 6 later on. All this information is readily available and the tests simply look up for them in Central Repository whenever the need arise.

6.5.1.3 Echo

According to the design, framework can only take one result from each diagnostic test in order to be mapped to the respective decision in the inference table. However, as we go along, we found out that Diagnostic Test 5 (TCP/IP) and Diagnostic Test 6 (HTTP/Web Page) should return two results each for the diagnostic tests to be more meaningful and useful. This has compelled us to find the most appropriate and feasible solution as wrong solution might lead to the framework losing its genericity or causing the diagnostic tests to run unnecessarily. Therefore, after a few thorough considerations and studies, we found out that ECHO command can be used as an ultimate solution.

By using the command, we just have to run the diagnostic test once and pass the second result as an argument to the second test. The second test simply echoes the result without carrying out the test again. For example, Diagnostic Test 5 is comprised of Remote Server IP Connectivity Test (Excerpt 1) and Remote Server TCP Connectivity Test (Excerpt 2). The diagnostic test is executed once but producing two different results. Therefore echo command is put in second test (excerpt 2) so that it can echo the result (send as argument called TCPCONN) produced during the execution of the first test (Excerpt 1).

Excerpt 1

```
 <testname>Remote Server IP Connectivity Test</testname>
             <command>python Test5.py</command>
             <args>PORT IPLIST_LDS</args>
```

Excerpt 2

```
<testname>Remote Server TCP Connectivity Test</testname>
             <command>echo</command>
             <args>TCPCONN</args>
```

6.5.1.4 User Level Direction

As mentioned earlier in this documentation, ANFIT is designed to cater for two different users. This is achieved through framework. When users start to use ANFIT, they will be asked to choose in what level they belong to. This information is then stored in central repository. Errors message is displayed by initializing the Errors.java class. It then looks up the user level in the central repository and reads the corresponding errors' text file. In the potential of more user levels to be introduced, this style of implementation ensures extensibility of the program. Moreover, this implementation can also accommodate the scalability of error messages in future.

```
In Errors.java:

public Errors() {

String user = (String)CentralRepository.get("USER")

      if(user.startsWith("A")){

              fileName = "errorsAdvanced.txt";

              }

              else fileName = "errors.txt" ;
```

6.5.1.5 Results-Decision Mapping Mechanism

Framework adopts expert system approach after obtaining the result from each diagnostic test by mapping result to the corresponding decisions in every problem layer and those will result in decision pairs. Based on these decision pairs, a final decision for every problem layer will be inferred. Referring to the design, ANFIT will choose the problem layer that is concluded as 'PR' or 'LP' for final decision.

In InferTable.java:

```
/* Find the corresponding decision for the result produced by the
test
* All the respective problems are also displayed
*/
public void findProblem(String testname, String result)
{
Test test  = null;
for (int i = 0; i < tests.size(); i++)
{
Test t = (Test)tests.get(i) ;
if(t.getName().equals(testname))
{
test = t ;
break ;
}
}
if(test == null) {
System.out.println("Test "+testname+" not found") ;
return ;
}

ArrayList resultBlocks = test.getResultBlocks() ;
for (int i = 0; i < resultBlocks.size(); i++) {
ResultBlock block = (ResultBlock)resultBlocks.get(i) ;
HashMap map = block.getMap() ;
String decision = (String) map.get(result) ;
if(decision!=null)
{
System.out.println("Decision: "+decision+ " Problem:
"+block.getProblemName()) ;
String probname = block.getProblemName();
Problem prob = getProblem(probname) ;
DecisionPair dp = new DecisionPair(testname,result,decision);
prob.addDecisionPair(dp);
    }
```

6.5.2 COLLECTION OF FAILURE SCENARIOS

Based on the key performance metrics in our objective, we need to gather 20 failure scenarios to fulfil the coverage requirements. It is indeed a challenging task to come out with specific scenarios to reveal the errors yet common enough to be possibly experienced by users as well as easily to be detected. Although we finally managed to list down the failure scenarios, it is during this implementation stage that we acknowledged not all the scenarios can reveal themselves though we put the necessary tests into code.

6.5.3 REALIZATION OF INFERENCE TABLE

Apart from that, to emerge with the inference table is another tough task as proven in the design where a few evolutions have taken place. The implementation has witnessed that this part remains challenging due to complexity of converting the inference table into program. The challenge was also involved in coming up with the XML schema as we need to consider the proper tags, attributes and nesting to use. It was originally implemented with just a few basic tags but as more functionality required, more tags are then added to support the various arguments.

6.5.4 VIOLATION OF RFC STANDARDS

As we go along, we have encountered a few existing implementations in the Internet community that do not conform to the RFC as illustrated below.

6.5.4.1 DNS IMPLEMENTATION

Name Servers Return No Information

- **Standards as outlined in RFC:**
  According to RFC 1034, every zone should have a NS record.

- **Violations:**
  A query on domain name (which is also a zone name) *bbc.net.uk* to its ADS, *ns0.thny.bbc.co.uk*, asking for the NS record returns no answer at all. This issue can be inferred as a violation to RFC since the name server, authoritative for the zone, should have returned a list of its NS records based on the query made.
  (Please refer to log #3)

- **Standards as outlined in RFC:**
  According to the RFC 2038(section 3), the authoritative name servers MUST return SOA record of the zone under the AUTHORITY SECTION if no data of the requested type exists.

- **Violations:**

A query on domain name *www.bbc.net.uk* to *ns0.thny.bbc.co.uk* which is the ADS for zone *bbc.net.uk*, asking for the domain's NS record returns no answer at all. This is not conformed to the RFC, as the authoritative name server is supposed to return SOA record since the requested NS record for this domain does not exist.
(Please refer to log #4)

- **Standards as outlined in the RFC:**
  According to the RFC 1034, the respond from the name servers should be one of the followings:
  1. Answers (positives or negatives)
  2. Referrals (Reference to other name server)
  3. Error conditions (ex:NXDOMAIN,SERVFAIL)

- **Violations:**
  The two examples illustrated above are also best to portray the violations of this standard as they did not return anything useful for the requester as well as leads to undesirable situation where negative caching should have taken place.

Answers return in AUTHORITY SECTION

- **Standards as outlined in the RFC:**
  According to RFC 1034, the name server will return the responses in:
  a. ANSWER SECTION, if it is the authoritative name server of the domain in query.
  b. ANSWER SECTION, if it knows the answer to the query though it is not an authoritative for the domain.
  c. AUTHORITY SECTION, if the responses are of type referral.

Issues arise when some of the name servers return answers in AUTHORITY SECTION.

- **Violations:**
  When queried for domain *.net.my* for NS records, one of the *.my* name server, *NS2.CUHK.EDU.HK* returns answers in the AUTHORITY SECTION instead of ANSWER SECTION as opposed to other *.my* name server (NS.UU.NET) does.
  (Please refer to log #5)

6.5.4.2 HTTP IMPLEMENTATION

- **Standards as outlined in the RFC:**
  According to the RFC, the web servers should return the status line, header lines and the entity body as outlined in the HTTP Response Message Format. The *status line* is the start line which incorporates two functions: to tell the client what version of the protocol the server is using and to communicate a summary of the results of processing the client's request. The *status code* and *reason phrase* provide information about the results of processing the client's request in two different forms. The status code is a three-digit number that indicates the formal result that the server is communicating to the client; it is intended for the client HTTP implementation to

process so the software can take appropriate action. The reason for having both a number and a text string is that computers can more easily "understand" the results of a request by looking at a number, and can then quickly respond accordingly. Humans, on the other hand, find numbers cryptic and text descriptions easier to comprehend

- **Violations:**
  Along the project, we have witnessed some implementation of the web serves that does not comply with the above. They do not reply any status code to indicate whether they are having internal problem or incapable of processing the request. This circumstances which are unaccepted by the protocol leave ANFIT no other alternative but to treat the HTTP connection as fail since the server cannot reply any HTTP status code to the user.
  (Please refer to Log#6)

6.5.5 OTHER INTERESTING ISSUES

Inconsistent Answers From Name Servers

For DNS implementation, according to the theory, all name servers being queried should return same contents of reply if being asked on the same query. Inconsistency exists if the name servers being queried return different answers from each other as well as different number of records, although being asked on the same domain.

Example 1:
Akamai DNS Servers will return 8 randomly chosen Authoritative DNS server(from a number of its' distributed, worldwide network of servers) as answers to the query made.The answers given are different from each other. The inconsistent answers are due to Akamai's enhanced DNS service, which is a global load balancing technique by directing initial requests to the closest server. As for that, this is not treated as error but warning was still flagged to the user.
(Please refer to log #1)

Example 2:
.my name servers return different answers when asked on the domain *.net.my.* Name server NS20.IIJ.AD.JP returns 5 answers but name server, NS3.NIC.FR only gives 4 NS records as the answers when asked on the same domain name.
(Please refer to log #2)

HTTP Redirection Implementation

Apart from the above issues, RFC 2616 also has given insights on how we perceived the implementation of ANFIT should be. As stated in the RFC, it is recommended that a client (browser) can implement a maximum to five redirections in the case of error code 3xx series occurs. Initially, a strategy has been arranged that ANFIT will firstly verify the new URL suggested at the redirection location prior to pointing out this URL to user. Owing to the fact that redirection of URL can take place more than once (potentially of being up to five!); it is therefore essential for ANFIT not to keep on verifying the new URL(s) as this obviously means that ANFIT will need to run the entire tool again to validate the new URL(s). This measure is taken as to avoid the infinite redirection loops, since such loops will cause ANFIT to generate network traffic for each redirection situation. .

No IP Address Associated To Hostname

Another interesting situation that we encountered was to discover that some DNS queries did not recover a matching host name and IP address. This has been further clarified through contacting them via telnet and wget which informed that no IP Address is associated to the hostname. This issue is highlighted realizing that the host name can be used as the user input since most of the browser's implementation does not support automatic appending of a valid URL. For example, if user enters 'harvard.edu', intelligent browser should modify the input into 'www.harvard.edu'. Since, it is further discovered that this domain name belongs to a mail server and not web server, ANFIT will regard this as an error in diagnostic test for DNS. This issue can be further verified in the next development should the scope be expanded to include mail server application.

Chapter 7

TEST PLAN

7.1 UNIT TESTING

Test Objective:

- Unit tests are scripting solutions to simulate user input or system input and written in order to be able to show the common paths of execution and to show how the program behaves.

Test Strategy:

- The initial strategy was to incorporate automated test class for every component in each diagnostic test code by using PyUnit, which is the de facto standard unit-testing framework for Python. The test class will be used in order to test each component in isolation of one another. However, due to unforeseen circumstances and time constraint, our unit testings were performed only by utilizing static input or mock objects where necessary. Mock objects, which still conforms to real object need to be deployed as certain diagnostic tests require pre-conditions and input from other tests as well as relying on them. Furthermore, by using PyUnit, we discovered that the coupling between the unit test and corresponding code is very strong that when the implementation of the code is changed, the unit tests may need to change, thus contributing to a burdensome task. As a consequence, we used the static and mock approach in unit testing with the ideas that as long as the component being tested behaves as expected, it passed the unit test.

Table 7.1: Results of Unit Testing

DIAGNOSTIC TEST	TEST CASES	TYPE OF TEST	METHOD	EXPECTED RESULT	STATUS/ REASON
Test 1: **Configuration Test**	• Check Multiple Interfaces	Failure	Run on Stegosaur that has multiple interfaces	CN	Success
	• Get IP Address	Success	Run on any hosts	Obtained IP Address	Success
	• Check validity of IP Address	Failure	Use various kind of static input of IP Addresses (class D, zero, private, loopback)	NO	Success
	• Get IP Address of Default Gateway	Success	Run on any hosts	Obtained IP Address of Default Gateway	Success

	• Check validity of Default Gateway	Failure	Use static input of IP Addresses	NO	Success
	• Check Default Gateway is in the same IP subnet	Failure	Use different static input IP Addresses for Local Host and Default Gateway (Based on netmask)	NO	Success
	• Get Netmask	Success	Run on any hosts	Obtained Netmask	Success
	• Check validity of Netmask	Failure	Use static input of IP Addresses	NO	Success
	• Check Netmask contiguity	Failure	Use static input of IP Addresses	NO	Success
Test 2: **Default Gateway Connectivity Test**	• Establish connectivity between local host and Default gateway	Failure	Break the connectivity between local host and default gateway by using ipfw and monitor the flow in tcpdump	No 'arp-reply' from default gateway	Unable to conduct
Test 3: **Root DNS Servers Connectivity**	• Establish connectivity between local host and Root DNS Servers	Failure	Break the root DNS servers	NO	Impossible
Test 4: **DNS Consistency Test**	• Check availability of All LDSs	Success	Automate the dig on all LDS (as listed in file etc/resolv.conf)	Every LDS is working	Success
	• Get current LDS's IP Address	Success	Automate dig and get the IP address of the server who returns the answer	Obtained current LDS IP address	Success
	• Get Resolved Name from current LDS	Success	Automate dig to query current LDS and check whether the resource record returned has any CNAME or SOA record. If none of the records exist, domain name remains as the Resolved Name	Obtained Resolved Name	Success

