

MINT 709 Capstone Project

Project: Implement virtual Data sharing application on WAN using DDNS and its performance investigation.



University of Alberta
Edmonton, Canada
Project adviser: Dr Mike MacGregor

Submitted by
Ashishkumar Patel

Project Reader
Dr. Mike MacGregor

Abstract

In this report I explain the solution of problem faced by the Datagardens Company. They have designed virtual software which is used to move virtual Database server from one network segment to other network segment in a WAN environment.

The virtual software is working fine, they are able to move Virtual Database server from one subnet of network to another but the real problem is once it is moved to another subnet, Clients who are already connected to the Database server or new clients do not know about the new IP address and domain name of the moved virtual Database server. There is no mechanism for the clients to reconnect to a new IP address of the Database server. Here the solution of problem is implemented by adding some new software and hardware to the existing network. I used a BIND server, DHCP server and the concept of Dynamic DNS which is used to send dynamic updates to the BIND server. Through out this project report you will get a basic understanding, installation, and configuration of the BIND server, DHCP server and DDNS. You will see in-depth detail about the solution of this problem and its performance investigation.

Acknowledgement

I am very thankful to Dr M.H. MacGregor for giving me an opportunity to work on this project. Without his thorough guidance and tremendous help, I would not have been able to achieve the goal of this project. I also would like to thank him for his contribution to the MINT program, which brings us a whole new world of computer networks.

A special thank also goes to Shahnawaz Mir for providing very flexible timings to access the MINT lab and technical guidance.

I am also thankful to the Datagardens Company for giving me the opportunity to demonstrate the project and to show an interest in the implemented solution.

Table of Contents

Chapter 1 Project Introduction.....	07
1.1 Project Subject.....	07
1.2 Project Solution.....	07
Chapter 2 DNS.....	09
2.1 History of DNS.....	09
2.2 Introduction of DNS.....	09
2.3 Domain Namespace.....	10
2.4 Domain Names and Domains.....	10
2.4.1 Domain Names.....	11
2.4.2 Domains.....	11
2.5 Resource Record.....	13
2.6 Internet Domain Namespace.....	13
2.6.1 Hierarchy of Domain Names.....	13
2.6.2 Top Level Domains.....	14
2.6.3 Country-Code Top-level Domains.....	15
2.6.4 New Top-level Domains.....	15
2.7 Delegation.....	15
2.8 Nameserver.....	16
2.8.1 Types of Nameserver.....	16
2.8.1.1 Authoritative Nameserver.....	16
2.8.1.2 The Primary Master.....	17
2.8.1.3 Slave Server.....	17
2.8.1.4 Caching Nameserver.....	17
2.8.1.5 Multiple or Hybrid Nameserver.....	17
2.9 Zones.....	17
2.10 DNS Resolver.....	19
2.11 Resolution and Recursion.....	19
2.12 Root Nameserver.....	19
2.13 Caching.....	20
2.14 TTL.....	20
Chapter 3 BIND.....	21
3.1 Introduction to BIND.....	21
3.2 Installation of BIND.....	22
3.2.1 Named.conf Default Configuration File.....	23
3.3 The Zone Datafiles.....	24
3.3.1 Zone File Directives.....	24
3.3.1.1 \$INCLUDE.....	24
3.3.1.2 \$ORIGIN.....	25
3.3.1.3 \$TTL.....	25
3.3.2 Comments.....	26
3.3.3 Zone File Resource Records.....	26
3.3.3.1 SOA Record.....	26
3.3.3.2 NS Record.....	28
3.3.3.3 A Record.....	28
3.3.3.4 PTR Record.....	28

3.3.3.5 CNAME.....	28
3.4 Configuration of BIND Server.....	29
3.4.1 Caching Nameserver.....	29
3.4.2 Primary Master.....	30
3.4.2.1 Forward Zone File.....	30
3.4.2.2 Reverse Zone File.....	31
3.4.3 Secondary Master.....	32
3.5 Logging.....	33
3.5.1 Logging Statement Grammar.....	33
3.6 Parenting.....	35
3.6.1 Domain Configuration for Parenting.....	35
3.6.2 Configuration For subdomain.....	36
3.7 DNS Features.....	37
3.7.1 Control and rndc.....	37
3.7.2 DDNS.....	39
3.7.2.1 Dynamic Update and Serial Numbers.....	39
3.7.2.2 Dynamic Update and The journal file.....	39
3.7.2.3 BIND Configuration for Dynamic Updates.....	40
3.7.3 DNS Notify.....	41
3.7.3.1 also-notify.....	41
3.7.3.2 allow-notify.....	41
3.7.4 Incremental Zone Transfer (IXFR).....	42
3.7.4.1 IXFR Configuration.....	43
3.7.5 TSIG.....	43
3.7.5.1 Automatic Generation of TSIG.....	43
3.7.5.2 Manual Generation.....	44
3.7.5.3 Configuring TSIG.....	44
3.8 Diagnostic Tools.....	44
3.8.1 nslookup.....	44
3.8.2 dig.....	46
3.8.3 host.....	48
3.9 Administrative Tools.....	48
3.9.1 named-checkconf.....	48
3.9.2 named-checkzone.....	48
Chapter 4 DHCP3 server.....	50
4.1 Why DHCP?.....	50
4.2 DHCP Operation.....	50
4.3 Installation of DHCP Server.....	51
4.4 Configuration of DHCP Server.....	52
4.4.1 Authoritative Configuration.....	52
4.4.2 Individual Subnet Configuration.....	52
4.4.3 Address Allocation.....	52
4.4.4 Client Option Information.....	53
4.4.4.1 DNS Option.....	53
4.4.4.2 Router Option.....	53
4.4.4.3 Broadcast Option.....	53

4.4.4.4 Subnet mask Option.....	53
4.4.5 The Server-identifier Statement.....	53
4.4.6 The Default-