	• Get IP List and ADS List from Current LDS using Resolved Name as query	Success	Automate dig to query current LDS and use Resolve Name as the domain name in query. Get the IP List in the record returned.	Obtained IP List and ADS List	Success
	• Get Resolved Name from ADS	Success	Automate dig to query ADS and check whether the resource record returned has any CNAME or SOA record. If none of the records exist, domain name remains as the Resolved Name	Obtained Resolved Name	Success
	• Get ADS List from ADS using Resolved Name as query	Success	Automate dig to query ADS and use Resolve Name as the domain name in query. Get the IP List in the record returned.	Obtained ADS List	Success
	• Get Resolved Name and ADSs' List from level above DNS Servers (iii)	Success	Automate dig to query on Root DNS server, TLD servers, ADS and get NS record from each using respective domain name of the zone. Obtain resolved Name and ADS List from the ADS.	Obtained Resolved Name and ADS List	Success
	• Check consistency of Resolved Name	Success	Compare Resolved Name returned by LDS, ADS and level above	Consistent Resolved Name	Success
	• Check consistency of answers return by all DNS Servers.	Success	Store All ADS List returned by LDS, ADS and level above respectively. Compare all ADS List with one another.	Consistent ADS List	Success

	• Check consistency of IP List returned by LDS and ADS	Success	Check whether the IP List(s) return by LDS is subset of the one(s) in ADS's record.		
Test 5: **Remote Server TCP/IP Connectivity Test**	• Establish TCP/IP connectivity to web server	Failure	Use static IP Address	'No match' (implicit time out)	Success
Test 6: **Remote Server Application Connectivity Test**	• Check HTTP reply-response connection	Success	Use static IP Address and append to index.html	Status Code	Success
	• Check web page existence	Failure	Use static IP address of yahoo.co.kr	Status Code = 302 (300 Series)	Success
		Failure	Use static web page	Status Code = 404 (400 series)	Success
		Failure	Decrease timeout after establishing TCP connection	Status Code = 500 (500 Series)	Failure - The web server does not return the 500 status code.

7.2 INTEGRATION TESTING

Test Objective:

- Integration testing is conducted in order to test combinations of pieces and to identify problems that occur when all units are combined. Precisely, the integration test is conducted in order to detect any bugs when all diagnostic tests are run together and to verify that the decisions (which were mapped based on the result returned by each diagnostic test) and final decision inferred by framework are correct. Primarily, this test is meant to check the integration between diagnostic tests and framework.

Test Strategy:

- The approach taken to integration testing is by simulating the failure scenarios. The simulation will aid in revealing a specific failure and whether or not the diagnostic tests can detect them and whether or not the framework can determine the respective decision and final decision. In order to do that the appropriate 'URL' is chosen as user input. To conclude whether the integration is successful or not, inter-session messages displayed will match the inference table and the error messages prompted as expected

Test Results:

Table 7.2: Results of Integration Testing

NO	TEST CASES	INPUT	EXPECTED RESULT	STATUS/REASON
1.	Normal scenario	www.ucl.ac.uk/index.html or www.linux.org/index.html	LHP – NP NWP – NP DP – NP RP – NP RSP - NP	Success
2.	Domain name not exist/Invalid domain name [NXDOMAIN]	www.goole234.com/index.html	LHP – UP NWP – UP DP – PR RP – NR RSP - NR	Success
3.	• Inconsistent DNS Answers • Web server has multiple IP Addresses	www.google.com/index.html	LHP – UP NWP – UP DP– NP RP – NP RSP - UP	Success Although DNS servers return inconsistent answers, this is finally treated as NP since it is not an error but ANFIT will flag warning to the user.
4.	No HTTP at the remote server to handle request	tapir.cs.ucl.ac.uk	LHP – UP NWP – UP DP– NP RP – NP RSP - LP	Success
5.	Web page exists but has been redirected to a new location	http://yahoo.co.kr/index.html	LHP – UP NWP – UP DP – NP RP – NP RSP - LP	Success
6.	Invalid web page/web page does not exist	www.linux.org/undox.html	LHP – UP NWP – UP DP– NP RP – NP RSP - LP	Success
7.	DNS Server's internal failure [SERVFAIL]	None	LHP – UP NWP – UP DP– PR RP – NR RSP - NR	Unable to conduct since need to break the DNS server
8.	Operation timed out during DNS look up	www.busan.edu/index.html	LHP – UP NWP – UP DP– LP RP – NR RSP - NR	Success
9.	HTTP request – response connection fails	yahoo.com/index.html	LHP – UP NWP – UP DP– NP RP – NP RSP - LP	Success

10.	Local host configuration problem	Unplug the cable/ Change the configuration of NIC	LHP - PR NWP- NR DP- NR RP - NR RSP - NR	Unable to conduct
11	Local connectivity problem	Use dummynet with ipfw packet filter	LHP - PR NWP- LP DP- NR RP - NR RSP - NR	Unable to conduct

Legend:
LHP - Local Host Problem
NWP - Network Problem
DP - DNS Problem
RP - Reachability Problem
RSP - Remote Server Problem

7.3 USABILITY TESTING

Test Objective:
- To ensure whether ANFIT can serve the intended users in order for them to meet their goals in using the tool. It is testing the software from a users point of view. The performance metrics involve are time, response and accuracy.

Usability Subjects:
- Normal Internet User
- Advanced Network User

Test Strategy:
- Prepare questionnaires based on ANFIT's performance metrics (Refer Appendix F)
- Test 7 subjects in lab - normal internet user
- Interview 3 subjects in lab - advanced network user
- Review the results

Test Results:
Listed below are the results of the usability test conducted.

- **Normal User**

i. Performance metric: Time
This metric is evaluated through asking on how fast ANFIT produce responses to the users. 57% out of total normal Internet user tested, disagree that ANFIT gives fast response. 29% of them were undecided and the remaining 14% were strongly disagree.

ii. Performance metric: Response
This metric is assessed by enquiring whether the users can understand the error message and whether or not they find the error messages useful and helpful. 71% of

total normal Internet user tested, agree that ANFIT gives understandable error message. The remaining 29% disagree with the idea.Moreover, 71% agree the error messages provided are useful and helpful for them. The remaining 29% disagree with the idea.

- **Advanced Network User**

i. <u>Performance metric: Accuracy</u>
This metric is evaluated by asking whether or not the responses of ANFIT similar or close to what they have expected. Based on the interview conducted, it can be said that the response given by ANFIT is similar to what the advanced user has expected. This is proven when we conducted the integration testing.

ii. <u>Performance metric: Coverage</u>
This metric is assessed by enquiring whether or not the failure cases covered by ANFIT sufficient to detect network failure. It can be said that ANFIT is sufficient to detect network failure. This is based on the interview conducted in the lab.

Chapter 8

MANAGEMENT

8.1 PROJECT MANAGEMENT PLAN (EXTREME PROGRAMMING)

The group has decided to use the Extreme Programming (XP) methodology as the right approach for developing the suggested diagnostic tool. Due to its agile and iterative approach, XP was chosen as it allows for the constant refinement of requirements and provides continuous feedback from previous cycles. Due to a vast amount of problems that could occur in a network and system requirements being inherently ill-defined, we had to analyse at least 20 possible failure scenarios which were suggested to diagnose any errors occurring at every layer of the TCP/IP based Internet Model. XP was therefore chosen in order to be able to define the requirements during the design phase of this project. Due to XP's responsiveness to changing requirements the group was provided with the ability to refine requirements and flexibly schedule the implementation of the suggested functionality in any of the following iterations. In terms of meeting time constraints, XP was also found suitable, since it is seen as a good practice to deal with items first that were classified to have a high priority and then move towards the items with the least priority. This therefore meant that in case any time related issues were encountered, the group could neglect low priority items, without affecting the outcome of the project in such a negative way as it would have been the case if high priority objectives were not met.

XP emphasises to adopt the pair-programming approach which suggests that two individuals would normally programme together on one computer. Whilst one team member codes the other one will look out for bugs in the code and give advice on alternatives which might be more efficient. XP therefore benefits all the group members for the reason that system implementation can progress faster and that the code usually contains less errors and is more efficient. A large emphasises is also put on teamwork and confidently responds to changing the customer requirements even late in the life cycle. However, we did not really follow XP all throughout the project. There were some slight modifications made to the guidelines as suggested by the methodology in order to suit our working style and project needs.

8.2 ITERATIONS AND RELEASES

Due to the dynamic nature of the project, we adopt one of the XP strategies, which is to plan the whole project by dividing it into two main releases. There will be three iterations for each release and each iteration will deliver running and useful components. In the first release, we managed to complete the development of framework and all the first stage tests. We evaluate that we already succeed in the first release of the project by meeting our milestone for this release, which is the delivering of intersession messages. As for the second release, we only managed to complete the first iteration as outline in the table 8.1 below. Due to the time constraints and unforeseen circumstances, we only managed to complete partly of the second iteration of this release and it is at this point we had to freeze all further development thus, not being able to complete the third iteration. From the management point of view, we consider that this project is a success though many obstacles arrived along the way. The group and the supervisor agreed collaboratively since the initial stage that it is sufficient to regard ANFIT a success if we managed to accomplish the first release. Therefore, we declare that this project is a success though there are many rooms for improvement.

Table 8.1: Release and Iteration Planning

1 First Release	
Iteration Planning	1.1 Main Tasks
1st	Develop Framework
2nd	Develop Diagnostic Test 1 Develop Diagnostic Test 2 Develop Diagnostic Test 3
3rd	Develop Diagnostic Test 4 Develop Diagnostic Test 5 Develop Diagnostic Test 6
Milestone	Deliver Intersession Message
2 Second Release	
1st	Demonstrate to Supervisor Prepare Documentation Skeleton
2nd	Reconstruct Inference Table Design Second Stage Tests
3rd	Develop Second Stage Tests Perform Proper Testing Write Documentation
Milestone	Deliver Error Message

8.3 ROLES AND RESPONSIBILITIES

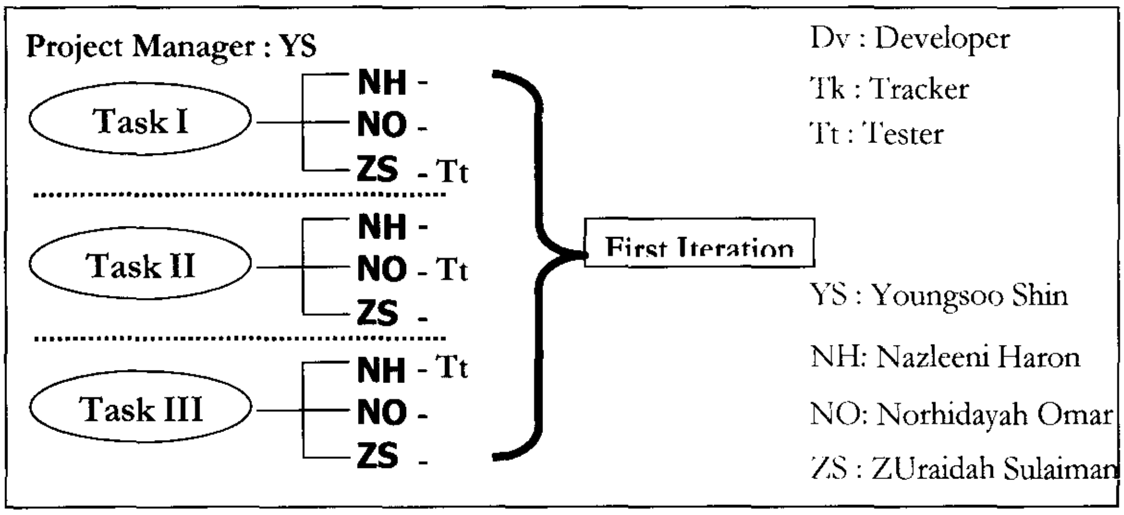

Figure 8.1: Task Allocation Strategy

Youngsoo has been elected as project manager to oversee the progress of the project as well as a coach to mentor the other group members in developing ANFIT. As depicted in the diagram above, for every task in each iteration , the three group members play the role as developer, tester and tracker interchangeably. This approach conforms to eXtreme Programming (XP) method which stresses on team work for higher productivity. We also adopt this strategy as to ensure everyone involves with every part of the development process and to ascertain that everyone acknowledges all facets of the project. This approach is well suited to our project as it aids to complement our limited knowledge and skills in programming.

Chapter 9

MONITORING AND EVALUATION

9.1 PROJECT EVALUATION

This section illustrates all necessary components in order to evaluate our project as a whole. In order to evaluate management strategy, we incorporate methods that we used in monitoring the project as well as managing the risks.

9.1.1 MONITORING AND TRACKING

Monitoring and tracking are indeed another other significant procedures that the group has adopted in order to ensure that the project progress smoothly as planned. The below documents has been produced along the project cycle as a medium to monitor the progress and track every achievements or decisions that have been made among the group members along the way. These documented materials are pertinent to ensure whether we could meet the milestones and project dateline as set forth.

- ***Minutes of Meeting***

Published on web (http://dcndsnetdiagnose.blogspot.com) occasionally after each meeting among the team members during the early stage of project but frequently updated of late; only tracks the weekly meeting with the supervisor. This document tracks the group's decisions that have been achieved collaboratively and outlines the next meeting agendas.

- ***Task and Issue Log***

Recorded in log book daily during the stand-up meetings and it specifies a collection of major tasks of each group member on that specific day or week as well as tracking the issues that has been raised daily. This document also will track the assignments of the programming pairs for each task.

- ***XP Task Card***

Held personal by every group member after stand-up meetings and each member is responsible to update his/her own task card. This "Stick-on Notes" functions as the "To-Do-List" which contains all the specific tasks or operations needs to be completed by every member on every working day.

- ***Progress Report***

Produced after project iteration is completed each time and it tracks all the group's partly achievements. This document is compared to the Gantt Chart regularly as a timely reminder of what we may have left behind schedule.

- ***Stakeholder's Document***

Produced each time when necessary before meeting with supervisor. This document captures in a more organized way of all the things need to be discussed and to seek opinions for. It brings along any related materials, logs, documents or research findings that have been prepared as "homework".

9.1.2 RISK MANAGEMENT

- ***Risk and Impact 1:***

Absence of team members due to unforeseen circumstances

Realizing the fact that this is a group project, the absence of any group members is among the risks that carries a high impact to the project. This will expose the remainder of the group members to undoubtedly great difficulties since each group member has been assigned a specific task and the contributions of each member is counted from day one of the project up to the very last day to ensure the success of the project collectively.

How we handle:

Should the project fail to be tackled as a group of five members, there is indeed a certain urgency in deciding again of the jobs splits and the rescheduling of tasks. The group may also need to remove certain functionalities and features of ANFIT, increase the work load of the remaining members or worse to worst, freeze the development of some parts if faced with the unforeseen circumstances such as member(s) withdraw from the course, fatefully meet with an accident or in the case of death among team members.

- ***Risk and Impact 2:***

Insufficient knowledge or related data useful to the project

It is apparent that Internet is an extremely complicated system. Given this enormous complexity that consists of numerous applications and protocols in between, it is obviously not an easy task to understand in precise how things work independently and comes together in the end. It is very unlikely that the group will be able to grasp all the immense and vast knowledge on the network to the very detailed points, given time constraints. As we go along, it is indeed challenging to sift through the overloaded of information to obtain the useful materials and documents that lies closely to the project. Often, misdirection of resources happens and this lead to misunderstanding of certain concepts.

How we handle:

As to counter the above risk, the great assistance from supervisor and network experts are needed to ensure the group's judgments as a whole are steered to the right direction. Materials related to the project should be directly pointed and shared with every member during stand-up meetings. The opportunity to meet with the supervisor on a weekly basis should be wisely used by all group members to seek out for opinions and discuss on uncertain and unconvinced issues that are arise.

- ***Risk and Impact 3:***

Dynamic requirement and design specification

Due to inherently ill-defined specifications of the project, it is expected for us to face the problem of having requirements and functionality to change recurrently. This project risk will certainly cause us significant impact as it requires modifications to the existing detailed design and sometimes to the already generic, global and workable codes. The alterations will definitely affect our project scope and worsen, delaying the schedule.

How we handle:

In order to address the changing requirements issue, we embrace an XP approach, which is to develop the project incrementally. Apart from that we also negotiate copes and schedules from time to time within the group as well as with our supervisor once a week in order to equip and prepare ourselves for the dynamic changes. As a consequent, the regular meetings aid to detect any requirement or functionality changes at early stage, therefore will lessen the impact to the overall project time and scope. We also carry out unit testing as to make certain that the functionality meets the requirement. If any major changes need to be made that will affect global and generic code, careful considerations on possible alternatives will take place before embarking on the modifications.

- ***Risk and Impact 4:***

Testing environment and related issues

Extensive test series including unit testing, functional testing and integration testing can not be carried out appropriately due to time constraint and unforeseen circumstances. Therefore it will not be possible to ensure the robustness and reliability of the tool and overall performance evaluation of ANFIT can not be given adequately

How we handle:

As the above mentioned risk entails that an overall performance evaluation of ANFIT could not be given, we have decided to design an alternative testing series which only utilizes static values in case of this risk occurring.

- ***Risk and Impact 5:***

Lack of experience with project's platform/environment/methods

Lacking or having limited experience and knowledge with project's platform, environment or methods such as software or hardware is one of the factors that is predicted to reduce our productivity which will hinder us from accomplishing more. This is mainly due to most of the time taken are used in familiarizing ourselves with the platform and environment as well as studying the methods to use.

How we handle:

Since Youngsoo have more and vast experience in dealing with the project's platform and environment, he acted as the mentor to guide the group in deploying only the desirable features of the platform and environment .This alleviate us the hassle to perform thorough study on them, besides contributing to a more effective learning method. Furthermore, we chose python as the development language since the nature of the syntax and its powerful built-in features are suitable and feasible for beginners. Therefore, the strategies carried out facilitate increased productivity.

- ***Risk and Impact 6:***

Miscommunication between team members

This has the impact of not being able to develop the suggested product as expected for a number of reasons; requirements might be poorly understood between team members as they were not discussed properly and therefore the final version of the diagnostic tool might not meet the defined objectives. Other issues that could occur due to the aspect of miscommunication is that the group might fall behind schedule as the they are reluctant to perform in a team and each team member instead decides to work on an individual basis.

How we handle:

This risk can be avoided by utilizing good time management techniques. It is suggested to meet everyday and mostly do the work as a team in the labs. A short meeting in the morning is helpful to decide on the achievements that should be accomplished on a particular day and to resolve any issues that might have occurred the previous day. It is also a good idea to have a stand up meeting once work for a particular day has finished. This serves the purpose of discussing whether the stated goals have been achieved and to keep track of the progress of the project.

9.2 PRODUCT EVALUATION

This section entails our evaluation on ANFIT. The assesment will include Key Performance Indicators (KPI) and critical appraisal, which is further divided into achievements and trade-offs of ANFIT.

9.2.1 KEY PERFORMANCE INDICATOR

As referenced to the objective set forth, these metrics below are evaluated based from the usability testing as well as further assessment by the group.

9.2.1.1 TIME

It was discovered from the usability testing, that ANFIT does not give fast response as expected by the users. Apart from that, we found that ANFIT takes longer than two(2) minutes. These reasons portrayed that ANFIT does meet the performance objective in terms of time, set earlier.

9.2.1.2 RESPONSE

From the result of the usability testing, it can be inferred that ANFIT produces understandable error messages to the user. These error messages are found to be useful and helpful to them.

9.2.1.3 ACCURACY

From the interview conducted in the lab with the advanced network user, it can be concluded that ANFIT produces similar response as expected. This is proven during the integration testing where ANFIT produced similar output as expected for one of the test cases. However, we did not carry out the test for all the test cases. Nevertheless, it is sufficient to conclude that ANFIT meet this performance metric.

9.2.1.4 COVERAGE

It can be concluded that ANFIT meet this performance metric. This is due to the accomplishment of coming out with more than twenty failure scenarios. Another indicator to this is clearly in the interview during the usability testing with advanced network user where the subject found out that ANFIT covers most of the potential network faults that may occur in the network environment.

9.2.2 CRITICAL APPRAISAL

This section highlights a series of evidences why we believe that this project is an achievement. Apart from that, there are also a few trade-offs that we are aware of but willing to sacrifice at the time of the development. These shortcomings are due to time-constraint and unforeseen circumstances.

- ***ANFIT ACHIEVEMENTS***

We consider ANFIT is a success because it manages to provide a user-friendly message. This is clearly demonstrated through explicit division of the error messages. Moreover, the accomplishment is contributed from the ability of ANFIT to educate the normal internet users by imparting the error messages in an understandable, low-level and appropriate language. This is made possible by devising our own error messages. ANFIT facilitates the integration of isolated yet related existing probing tools or applications into one unified diagnostic tool which simplifies the job of network administrator and support team. We believe that both categories of user; normal internet user and advanced user will gain benefit throughout their troubleshooting experiences. Another accomplishment of ANFIT that worth to be pointed out is the reliability of the tool as it covers potential problems that could happen on every layer of TCP /IP Internet model. It is undoubtedly perceived that the existing diagnostic tools that are marketed in the commercial industry, exhibit more elegant and comprehensive features but they are meant for business benefit. On the other hand, ANFIT is developed not for commercial purposes; hence it can be used freely at no cost.

- ***ANFIT TRADE-OFFS***

Upon completion of the project, we realize that there are a few limitations and shortcomings of ANFIT. The most significant one is ANFIT was designed without putting high priority on platform-independent issue. As a result, ANFIT can only be executed on FreeBSD. We realize that this issue has imposed slight difficulty because some of the tools used in ANFIT will return unexpected results. Another drawback is that ANFIT cannot accept IP address as input from the user. It can only accept URL, as the utility in the diagnostic test for DNS fails to resolve if IP address is used. Some of the command line utilities used in ANFIT such as tcpdump and arp, can only be executed by a super user. However, ANFIT does not support the automation of root access privilege. Though we are aware that a host can have multiple interfaces but ANFIT will fail to execute if it were to be deployed in such environment. It is also expected that there will be a slight degradation in the performance of ANFIT, if the host is connected via Point-to-Point (PPP) link. These are the trade-offs that we are willing to sacrifice in order to simplify our design and implementation.

9.3 FUTURE DEVELOPMENT

This section describes the issues that we have not been able to resolve and essential for ANFIT further development.

9.3.1 FRAMEWORK ARCHITECTURE

Framework plays a vital role to materialize this project. As for that, refinements and enhancements could be undertaken in order to boost the capabilities of inferencing. Listed below are ways and approaches to achieve this purpose.

- Improving the data structure that holds tests. In particular, each test should have a set of preconditions, which must be satisfied before the test can be run.
- Improving the XML format. The current XML file is very verbose. Ways to trim the file should be investigated.
- Improving the central hash table, which stores everything output by a program. A possible idea might be to store output only if it starts with the word "STORE". Another idea is to have *chained hashing* so that values are not overwritten but are stored in a *chain* (a list) at the same keyword.
- It would be better to store the errors in XML format rather than in a flat file structure. Instead of running tests sequentially it should be possible to run them all in parallel as separate threads. In this case, if the thread needs a value from another test it should wait until the value becomes available.

9.3.2 WHAT TO DEVELOP IF TIME PERMITS

If we were given more time, there are certain aspects of the project that we would like to improve on. First and foremost, we would refine the design of the detailed diagnostic tests of second stage. The tests that we are supposed to develop in second stage should be able to detect what are the causes of the network problem, thus it is vital to design and develop them properly. In addition, testing t is crucial in order to evaluate the overall performance of ANFIT. If time permits, we will certainly conduct a proper testing. Therefore, robustness and reliability of ANFIT can be evaluated more adequately. As ANFIT is applying one of the techniques in expert system, it is pertinent to have level of certainty for any solution or suggestion given as an indicator for the degree of accuracy of the answer. Consequently, this has to be incorporated in the implementation of ANFIT.

9.3.3 WHAT TO DEVELOP DIFFERENTLY

Apparently, if we are given chance to do the project once more; we will make certain that platform independent issue will be regarded as one of the top priorities. In order to realize this, our implementation will include the functions to download all the associated tools whenever the need arise. Another aspect of the project that we would like to consider is to cater for unstable network conditions. We can accomplish this by running ANFIT a few times before concluding to a final result. As we are aware that the IP Address(es) of the root DNS servers can be changed once in three years, therefore we have to store these IP Address(s) dynamically. It can be achieved by using XML file data structure. Furthermore, we will add more flexibility to the users by allowing them to enter IP address as the input to ANFIT.

9.3.4 WHAT OTHERS COULD CONTINUE

In the case that ANFIT to be extended by other parties, the priority lies in completing the tasks that we did not have time to accomplish as mentioned in Section 9.3.2. Furthermore, it is also encouraged to address the trade-off issues and find the suitable solution in order to make ANFIT really serve its purpose as the diagnostic tool for web service application. The major things that we find a must to include are automation of root access, multiple interfaces, testing issues, second stage tests and level of certainty. In addition, performance-related problems of the network could be considered as one of the problem layers that could be the cause of network failure. For instance, the problems may relate to bandwidth utilization, MTU mismatch and more. Since this is supposed to be a network diagnostic tool, ANFIT should be extended to diagnose other Internet applications such as File Transfer Protocol (FTP), mail service and so on.

Chapter 10

CONCLUSION

The main aims of this project are to build a solution that can assist Internet users in detecting problems why they cannot access certain web sites as well as to enhance knowledge gain by comprehending profoundly on how the normal network behavior actually works in term of Web Service application. We also believe that the suggestions and solutions given must be understandable and workable to users with varying degree of skills.

Our goals were met in most respects. We have developed a network diagnostic tool, which is well suited to detect a failure that may occur in an unstable network condition, called ANFIT for Automated Network Fault Inference Tool. While ANFIT is not the first tool that integrates the related yet isolated network-probing tools into one unified web service tool, it is special because besides inspired by human expert system approach, the automatic diagnosis employs an inference engine which maps the result from the diagnostic tests to the corresponding decision and outputs the final decision based on the flexible inference table. Since ANFIT has been conceived to cater for different type of users, it was realized through the implementation of two types of error messages that will be useful and helpful to the normal Internet user and advanced user respectively.

ANFIT 's design and implementation were significantly originated and based on the failure scenarios which were devised through extensive researches on normal network behaviour and possible causes of failures in network environment. This has undoubtedly increased our knowledge in this area. Though ANFIT does posses a few deficiencies such as platform-dependent, slow response and non-automated of root access but we still regard this project and in particular, the product as a success. This is demonstrated through the number of failure scenarios, which covers all Internet layers, that ANFIT able to identify and reveal.

Throughout this project, we realized that flexible design and implementation is important because we had to deal with the dynamic nature of the network condition. Therefore, not only the diagnostic tests need to be flexible but essentially, the inference engine that collate and process the results should also be extensible and scalable to suit the network growth. Apart from that, another interesting discovery was that some of the existing network implementations do not adhere by the standards outlined by RFC despite many effort has been done in producing such documents.

There are also other interesting issues that we have not been able to resolve and are crucial for further development. This has been illustrated in the project evaluation section of the report. Additionally, it is hope that ANFIT will be further extended to include other web service application to turn it into a more resourceful tool to be deployed in a heterogeneous network environment. Nevertheless, our fervent hope is that ANFIT can serve as a huge community supportive tool due its extensible and flexible nature.

APPENDIX A: **USER MANUAL**

Appendix A: User Manual

Installation:

The first step in the installation process is to download and install all relevant files from the **"source"** directory on any machine running FreeBSD. In case of running ANFIT on any other platform than FreeBSD such as Linux or Solaris the results given might be different as these platforms may employ a different version of the tools mentioned in the system requirements section.

Systems Requirements:

- FreeBSD Machine (Could still run on Solaris or Linux)
- Xerces
- ifconfig
- netstat
- ping
- arp
- tcpdump
- dig
- telnet
- wget

How to run the program:

Once the files have been downloaded, ANFIT is initialized by running the Main.java file in the command line. Once initialized the user is then asked to enter the user level from any of the provided options (normal or advanced). Once this has been accomplished the user is then required to enter the URL of the web pages which is subject to a problem.

APPENDIX B: **ERROR AND WARNING MESSAGES**

APENDIX B: ERROR AND WARNING MESSAGES (FOR NORMAL INTERNET USER)

ERR-DNS0:ANFIT has detected that your primary Local DNS Server($DATA) is not working. Please contact your system administrator!
ERR-DNS1:The host name for the web page($DATA) that you requested does not exist.Please check your spelling and try again.
ERR-DNS2:ANFIT is unable to get the IP Address for web page($DATA) from the Local DNS Server due to the server's internal failure.
ERR-DNS3: Local DNS Server that ANFIT contacted in order to get the IP address for web page($DATA),fails to respond. This may due to the server is unable to handle the request or unable to generate the reply.
ERR-DNS4:ANFIT is unable to get the IP Address for web page from the Remote DNS Server($DATA) due to the server's internal failure.
ERR-DNS5:ANFIT has detected that the Remote DNS Server($DATA) for web page replies with no relevant answer.This may due to incorrect configuration of the server.
ERR-DNS6:Remote DNS Server($DATA) that ANFIT contacted in order to get the IP Address for web page fails to respond. This may due to the server is unable to handle the request or unable to generate the reply.
ERR-DNS7:Remote DNS Server that ANFIT contacted in order to get the IP address for web page($DATA) fails to respond due to the server's internal failure.
ERR-DNS8:Remote DNS Server that ANFIT contacted in order to get the IP Address for web page($DATA) fails to respond. This may due to the server is unable to handle the request or unable to generate the reply.
ERR-DNS9:ANFIT has detected that the Remote DNS Server for web page($DATA) replies with no relevant answer. This may due to incorrect configuration of the server.
ERR-DNS10:ANFIT has detected that the IP Address(es) returns from Local DNS Server($DATA) is not the same as the one(s) from Remote DNS Server.
ERR-DNS11:ANFIT has detected unindentified error with the DNS Server($DATA).
ERR-DNS12:ANFIT has detected unidentified error with the web page($DATA). It may due to no IP Address is associated to the web server that hosts the web page.
ERR-TCP/IP1:Connection refused by the web server($DATA)
ERR-TCP/IP2:Connection timeout maybe because the web server is busy($DATA)
ERR-TCP/IP3:Connection timeout maybe the destination host is unreachable due to power off or it may crashes($DATA)
ERR-HTTP0:ANFIT has not received any respond at all from the server $DATA that hold the web page you are trying to request. The server failed to complete the HTTP connection.
ERR-HTTP1:ANFIT has detected that the page you requested at web server $DATA has permanently moved to other location.
ERR-HTTP2:ANFIT has detected that the page you requested at web server $DATA has temporarily moved to other location.
ERR-HTTP3:ANFIT has detected that the page you requested at web server $DATA has moved to other location.
ERR-HTTP4:ANFIT has detected that the page you requested at web server $DATA must be sent through other URL.
ERR-HTTP5:ANFIT has detected that the page you requested at web server $DATA has temporarily moved to other location. Please try again later with the same URL!
ERR-HTTP6:ANFIT has detected that the page you requested cannot be understood by the server $DATA. Please check your spelling or modify your URL input and try again. Contact web server's administrator if this problem persists!
ERR-HTTP7:ANFIT has detected that the page you requested cannot be understood by the server $DATA as your logon failed. Please check your spelling or modify your URL input and try again. If problem still occurs, please contact the web server's administrator to verify that you have permission to access the requested page!

ERR-HTTP8:ANFIT has detected that the page you requested cannot be understood by the server $DATA as the access is forbidden. Please check your spelling or modify your URL input and try again. Contact the web server's administrator if the problems persists!
ERR-HTTP9:ANFIT has detected that the page you requested cannot be found by the server $DATA. Please check your spelling or modify your URL input to ensure the path is correct. Contact the web server's administrator if this problem persists!
ERR-HTTP10:ANFIT has detected that the remote web server $DATA is experiencing some internal problem that it is now incapable to fulfill your requested page. Please try your request again later. Contact the web server's administrator if this problem persists!
ERR-HTTP11:ANFIT has detected that the remote web server $DATA is experiencing some internal problem that it is now incapable to fulfill your requested page. Please try again later! Contact the web server's administrator if this problem persists!
ERR-HTTP12:ANFIT has detected that the remote web server $DATA is experiencing some internal problem that it is now incapable to fulfill your requested page. Please try again later! Contact the web server's administrator if this problem persists!
ERR-HTTP13:ANFIT has detected that the remote web server $DATA is now temporarily unavailable. Thus, it is incapable to fulfill your requested page due to maintenance or overloading (busy). Please try again later!
ERR-HTTP14:ANFIT has detected that the remote web server $DATA is experiencing some internal problem that it is now incapable to fulfill your requested page. This may due to a network outage or the web site might be experiencing technical difficulties. Please try again later!
ERR-HTTP15:ANFIT has detected that the remote web server $DATA is now incapable to fulfill your requested page as it does not support the version of HTTP protocol indicated in request message of the page.
WARN-DNS1:ANFIT has detected that the Remote DNS Servers for domain($DATA) return inconsistent answers from each other.
WARN-DNS2:ANFIT has detected that the Remote DNS Servers return inconsistent answers for web page($DATA)
WARN-DNS3:ANFIT has detected that the destination host name returns from Local DNS Server($DATA) is different from the one returned by Remote DNS Server.
WARN-DNS4:ANFIT has detected that none of your non-primary Local DNS Server is working.Please contact your network administrator.(The IP Address(es) of the DNS Server(s) is $DATA)
WARN-HTTP1:This is not exactly an error on your host or the web server $DATA that hold the page! The request can still be fulfilled with the new redirection URL.
WARN-HTTP2:This is not exactly an error on your host or the web server $DATA that hold the page! The request can still be fulfilled with the new redirection URL.
WARN-HTTP3:This is not exactly an error on your host or the web server $DATA that hold the page! The request can still be fulfilled with the new redirection URL.
WARN-HTTP4:This is not exactly an error on your host or the web server $DATA that hold the page! The request can still be fulfilled with the new redirection URL.
WARN-HTTP5:This is not exactly an error on your host or the web server $DATA that hold the page! The request can still be fulfilled with the new redirection URL.
WARN-HTTP6:This is not exactly an error. Everything is working just fine!You should not receive this warning unless under experimental conditions!
WARN-HTTP7:This is not exactly an error. Everything is working just fine!You should not receive this warning unless under experimental conditions!

```
WARN-HTTP8:This is not exactly an error. Everything is working just
fine!However if you receive this message, please contact your web server's
administrator!This error message should only be written to your server logs.
WARN-HTTP9:This is not exactly an error. Everything is working just
fine!However if you receive this message, please contact your web server's
administrator!This error message should only be written to your server log
WARN-HTTP10:This is an unidentified error of HTTP connection that nay not be
covered by ANFIT at the moment. ANFIT predicts that there must be a problem
either with your URL input, the remote web server are having problem or
redirection. Please try the later version of ANFIT soon!
```

APENDIX B: ERROR AND WARNING MESSAGES (FOR ADVANCED NETWORK USER)

ERR-DNS0:ANFIT has detected that your primary Local DNS Server is not working.
ERR-DNS1:The host name for the web page($DATA) that you requested does not exist.Please check your spelling and try again.
ERR-DNS2:ANFIT is unable to get the IP Address from the Local DNS Server($DATA) due to the server's internal failure.
ERR-DNS3:The Local DNS Server, that ANFIT contacted to resolve for domain $DATA fails to respond.This may due to the server is unable to handle the request or unable to generate the reply.
ERR-DNS4: The Authoritatice DNS Server($DATA) unable to process the query due to the server's internal failure.
ERR-DNS5:ANFIT has detected that the Authoritative DNS Server($DATA) replies with no relevant answer.This may due to incorrect configuration of the server.
ERR-DNS6:The Authoritative DNS Server($DATA) that ANFIT contacted fails to respond. This may due to the server is unable to handle the request or unable to generate the reply.
ERR-DNS7:The DNS Server for the domain $DATA fails to respond due to internal server failure.
ERR-DNS8:The DNS Server for the domain $DATA fails to respond. This may due to the server unable to handle the request or unable to generate the reply.
ERR-DNS9:ANFIT has detected that the DNS Server for the domain $DATA replies with no relevant answer. This may due to incorrect configuration of the server.
ERR-DNS10:ANFIT has detected that the IP Address(s) returns from Local DNS Server($DATA) is not the same as the one(s) from Authoritative DNS Server.
ERR-DNS11:ANFIT has detected unindentified error with the DNS Server($DATA).
ERR-DNS12:ANFIT has detected unidentified error with the web page($DATA).It may due to no IP Address is associated to the web server.
ERR-TCP/IP1:Connection refused for $DATA maybe due to no http process to handle your request or the web server refused the connection or it maybe busy
ERR-TCP/IP2:Connection time out for $DATA maybe because it is busy
ERR-TCP/IP3:No Match-Implicit time out for $DATA maybe because the Destination Host is unreachable due to power off or it may crashes
ERR-HTTP0:ANFIT has not received any respond at all from the server $DATA that hold the web page you are trying to request. The server may has failed to complete the HTTP connection.
ERR-HTTP1:The page you requested at web server $DATA has moved permanently to other URL (Redirection issue)
ERR-HTTP2:The page you requested at web server $DATA has moved temporarily to other URL (Redirection issue)
ERR-HTTP3:The page you requested at web server $DATA can be found under a preferred alternative URL at present (Redirection issue)
ERR-HTTP4:The page you requested at web server $DATA must be sent through the indicated proxy server (redirection issue)
ERR-HTTP5:The page you requested at web server $DATA has moved temporarily to other URL. Please try again later with the same URL!(Redirection issue)
ERR-HTTP6:The page you requested could not be understood by the web server $DATA due to malformed syntax. Please modify input!Contact web server's administrator if problem persists!(Client error)
ERR-HTTP7:The page you requested at web server $DATA requires user authentication as your logon failed!Please check spelling or modify input and try again.If problem occurs, contct web server's administrator to verify that you have permission to access the requested page!(Client error)
ERR-HTTP8:The page you requested is understood by the web server $DATA but server refused to fulfill the request as the access is forbidden.(Client

error)Please check spelling or modify input and try again.If problem occurs, please contact web administrators!
ERR-HTTP9:The web server $DATA has not found anything matching to the page you requested.Please check spelling or modify URL input to ensure path is correct.If problem occurs, please contact web administrators(Client error)
ERR-HTTP10:The web server $DATA that hold the page is experiencing some internal server error. It may encountered some unexpected condition that prevented it to fulfill the requested page.Please try again later.If problem occurs, please contact web administrators! (Web Server error)
ERR-HTTP11:The web server $DATA does not support the functionality required to fulfill the requested page.Please try again later.If problem occurs, please contact web administrators!(Web Server error)
ERR-HTTP12:The web server $DATA while acting as a gateway or proxy, indicated that it got a bad response from the remote servers.Please try again later.If problem occurs, please contact web administrators!(Web Server error)
ERR-HTTP13:The web server $DATA that hold the page cannot respond due to maintenance or overloading.Please try again later!(Web Server error)
ERR-HTTP14:The web server $DATA while acting as a gateway or proxy, indicated that it did not get a response from the remote server in time.Please try again later!(Web Server error)
ERR-HTTP15:The web server $DATA does not support version of HTTP protocol indicated in request message of the page.(Web Server error)
WARN-DNS1:ANFIT has detected that the Authoritative DNS Servers for web page $DATA return inconsistent answers from each other.This may due to distribution of DNS contents among the servers for load balancing purpose.
WARN-DNS2:ANFIT has detected that the Authoritative DNS Servers for domain $DATA return inconsistent answers from each other.
WARN-DNS3:ANFIT has detected that the destination host name returns from Local DNS Server($DATA) is different from the one returned by Remote DNS Server.
WARN-DNS4: ANFIT has detected that the none of your non-primary Local DNS Server is working.The IP Address(es) of the servers are $DATA
WARN-HTTP1:This is not exactly an error on your host or the web server $DATA that hold the page! The request can still be fulfilled with the new redirection URL
WARN-HTTP2:This is not exactly an error on your host or the web server $DATA that hold the page! The request can still be fulfilled with the new redirection URL
WARN-HTTP3:This is not exactly an error on your host or the web server $DATA that hold the page! The request can still be fulfilled with the new redirection URL
WARN-HTTP4:This is not exactly an error on your host or the web server $DATA that hold the page! The request can still be fulfilled with the new redirection URL
WARN-HTTP5:This is not exactly an error on your host or the web server $DATA that hold the page! The request can still be fulfilled with the new redirection URL
WARN-HTTP6:This is not exactly an error. Everything is working just fine!You should not receive this warning unless under experimental conditions!
WARN-HTTP7:This is not exactly an error. Everything is working just fine!You should not receive this warning unless under experimental conditions!
WARN-HTTP8:This is not exactly an error. Everything is working just fine!However if you receive this message, please contact your web server's administrator!This error message should only be written to your server logs.
WARN-HTTP9:This is not exactly an error. Everything is working just fine!However if you receive this message, please contact your web server's administrator!This error message should only be written to your server log
WARN-HTTP10:This is an unidentified error of HTTP connection that nay not be covered by ANFIT at the moment. ANFIT predicts that there must be a problem

either with your URL input, the remote web server are having problem or redirection. Please try the later version of ANFIT soon!

APPENDIX C: **LIST OF HTTP STATUS CODE**

APPENDIX C: LIST OF HTTP STATUS CODE

Note: Please refer to **RFC 2616** for further description of each status code.

INFORMATIONAL 1XX
100 : Continue
101 : Switching Protocols

SUCCESSFUL 2xx
200 : OK
201 : Created
202 : Accepted
203 : Non-Authoritative Information
204 : No Content
205 : Reset Content
206 : Partial Content

REDIRECTION 3xx
300 : Multiple Choice
301 : Moved Permanently
302 : Found
303 : See Other
304 : Not Modified
305 : Use Proxy
306 : (Unused – the code is reserved)
307 : Temporary Redirect

CLIENT ERROR 4xx

400 : Bad Request
401 : Unauthorized
402 : Payment Required
403 : Forbidden
404 : Not Found
405 : Method Not Allowed
406 : Not Acceptable
407 : Proxy Authentication Required
408 : Request Timeout
409 : Conflict
410 : Gone
411 : Length Required
412 : Preconditioned Failed
413 : Request Entity Too Large
414 : Request-URI Too Long
415 : Unsupported Media Type
416 : Requested Range Not Satisfiable
417 : Expectation Failed

SERVER ERROR 5xx
500 : Internal Server Error
501 : Not Implemanted
502 : Bad Gateway
503 : Service Unavailable
504 : Gateway Timeout
505 : HTTP Version Not Supported

APPENDIX D: **TEST CASES LOG (INTEGRATION TEST)**

APPENDIX D: TEST CASES LOG

Test Case 1:
Normal Scenario

```
Enter User Level:
[N] Normal
[A] Advanced
> A
Enter the URL: www.ucl.ac.uk/index.html
Using domain name: www.ucl.ac.uk
Using port: 80
End is: index.html
----------
Running Configuration Test...
Command: python Test1.py
RESULT OK
IPADDRESS 128.16.66.93
NETMASK 0xfffff000
DEFAULT_GATEWAY 128.16.64.1
Decision: UP Problem: Local Host Problem
----------
Running Default Gateway Connectivity Test...
Command: python sample_arp_ok.py OK 128.16.66.93 128.16.64.1
RESULT OK
Decision: UP Problem: Local Host Problem
Decision: UP Problem: Network Problem
----------
Running Root DNS Servers Connectivity Test...
Command: python Test3.py OK OK
RESULT OK
Decision: UP Problem: Local Host Problem
Decision: UP Problem: Network Problem
----------
Running DNS Consistency Test...
Command: python Test4.py OK OK OK www.ucl.ac.uk
RESULT OK
CURRENT_LDS 128.16.6.8
IPLIST_LDS \['128.40.105.130', '144.82.100.130']
RESOLVED_NAME www.ucl.ac.uk
ADSLIST  ['bas-a.bcc.ac.uk.', 'link-1.ts.bcc.ac.uk.', 'ns1.cs.ucl.ac.uk.',
'ns2.ja.net.']
IPLIST_ADS ['128.40.105.130', '144.82.100.130']
Decision: NP Problem: DNS Problem
----------
Running Remote Server IP Connectivity Test...
Command: python Test5.py 80 \['128.40.105.130', '144.82.100.130']
Command2 exec telnet '128.40.105.130' 80
Command2 exec telnet '144.82.100.130' 80
IPSTRING   '128.40.105.130' '144.82.100.130'
RESULT   OK OK
TCPCONN RESULT   OK OK
OKLIST  ["'128.40.105.130'", "'144.82.100.130'"]
'128.40.105.130'
Result OK
Decision: NR Problem: DNS Problem
Decision: NP Problem: Reachability Problem
'144.82.100.130'
Result OK
Decision: NR Problem: DNS Problem
```

```
Decision: NP Problem: Reachability Problem
----------
Running Remote Server TCP Connectivity Test...
Command: echo RESULT   OK OK
RESULT OK OK
'128.40.105.130'
Result OK
Decision: NR Problem: DNS Problem
Decision: NR Problem: Remote Server Problem
'144.82.100.130'
Result OK
Decision: NR Problem: DNS Problem
Decision: NR Problem: Remote Server Problem
----------
Running Remote Server Application Test(HTTP)...
Command:     python     Test6.py     index.html          ["'128.40.105.130'",
"'144.82.100.130'"]
Command2 exec wget --spider -T 30 -t 3 '128.40.105.130'/'index.html'
'128.40.105.130' HTTP MESSAGE OK BUT URL NOT OK
'128.40.105.130' ERROR CODE: 404 Not Found
Added error ERR-HTTP9_'128.40.105.130'
Command2 exec wget --spider -T 30 -t 3 '144.82.100.130'/'index.html'
'144.82.100.130' HTTP MESSAGE OK BUT URL NOT OK
'144.82.100.130' ERROR CODE: 404 Not Found
Added error ERR-HTTP9_'144.82.100.130'
HTTP SUCCESS
HTTP SUCCESS
RESULT OK
URL FAIL
URL FAIL
URLCONN RESULT NO
Decision: NR Problem: DNS Problem
Decision: UP Problem: Remote Server Problem
----------
Running Remote Server Application Test(URL)...
Command: echo RESULT NO
RESULT NO
Decision: LP Problem: DNS Problem
Decision: LP Problem: Remote Server Problem
==========
Launching 2nd stage of: Remote Server Problem:LP
Please wait...
Local Host Problem : UP
Network Problem : UP
DNS Problem : NP
Reachability Problem : NP
Remote Server Problem : LP
====ERROR RESULTS======
ERR-HTTP9 : The web server '128.40.105.130' has not found anything matching
to the page you requested. (Client error)
ERR-HTTP9 : The web server '144.82.100.130' has not found anything matching
to the page you requested. (Client error)
```

Test Case 2:

```
Domain name notexist/Invalid domain name [NXDOMAIN]
Enter User Level:
[N] Normal
[A] Advanced
> A
Enter the URL: www.goole234.com/index.html
Using domain name: www.goole234.com
Using port: 80
End is: index.html
----------
Running Configuration Test...
Command: python Test1.py
RESULT OK
IPADDRESS 128.16.66.93
NETMASK 0xfffff000
DEFAULT_GATEWAY 128.16.64.1
Decision: UP Problem: Local Host Problem
----------
Running Default Gateway Connectivity Test...
Command: python sample_arp_ok.py OK 128.16.66.93 128.16.64.1
RESULT OK
Decision: UP Problem: Local Host Problem
Decision: UP Problem: Network Problem
----------
Running Root DNS Servers Connectivity Test...
Command: python Test3.py OK OK
RESULT OK
Decision: UP Problem: Local Host Problem
Decision: UP Problem: Network Problem
----------
Running DNS Consistency Test...
Command: python Test4.py OK OK OK www.goole234.com
Added error ERR-DNS1_www.goole234.com
NXDOMAIN
Added error ERR-DNS1_www.goole234.com
RESULT NO
CURRENT_LDS 128.16.6.8
IPLIST_LDS \[]
RESOLVED_NAME www.goole234.com
ADSLIST []
IPLIST_ADS []
Decision: PR Problem: DNS Problem
----------
Running Remote Server IP Connectivity Test...
Command: python Test5.py 80 \[]
RESULT CN
TCPCONN RESULT CN
Result CN
Decision: NR Problem: DNS Problem
Decision: NR Problem: Reachability Problem
----------
Running Remote Server TCP Connectivity Test...
Command: echo RESULT CN
RESULT CN
Result CN
Decision: NR Problem: DNS Problem
Decision: NR Problem: Remote Server Problem
----------
Running Remote Server Application Test(HTTP)...
Command: python Test6.py index.html
RESULT CN
Decision: NR Problem: DNS Problem
Decision: NR Problem: Remote Server Problem
----------
Running Remote Server Application Test(URL)...
Command: echo

==========
Launching 2nd stage of: DNS Problem:PR
Local Host Problem : UP
Network Problem : UP
DNS Problem : PR
Reachability Problem : NR
Remote Server Problem : NR
====ERROR RESULTS======
ERR-DNS1  : The host name for the web page(www.goole234.com) that you
requested does not exist.Please check your spelling and try again.
```

Test Case 3 :
Inconsistent DNS answers and the web servers have multiple IP Addresses

```
Java HotSpot(TM) Client VM warning: Can't detect initial thread stack
location
Enter User Level:
[N] Normal
[A] Advanced
> A
Enter the URL: www.google.com/index.html
Using domain name: www.google.com
Using port: 80
End is: index.html
----------
Running Configuration Test...
Command: python Test1.py
RESULT OK
IPADDRESS 128.16.66.93
NETMASK 0xfffff000
DEFAULT_GATEWAY 128.16.64.1
Decision: UP Problem: Local Host Problem
----------
Running Default Gateway Connectivity Test...
Command: python sample_arp_ok.py OK 128.16.66.93 128.16.64.1
RESULT OK
Decision: UP Problem: Local Host Problem
Decision: UP Problem: Network Problem
----------
Running Root DNS Servers Connectivity Test...
Command: python Test3.py OK OK
RESULT OK
Decision: UP Problem: Local Host Problem
Decision: UP Problem: Network Problem
----------
Running DNS Consistency Test...
Command: python Test4.py OK OK OK www.google.com
Added warning WARN-DNS1_google.akadns.net.
Added warning WARN-DNS2_www.google.com
RESULT MN
CURRENT_LDS 128.16.6.8
IPLIST_LDS \['66.102.11.104', '66.102.11.99']
RESOLVED_NAME www.google.akadns.net.
ADSLIST []
IPLIST_ADS []
Decision: LP Problem: DNS Problem
----------
Running Remote Server IP Connectivity Test...
Command: python Test5.py 80 \['66.102.11.104', '66.102.11.99']
Command2 exec telnet '66.102.11.104' 80
Command2 exec telnet '66.102.11.99' 80
IPSTRING   '66.102.11.104' '66.102.11.99'
```

```
RESULT   OK OK
TCPCONN RESULT   OK OK
OKLIST  ["'66.102.11.104'", "'66.102.11.99'"]
'66.102.11.104'
Result OK
Decision: NR Problem: DNS Problem
Decision: NP Problem: Reachability Problem
'66.102.11.99'
Result OK
Decision: NR Problem: DNS Problem
Decision: NP Problem: Reachability Problem
----------
Running Remote Server TCP Connectivity Test...
Command: echo RESULT   OK OK
RESULT OK OK
'66.102.11.104'
Result OK
Decision: NR Problem: DNS Problem
Decision: NR Problem: Remote Server Problem
'66.102.11.99'
Result OK
Decision: NR Problem: DNS Problem
Decision: NR Problem: Remote Server Problem
----------
Running Remote Server Application Test(HTTP)...
Command: python Test6.py index.html  ["'66.102.11.104'", "'66.102.11.99'"]
Command2 exec wget --spider -T 30 -t 3 '66.102.11.104'/'index.html'
'66.102.11.104' HTTP MESSAGE OK AND URL OK
Command2 exec wget --spider -T 30 -t 3 '66.102.11.99'/'index.html'
'66.102.11.99' HTTP MESSAGE OK AND URL OK
HTTP SUCCESS
HTTP SUCCESS
RESULT OK
URL SUCCESS
URL SUCCESS
URLCONN RESULT OK
Decision: NR Problem: DNS Problem
Decision: UP Problem: Remote Server Problem
----------
Running Remote Server Application Test(URL)...
Command: echo RESULT OK
RESULT OK
Decision: NP Problem: DNS Problem
Decision: UP Problem: Remote Server Problem
==========
Local Host Problem : UP
Network Problem : UP
DNS Problem : NP
Reachability Problem : NP
Remote Server Problem : UP
====ERROR RESULTS======
====WARNING RESULTS======
WARN-DNS1 : ANFIT has detected that the Authoritative DNS Servers for web
page google.akadns.net. return inconsistent answers from each other.This may
due to distribution of DNS contents among the servers for load balancing
purpose.
WARN-DNS2 : ANFIT has detected that the Authoritative DNS Servers for domain
www.google.com return inconsistent answers from each other.
```

Test Case 4:
No HTTP at the remote server to handle request

```
Enter User Level:
[N] Normal
[A] Advanced
> A
Enter the URL: tapir.cs.ucl.ac.uk
Using domain name: tapir.cs.ucl.ac.uk
Using port: 80
End is:
----------
Running Configuration Test...
Command: python Test1.py
RESULT OK
IPADDRESS 128.16.66.93
NETMASK 0xfffff000
DEFAULT_GATEWAY 128.16.64.1
Decision: UP Problem: Local Host Problem
----------
Running Default Gateway Connectivity Test...
Command: python sample_arp_ok.py OK 128.16.66.93 128.16.64.1
RESULT OK
Decision: UP Problem: Local Host Problem
Decision: UP Problem: Network Problem
----------
Running Root DNS Servers Connectivity Test...
Command: python Test3.py OK OK
RESULT OK
Decision: UP Problem: Local Host Problem
Decision: UP Problem: Network Problem
----------
Running DNS Consistency Test...
Command: python Test4.py OK OK OK tapir.cs.ucl.ac.uk
RESULT OK
CURRENT_LDS 128.16.6.8
IPLIST_LDS \['128.16.66.93']
RESOLVED_NAME tapir.cs.ucl.ac.uk
ADSLIST  ['ns0.ja.net.', 'ns1.cs.ucl.ac.uk.', 'sun.mhs-relay.ac.uk.', 'bas-
a.bcc.ac.uk.', 'link-1.ts.bcc.ac.uk.', 'ns2.ja.net.']
IPLIST_ADS ['128.16.66.93']
Decision: NP Problem: DNS Problem
----------
Running Remote Server IP Connectivity Test...
Command: python Test5.py 80 \['128.16.66.93']
Command2 exec telnet '128.16.66.93' 80
Added error ERR-TCP/IP1_'128.16.66.93'
IPSTRING   '128.16.66.93'
RESULT   OK
TCPCONN RESULT   NO
OKLIST  []
'128.16.66.93'
Result OK
Decision: NR Problem: DNS Problem
Decision: NP Problem: Reachability Problem
----------
Running Remote Server TCP Connectivity Test...
Command: echo RESULT   NO
RESULT NO
'128.16.66.93'
Result NO
```

```
Decision: LP Problem: DNS Problem
Decision: LP Problem: Remote Server Problem
----------
Running Remote Server Application Test(HTTP)...
Command: python Test6.py   []
RESULT CN
Decision: NR Problem: DNS Problem
Decision: NR Problem: Remote Server Problem
----------
Running Remote Server Application Test(URL)...
Command: echo

==========
Launching 2nd stage of: Remote Server Problem:LP
Please wait...
Local Host Problem : UP
Network Problem : UP
DNS Problem : NP
Reachability Problem : NP
Remote Server Problem : LP
====ERROR RESULTS======
ERR-TCP/IP1  : Connection  refused  for  '128.16.66.93'  maybe  due  to  no  http
process  to  handle  your  request  or  the  web  server  refused  the  connection  or
it maybe busy
```

Test Case 5:
Web page exists but has been redirected to a new location

```
Enter Level:
[N] Normal
[A] Advanced
> A
Enter the URL: http://yahoo.co.kr/index.html
Using domain name: yahoo.co.kr
Using port: 80
End is: index.html
----------
Running Configuration Test...
Command: python Test1.py
RESULT OK
IPADDRESS 128.16.66.93
NETMASK 0xffff f000
DEFAULT_GATEWAY 128.16.64.1
Decision: UP Problem: Local Host Problem
----------
Running Default Gateway Connectivity Test...
Command: python Test2.py OK 128.16.66.93 128.16.64.1
RESULT OK
Decision: UP Problem: Local Host Problem
Decision: UP Problem: Network Problem
----------
Running Root DNS Servers Connectivity Test...
Command: python Test3.py OK OK
RESULT OK
Decision: UP Problem: Local Host Problem
Decision: UP Problem: Network Problem
----------
Running DNS Consistency Test...
Command: python Test4.py OK OK OK yahoo.co.kr
RESULT OK
CURRENT_LDS 128.16.6.8
```

```
IPLIST_LDS \['202.43.214.151']
RESOLVED_NAME yahoo.co.kr
ADSLIST      ['ns1.yahoo.com.',      'ns2.yahoo.com.',      'ns3.yahoo.com.',
'ns4.yahoo.com.', 'ns5.yahoo.com.', 'ns6.yahoo.com.']
IPLIST_ADS ['202.43.214.151']
Decision: NP Problem: DNS Problem
----------
Running Remote Server IP Connectivity Test...
Command: python Test5.py 80 \['202.43.214.151']
Command2 exec telnet '202.43.214.151' 80
IPSTRING   '202.43.214.151'
RESULT   OK
TCPCONN RESULT   OK
OKLIST  ["'202.43.214.151'"]
'202.43.214.151'
Result OK
Decision: NR Problem: DNS Problem
Decision: NP Problem: Reachability Problem
----------
Running Remote Server TCP Connectivity Test...
Command: echo RESULT   OK
RESULT OK
'202.43.214.151'
Result OK
Decision: NR Problem: DNS Problem
Decision: NR Problem: Remote Server Problem
----------
Running Remote Server Application Test(HTTP)...
Command: python Test6.py index.html  ["'202.43.214.151'"]
Command2 exec wget --spider -T 30 -t 3 '202.43.214.151'/'index.html'
'202.43.214.151' HTTP MESSAGE OK BUT URL NOT OK
'202.43.214.151' ERROR CODE: 302 Found
'202.43.214.151' REDIRECTION LOCATION:
Added error ERR-HTTP2_'202.43.214.151'
Added warning WARN-HTTP2_'202.43.214.151'
HTTP SUCCESS
RESULT OK
URL FAIL
URLCONN RESULT NO
Decision: NR Problem: DNS Problem
Decision: UP Problem: Remote Server Problem
----------
Running Remote Server Application Test(URL)...
Command: echo RESULT NO
RESULT NO
Decision: LP Problem: DNS Problem
Decision: LP Problem: Remote Server Problem
==========
Launching 2nd stage of: Remote Server Problem:LP
Please wait...
Local Host Problem : UP
Network Problem : UP
DNS Problem : NP
Reachability Problem : NP
Remote Server Problem : LP
====ERROR RESULTS======
ERR-HTTP2 : The page you requested at web server '202.43.214.151' has moved
temporarily to other URL (Redirection issue)
====WARNING RESULTS======
WARN-HTTP2 : This is not exactly an error on your host or the web server
'202.43.214.151' that hold the page! The request can still be fulfilled with
the new redirection URL
```

Test Case 6:

```
Invalid web page/web page does not exist

Enter User Level:

[N] Normal

[A] Advanced

> N

Enter the URL: www.linux.org/undox.html

Using domain name: www.linux.org

Using port: 80

End is: undox.html

----------

Running Configuration Test...

Command: python Test1.py

RESULT OK

IPADDRESS 128.16.66.93

NETMASK 0xfffff000

DEFAULT_GATEWAY 128.16.64.1

Decision: UP Problem: Local Host Problem

----------

Running Default Gateway Connectivity Test...
```

```
Command: python sample_arp_ok.py OK 128.16.66.93 128.16.64.1

RESULT OK

Decision: UP Problem: Local Host Problem

Decision: UP Problem: Network Problem

----------

Running Root DNS Servers Connectivity Test...

Command: python Test3.py OK OK

RESULT OK

Decision: UP Problem: Local Host Problem

Decision: UP Problem: Network Problem

----------

Running DNS Consistency Test...

Command: python Test4.py OK OK OK www.linux.org

RESULT OK

CURRENT_LDS 128.16.6.8

IPLIST_LDS \['198.182.196.56']

RESOLVED_NAME www.linux.org

ADSLIST ['ns.invlogic.com.', 'ns0.aitcom.net.']

IPLIST_ADS ['198.182.196.56']

Decision: NP Problem: DNS Problem

----------

Running Remote Server IP Connectivity Test...

Command: python Test5.py 80 \['198.182.196.56']
```

```
Command2 exec telnet '198.182.196.56' 80

IPSTRING   '198.182.196.56'

RESULT   OK

TCPCONN RESULT   OK

OKLIST  ["'198.182.196.56'"]

'198.182.196.56'

Result OK

Decision: NR Problem: DNS Problem

Decision: NP Problem: Reachability Problem

----------

Running Remote Server TCP Connectivity Test...

Command: echo RESULT   OK

RESULT OK

'198.182.196.56'

Result OK

Decision: NR Problem: DNS Problem

Decision: NR Problem: Remote Server Problem

----------

Running Remote Server Application Test(HTTP)...

Command: python Test6.py undox.html  ["'198.182.196.56'"]

Command2 exec wget --spider -T 30 -t 3 '198.182.196.56'/'undox.html'

'198.182.196.56' HTTP MESSAGE OK BUT URL NOT OK

'198.182.196.56' ERROR CODE: 403 Forbidden
```

```
Added error ERR-HTTP8_'198.182.196.56'

HTTP SUCCESS

RESULT OK

URL FAIL

URLCONN RESULT NO

Decision: NR Problem: DNS Problem

Decision: UP Problem: Remote Server Problem

----------

Running Remote Server Application Test(URL)...

Command: echo RESULT NO

RESULT NO

Decision: LP Problem: DNS Problem

Decision: LP Problem: Remote Server Problem

==========

Launching 2nd stage of: Remote Server Problem:LP

Please wait...

Local Host Problem : UP

Network Problem : UP

DNS Problem : NP

Reachability Problem : NP

Remote Server Problem : LP

====ERROR RESULTS======
ERR-HTTP8 : ANFIT has detected that the page you requested cannot be
understood by the server '198.182.196.56' as the access is forbidden. Please
check your spelling or modify your URL input and try again. Contact the web
server's administrator if the problems persists!

====WARNING RESULTS======
```

Test Case 8:

```
Operation timed out during DNS look up

Enter User Level:

[N] Normal

[A] Advanced

> A

Enter the URL: www.busan.edu/index.html

Using domain name: www.busan.edu

Using port: 80

End is: index.html

----------

Running Configuration Test...

Command: python Test1.py

RESULT OK

IPADDRESS 128.16.66.93

NETMASK 0xffff000

DEFAULT_GATEWAY 128.16.64.1

Decision: UP Problem: Local Host Problem

----------
```

```
Running Default Gateway Connectivity Test...

Command: python sample_arp_ok.py OK 128.16.66.93 128.16.64.1

RESULT OK

Decision: UP Problem: Local Host Problem

Decision: UP Problem: Network Problem

----------

Running Root DNS Servers Connectivity Test...

Command: python Test3.py OK OK

RESULT OK

Decision: UP Problem: Local Host Problem

Decision: UP Problem: Network Problem

----------

Running DNS Consistency Test...

Command: python Test4.py OK OK OK www.busan.edu

RESULT MN

Added error ERR-DNS3_www.busan.edu

Decision: LP Problem: DNS Problem

----------

Running Remote Server IP Connectivity Test...

Command: python Test5.py 80

RESULT CN

TCPCONN RESULT CN

Result CN
```

```
Decision: NR Problem: DNS Problem

Decision: NR Problem: Reachability Problem

----------

Running Remote Server TCP Connectivity Test...

Command: echo RESULT CN

RESULT CN

Result CN

Decision: NR Problem: DNS Problem

Decision: NR Problem: Remote Server Problem

----------

Running Remote Server Application Test(HTTP)...

Command: python Test6.py index.html

RESULT CN

Decision: NR Problem: DNS Problem

Decision: NR Problem: Remote Server Problem

----------

Running Remote Server Application Test(URL)...

Command: echo




==========

Launching 2nd stage of: DNS Problem:LP
```

```
Please wait...

Local Host Problem : UP

Network Problem : UP

DNS Problem : LP

Reachability Problem : NR

Remote Server Problem : NR

====ERROR RESULTS======

ERR-DNS3 : The Local DNS Server, that ANFIT contacted to resolve for domain
www.busan.edu fails to respond.This may due to the server is unable to
handle the request or unable to generate the reply.
```

Test Case 9:

HTTP Request - Response connection fails

```
Enter User Level:

[N] Normal

[A] Advanced

> N

Enter the URL: yahoo.com/index.html

Using domain name: yahoo.com

Using port: 80

End is: index.html

----------
```

```
Running Configuration Test...

Command: python Test1.py

RESULT OK

IPADDRESS 128.16.66.93

NETMASK 0xfffff000

DEFAULT_GATEWAY 128.16.64.1

Decision: UP Problem: Local Host Problem

----------

Running Default Gateway Connectivity Test...

Command: python sample_arp_ok.py OK 128.16.66.93 128.16.64.1

RESULT OK

Decision: UP Problem: Local Host Problem

Decision: UP Problem: Network Problem

----------

Running Root DNS Servers Connectivity Test...

Command: python Test3.py OK OK

RESULT OK

Decision: UP Problem: Local Host Problem

Decision: UP Problem: Network Problem

----------

Running DNS Consistency Test...

Command: python Test4.py OK OK OK yahoo.com

RESULT OK
```

```
CURRENT_LDS 128.16.6.8

IPLIST_LDS \['216.109.112.135', '66.94.234.13']

RESOLVED_NAME yahoo.com

ADSLIST      ['ns1.yahoo.com.',      'ns2.yahoo.com.',      'ns3.yahoo.com.',
'ns4.yahoo.com.', 'ns5.yahoo.com.']

IPLIST_ADS ['216.109.112.135', '66.94.234.13']

Decision: NP Problem: DNS Problem

----------

Running Remote Server IP Connectivity Test...

Command: python Test5.py 80 \['216.109.112.135', '66.94.234.13']

Command2 exec telnet '216.109.112.135' 80

Command2 exec telnet '66.94.234.13' 80

IPSTRING   '216.109.112.135' '66.94.234.13'

RESULT   OK OK

TCPCONN RESULT   OK OK

OKLIST  ["'216.109.112.135'", "'66.94.234.13'"]

'216.109.112.135'

Result OK

Decision: NR Problem: DNS Problem

Decision: NP Problem: Reachability Problem

'66.94.234.13'

Result OK

Decision: NR Problem: DNS Problem
```

```
Decision: NP Problem: Reachability Problem

----------

Running Remote Server TCP Connectivity Test...

Command: echo RESULT   OK OK

RESULT OK OK

'216.109.112.135'

Result OK

Decision: NR Problem: DNS Problem

Decision: NR Problem: Remote Server Problem

'66.94.234.13'

Result OK

Decision: NR Problem: DNS Problem

Decision: NR Problem: Remote Server Problem

----------

Running Remote Server Application Test(HTTP)...

Command: python Test6.py index.html  ["'216.109.112.135'", "'66.94.234.13'"]

Command2 exec wget --spider -T 30 -t 3 '216.109.112.135'/'index.html'

'216.109.112.135' HTTP MESSAGE FAIL

Added error ERR-HTTP0_'216.109.112.135'

Command2 exec wget --spider -T 30 -t 3 '66.94.234.13'/'index.html'

'66.94.234.13' HTTP MESSAGE FAIL

Added error ERR-HTTP0_'66.94.234.13'

HTTP FAIL
HTTP FAIL

RESULT NO

URL FAIL

URL FAIL

URLCONN RESULT NO

Decision: LP Problem: DNS Problem

Decision: LP Problem: Remote Server Problem

----------

Running Remote Server Application Test(URL)...

Command: echo RESULT NO

RESULT NO

Decision: LP Problem: DNS Problem

Decision: LP Problem: Remote Server Problem

==========

Launching 2nd stage of: Remote Server Problem:LP

Please wait...

Local Host Problem : UP

Network Problem : UP

DNS Problem : NP

Reachability Problem : NP

Remote Server Problem : LP

====ERROR RESULTS======
ERR-HTTP0 : ANFIT has not received any respond at all from the server
'216.109.112.135' that hold the web page you are trying to request. The
server failed to complete the HTTP connection.

ERR-HTTP0 : ANFIT has not received any respond at all from the server
'66.94.234.13' that hold the web page you are trying to request. The server
failed to complete the HTTP connection.
```

APPENDIX E: **OUTPUT FROM COMMAND LINE UTILITY**

APPENDIX E : Output from Command Line Utility

```
[root@tapir.cs.ucl.ac.uk: u0/nharon 101] ifconfig -a
rl0: flags=8943<UP,BROADCAST,RUNNING,PROMISC,SIMPLEX,MULTICAST> mtu 1500
      inet6 fe80::230:bdff:fe70:c9e3%rl0 prefixlen 64 scopeid 0x1
      ether 00:30:bd:70:c9:e3
      media: Ethernet autoselect (100baseTX <full-duplex>)
      status: active
vr0: flags=8943<UP,BROADCAST,RUNNING,PROMISC,SIMPLEX,MULTICAST> mtu 1500
      inet 128.16.66.93 netmask 0xfffff000 broadcast 128.16.79.255
      inet6 fe80::20e:a6ff:fe81:c27d%vr0 prefixlen 64 scopeid 0x2
      ether 00:0e:a6:81:c2:7d
      media: Ethernet autoselect (100baseTX <full-duplex>)
      status: active
rl1: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
      inet6 fe80::230:bdff:fe70:8ece%rl1 prefixlen 64 scopeid 0x3
      inet 10.0.0.1 netmask 0xffffff00 broadcast 10.0.0.255
      ether 00:30:bd:70:8e:ce
      media: Ethernet autoselect (100baseTX <full-duplex>)
      status: active
sl0: flags=c010<POINTOPOINT,LINK2,MULTICAST> mtu 552
faith0: flags=8002<BROADCAST,MULTICAST> mtu 1500
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 16384
      inet6 ::1 prefixlen 128
      inet6 fe80::1%lo0 prefixlen 64 scopeid 0x6
      inet 127.0.0.1 netmask 0xff000000
ppp0: flags=8010<POINTOPOINT,MULTICAST> mtu 1500
```

```
[root@tapir.cs.ucl.ac.uk: u0/nharon 102] netstat -nr
Routing tables

Internet:
Destination        Gateway            Flags    Refs      Use  Netif Expire
default            128.16.64.1        UGSc       85    10083    vr0
10/24              link#3             UC          2        0    rl1
10.0.0.1           00:30:bd:70:8e:ce  UHLW        0        6    lo0
10.0.0.255         ff:ff:ff:ff:ff:ff  UHLWb       0       17    rl1
127.0.0.1          127.0.0.1          UH          0  3337323    lo0
128.16.64/20       link#2             UC          5        0    vr0
128.16.64.1        00:06:d6:ce:54:06  UHLW       85        1    vr0   1200
128.16.66.93       00:0e:a6:81:c2:7d  UHLW        0       22    lo0
128.16.67.65       00:e0:18:34:15:53  UHLW        0      192    vr0   1125
128.16.79.255      ff:ff:ff:ff:ff:ff  UHLWb       0       17    vr0
```

```
[root@tapir.cs.ucl.ac.uk: u0/nharon 110] dig @ns1.google.com google.com
+retry=7 +time=3 +norec +aa +noad

; <<>> DiG 8.3 <<>> @ns1.google.com google.com +retry=7 +time=3 +norec +aa
+noad
; (1 server found)
;; res options: init aaonly(unimpl) defnam dnsrch
;; got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 30399
;; flags: qr aa; QUERY: 1, ANSWER: 3, AUTHORITY: 4, ADDITIONAL: 4
;; QUERY SECTION:
;;      google.com, type = A, class = IN

;; ANSWER SECTION:
google.com.             5M IN A         216.239.39.99
google.com.             5M IN A         216.239.37.99
google.com.             5M IN A         216.239.57.99

;; AUTHORITY SECTION:
google.com.             4D IN NS        ns1.google.com.
google.com.             4D IN NS        ns2.google.com.
google.com.             4D IN NS        ns3.google.com.
google.com.             4D IN NS        ns4.google.com.

;; Total query time: 144 msec
;; FROM: tapir.cs.ucl.ac.uk to SERVER: 216.239.32.10
;; WHEN: Sat Sep  4 16:53:28 2004
;; MSG SIZE  sent: 28  rcvd: 212
```

```
[root@tapir.cs.ucl.ac.uk: u0/nharon 105] tcpdump -n -l arp
tcpdump: WARNING: rl0: no IPv4 address assigned
tcpdump: listening on rl0
16:42:05.687309 arp who-has 128.16.67.3 tell 128.16.67.69
16:42:06.185172 arp who-has 128.16.67.147 tell 128.16.64.1
16:42:06.686575 arp who-has 128.16.67.3 tell 128.16.67.69
16:42:07.057225 arp who-has 128.16.65.34 tell 128.16.64.1
16:42:07.499072 arp who-has 128.16.64.165 tell 128.16.67.12
16:42:07.686368 arp who-has 128.16.67.3 tell 128.16.67.69
16:42:11.498993 arp who-has 128.16.64.165 tell 128.16.67.12
16:42:11.855758 arp who-has 128.16.64.28 tell 128.16.64.1
16:42:12.271268 arp who-has 128.16.68.16 tell 128.16.64.1
16:42:13.076766 arp who-has 128.16.65.34 tell 128.16.64.1
16:42:14.292514 arp who-has 128.16.64.28 tell 128.16.64.1
16:42:15.498929 arp who-has 128.16.64.165 tell 128.16.67.12
16:42:19.125028 arp who-has 128.16.64.168 tell 128.16.67.13
16:42:19.395551 arp who-has 128.16.5.136 tell 128.16.10.130
16:42:19.498961 arp who-has 128.16.64.165 tell 128.16.67.12
```

```
[root@tapir.cs.ucl.ac.uk: u0/nharon 106] ping -c 1 128.16.64.1
PING 128.16.64.1 (128.16.64.1): 56 data bytes
64 bytes from 128.16.64.1: icmp_seq=0 ttl=255 time=0.321 ms

--- 128.16.64.1 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.321/0.321/0.321/0.000 ms
```

```
[root@tapir.cs.ucl.ac.uk: u0/nharon 107] telnet 144.82.100.130 80
Trying 144.82.100.130...
Connected to wwws-a.ucl.ac.uk.
Escape character is '^]'.
^]
telnet> close
Connection closed.
```

```
[root@tapir.cs.ucl.ac.uk:  u0/nharon  110] exec wget --spider -T 30 -t 3
198.182.196.56/index.html
--09:05:02--  http://198.182.196.56:80/index.html
           => `index.html'
Connecting to 198.182.196.56:80... connected!
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/html]
200 OK
```

```
[zsulaima@tapir.cs.ucl.ac.uk:   u0/zsulaima   101]   exec   wget   --spider
www.yahoo.co.kr -T 30 -t
--05:14:37--  http://www.yahoo.co.kr/
           => `index.html.4'
Resolving www.yahoo.co.kr... done.
Connecting to www.yahoo.co.kr[202.43.214.151]:80... connected.
HTTP request sent, awaiting response... 302 Found
Location: http://kr.yahoo.com/ [following]
--05:14:38--  http://kr.yahoo.com/
           => `index.html.4'
Resolving kr.yahoo.com... done.
Connecting to kr.yahoo.com[202.43.214.190]:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/html]
200 OK
```

APPENDIX F: **QUESTIONNAIRE (USABILITY TESTING)**

USABILITY TEST QUESTIONNAIRES

1. How do you consider yourself in the computer networking area?

   ☐ Normal Internet User

   ☐ Advanced Internet User(Network Administrator)

2. Are you a computer science student or having any computer background?

   - ☐ Yes
   - ☐ No

3. How often do you use internet?

   - ☐ Never
   - ☐ Once or more a month
   - ☐ Once or more a week
   - ☐ Once or more a day
   - ☐ Once a week

4. ANFIT gives fast response as you have expected?

   - ☐ Strongly Agree
   - ☐ Agree
   - ☐ Unsure
   - ☐ Disagree
   - ☐ Strongly disagree

5. ANFIT gives understandable error meassages as you have expected?

   - ☐ Strongly Agree
   - ☐ Agree
   - ☐ Unsure
   - ☐ Disagree
   - ☐ Strongly disagree

6. The error messages are useful and helpful to aid you in making the next decision (continue troubleshoot or do nothing on your side)?

   - ☐ Strongly Agree
   - ☐ Agree
   - ☐ Unsure
   - ☐ Disagree
   - ☐ Strongly disagree

7. ANFIT is more helpful and user friendly to you than human expert in poiting to you the network problem and state why you prefer one not the other?

   Reasons:

   ☐ Strongly Agree --

- ☐ Agree --

- ☐ Unsure --

- ☐ Disagree --

- ☐ Strongly disagree --

----

8. Please give any comments that you have about your <u>overall</u> experience using ANFIT

THANK YOU VERY MUCH FOR YOUR PRECIOUS TIME SPENT TO COMPLETE THESE QUESTIONNAIRE! ☺

DNS Implementations Issues

```
Issue 1: Inconsistent Answers from Name Servers
Log #1
Akamai dns servers will return 8 randomly chosen Authoritative DNS as
Answers and the Answers given are inconsistent with each other. from each
other.

[nharon@tapir.cs.ucl.ac.uk: z15_project/Framework 112] !110 +noad
```

```
dig @asia3.akam.net. google.akadns.net. ns +retry=7 +time=3 +norec +noad

; <<>> DiG 8.3 <<>> @asia3.akam.net. google.akadns.net. ns +retry=7 +time=3
+norec +noad
; (1 server found)
;; res options: init defnam dnsrch
;; got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 38065
;; flags: qr aa; QUERY: 1, ANSWER: 8, AUTHORITY: 0, ADDITIONAL: 8
;; QUERY SECTION:
;;      google.akadns.net, type = NS, class = IN

;; ANSWER SECTION:
google.akadns.net.      1d1h IN NS      use3.akam.net.
google.akadns.net.      1d1h IN NS      usw2.akam.net.
google.akadns.net.      1d1h IN NS      use4.akam.net.
google.akadns.net.      1d1h IN NS      usw5.akam.net.
google.akadns.net.      1d1h IN NS      usw7.akam.net.
google.akadns.net.      1d1h IN NS      zh.akadns.net.
google.akadns.net.      1d1h IN NS      zf.akadns.net.
google.akadns.net.      1d1h IN NS      use1.akam.net.

;; Total query time: 248 msec
;; FROM: tapir.cs.ucl.ac.uk to SERVER: 193.108.154.9
;; WHEN: Tue Jul 27 18:10:32 2004
;; MSG SIZE  sent: 35  rcvd: 326


[nharon@tapir.cs.ucl.ac.uk: z15_project/Framework 111] dig @eur3.akam.net.
google.akadns.net. ns +retry=7 +time=3 +norec +aa +noad

; <<>> DiG 8.3 <<>> @eur3.akam.net. google.akadns.net. ns +retry=7 +time=3
+norec +aa +noad
; (1 server found)
;; res options: init aaonly(unimpl) defnam dnsrch
;; got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 54084
;; flags: qr aa; QUERY: 1, ANSWER: 8, AUTHORITY: 0, ADDITIONAL: 8
;; QUERY SECTION:
;;      google.akadns.net, type = NS, class = IN

;; ANSWER SECTION:
google.akadns.net.      1d1h IN NS      usw7.akam.net.
google.akadns.net.      1d1h IN NS      zc.akadns.net.
google.akadns.net.      1d1h IN NS      eur3.akam.net.
google.akadns.net.      1d1h IN NS      zb.akadns.net.
google.akadns.net.      1d1h IN NS      ze.akadns.net.
google.akadns.net.      1d1h IN NS      usw6.akam.net.
google.akadns.net.      1d1h IN NS      zc.akadns.org.
google.akadns.net.      1d1h IN NS      use1.akam.net.

;; Total query time: 19 msec
;; FROM: tapir.cs.ucl.ac.uk to SERVER: 193.45.1.103
;; WHEN: Tue Jul 27 18:08:37 2004
;; MSG SIZE  sent: 35  rcvd: 332


-----------------------------------------------------------------------
Log #2
.my name servers return different number of records as answers although
being asked on the same domain with the same type of query.
```

```
[younshin@tapir.cs.ucl.ac.uk: younshin/project 111] dig @NS20.IIJ.AD.JP.
net.my. ns +retry=7 +time=3 +norec +aa +noad

; <<>> DiG 8.3 <<>> @NS20.IIJ.AD.JP. net.my. ns +retry=7 +time=3 +norec +aa
+noad
; (1 server found)
;; res options: init aaonly(unimpl) defnam dnsrch
;; got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 23115
;; flags: qr aa; QUERY: 1, ANSWER: 5, AUTHORITY: 0, ADDITIONAL: 0
;; QUERY SECTION:
;;      net.my, type = NS, class = IN

;; ANSWER SECTION:
net.my.                 1D IN NS        gate1.jaring.my.
net.my.                 1D IN NS        ns5.jaring.my.
net.my.                 1D IN NS        ns6.jaring.my.
net.my.                 1D IN NS        dns1.mynic.net.my.
net.my.                 1D IN NS        ns20.iij.ad.jp.

;; Total query time: 246 msec
;; FROM: tapir.cs.ucl.ac.uk to SERVER: 202.232.2.161
;; WHEN: Tue Jul 27 00:36:22 2004
;; MSG SIZE  sent: 24  rcvd: 140


[younshin@tapir.cs.ucl.ac.uk: younshin/project 112] dig @NS3.NIC.FR. net.my.
ns +retry=7 +time=3 +norec +aa +noad

; <<>> DiG 8.3 <<>> @NS3.NIC.FR. net.my. ns +retry=7 +time=3 +norec +aa
+noad
; (2 servers found)
;; res options: init aaonly(unimpl) defnam dnsrch
;; got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 26216
;; flags: qr; QUERY: 1, ANSWER: 0, AUTHORITY: 4, ADDITIONAL: 0
;; QUERY SECTION:
;;      net.my, type = NS, class = IN

;; AUTHORITY SECTION:
net.my.                 1D IN NS        ns5.jaring.my.
net.my.                 1D IN NS        ns6.jaring.my.
net.my.                 1D IN NS        dns1.mynic.net.my.
net.my.                 1D IN NS        gate1.jaring.my.

;; Total query time: 10 msec
;; FROM: tapir.cs.ucl.ac.uk to SERVER: 192.134.0.49
;; WHEN: Tue Jul 27 00:36:50 2004
;; MSG SIZE  sent: 24  rcvd: 112

=============================================================================
==
Issue 2: Name Servers Return No Information

Log #3
Authoritative DNS(ADS) should return ADS list as ANSWERS if the query made
is using zone name(bbc.net.uk).

[nharon@tapir.cs.ucl.ac.uk: tapir/u0 105] dig @ns0.thny.bbc.co.uk.
bbc.net.uk ns +retry=7 +time=3 +norec +aa +noad
```

```
; <<>> DiG 8.3 <<>> @ns0.thny.bbc.co.uk. bbc.net.uk ns +retry=7 +time=3
+norec +aa +noad
; (1 server found)
;; res options: init aaonly(unimpl) defnam dnsrch
;; got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 57188
;; flags: qr aa; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 0
;; QUERY SECTION:
;;      bbc.net.uk, type = NS, class = IN

;; Total query time: 72 msec
;; FROM: tapir.cs.ucl.ac.uk to SERVER: 212.58.240.20
;; WHEN: Tue Jul 27 21:00:40 2004
;; MSG SIZE  sent: 28  rcvd: 28

[nharon@tapir.cs.ucl.ac.uk: tapir/u0 107] dig @ns1.primarydns.com.
linux.net ns +retry=7 +time=3 +norec +aa +noad

; <<>> DiG 8.3 <<>> @ns1.primarydns.com. linux.net ns +retry=7 +time=3
+norec +aa +noad
; (1 server found)
;; res options: init aaonly(unimpl) defnam dnsrch
;; got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 45643
;; flags: qr aa ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 2
;; QUERY SECTION:
;;      linux.net, type = NS, class = IN

;; ANSWER SECTION:
linux.net.              1H IN NS        ns1.primarydns.com.
linux.net.              1H IN NS        ns2.primarydns.com.

;; Total query time: 103 msec
;; FROM: tapir.cs.ucl.ac.uk to SERVER: 216.219.239.7
;; WHEN: Tue Jul 27 21:02:59 2004
;; MSG SIZE  sent: 27  rcvd: 109
--------------------------------------------------------------------------
```

Log #4
Authoritative DNS server (ADS) should return SOA if the query made is using domain name(www.bbc.net.uk).

```
[nharon@tapir.cs.ucl.ac.uk: z15_project/Framework 117] dig
@ns0.thny.bbc.co.uk. www.bbc.net.uk ns +retry=7 +time=3 +norec +aa +noad

; <<>> DiG 8.3 <<>> @ns0.thny.bbc.co.uk. www.bbc.net.uk ns +retry=7 +time=3
+norec +aa +noad
; (1 server found)
;; res options: init aaonly(unimpl) defnam dnsrch
;; got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 625
;; flags: qr aa; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 0
;; QUERY SECTION:
;;      www.bbc.net.uk, type = NS, class = IN

;; Total query time: 72 msec
;; FROM: tapir.cs.ucl.ac.uk to SERVER: 212.58.240.20
;; WHEN: Tue Jul 27 18:22:29 2004
;; MSG SIZE  sent: 32  rcvd: 32
```

```
[nharon@tapir.cs.ucl.ac.uk: tapir/u0 104] dig @ns1.primarydns.com.
www.linux.net ns +retry=7 +time=3 +norec +aa +noad

; <<>> DiG 8.3 <<>> @ns1.primarydns.com. www.linux.net ns +retry=7 +time=3
+norec +aa +noad
; (1 server found)
;; res options: init aaonly(unimpl) defnam dnsrch
;; got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 12907
;; flags: qr aa ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 0
;; QUERY SECTION:
;;      www.linux.net, type = NS, class = IN

;; AUTHORITY SECTION:
linux.net.              1H IN SOA       ns1.primarydns.com.
hostmaster.primarydns.com. (
                                        200407270       ; serial
                                        3H              ; refresh
                                        1H              ; retry
                                        6H              ; expiry
                                        1H )            ; minimum


;; Total query time: 109 msec
;; FROM: tapir.cs.ucl.ac.uk to SERVER: 216.219.239.7
;; WHEN: Tue Jul 27 20:56:51 2004
;; MSG SIZE  sent: 31  rcvd: 96



===========================================================================
Issue 3: Answers return in AUTHORITY SECTION

Log #5
.my name server NS2.CUHK.EDU.HK. returns answer in AUTHORITY
SECTION,although it is supposed to return them in ANSWER SECTION
as what the other .my name server, @NS.UU.NET. does.

[younshin@tapir.cs.ucl.ac.uk: u0/younshin 101] dig @NS2.CUHK.EDU.HK. net.my.
ns +retry=7 +time=3 +norec +aa

; <<>> DiG 8.3 <<>> @NS2.CUHK.EDU.HK. net.my. ns +retry=7 +time=3 +norec
+aa
; (1 server found)
;; res options: init aaonly(unimpl) defnam dnsrch
;; got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 60953
;; flags: qr; QUERY: 1, ANSWER: 0, AUTHORITY: 4, ADDITIONAL: 0
;; QUERY SECTION:
;;      net.my, type = NS, class = IN

;; AUTHORITY SECTION:
net.my.                 1D IN NS        ns5.jaring.my.
net.my.                 1D IN NS        ns6.jaring.my.
net.my.                 1D IN NS        dns1.mynic.net.my.
net.my.                 1D IN NS        gate1.jaring.my.

;; Total query time: 275 msec
;; FROM: tapir.cs.ucl.ac.uk to SERVER: 137.189.6.21
;; WHEN: Mon Jul 26 20:06:07 2004
;; MSG SIZE  sent: 24  rcvd: 112
```

```
[younshin@tapir.cs.ucl.ac.uk: u0/younshin 102] dig @NS.UU.NET. net.my. ns
+retry=7 +time=3 +norec +aa

; <<>> DiG 8.3 <<>> @NS.UU.NET. net.my. ns +retry=7 +time=3 +norec +aa
; (1 server found)
;; res options: init aaonly(unimpl) defnam dnsrch
;; got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 20886
;; flags: qr; QUERY: 1, ANSWER: 4, AUTHORITY: 0, ADDITIONAL: 0
;; QUERY SECTION:
;;      net.my, type = NS, class = IN

;; ANSWER SECTION:
net.my.                 1D IN NS        ns5.jaring.my.
net.my.                 1D IN NS        ns6.jaring.my.
net.my.                 1D IN NS        dns1.mynic.net.my.
net.my.                 1D IN NS        gate1.jaring.my.

;; Total query time: 86 msec
;; FROM: tapir.cs.ucl.ac.uk to SERVER: 137.39.1.3
;; WHEN: Mon Jul 26 20:06:38 2004
;; MSG SIZE  sent: 24  rcvd: 112

=======================================================================
```

HTTP Implementation Issues

Issue : Web server does not reply any status code

Log #6
yahoo.com does not return status code

```
[guadalupe: nharon/digdesign 7] wget --spider 66.94.231.99/index.html
--11:07:14--  http://66.94.231.99:80/index.html
           => `index.html'
Connecting to 66.94.231.99:80... connected!
HTTP request sent, awaiting response...
11:07:14 ERROR -1: Malformed status line.

-----------------------------------------------------------------------
```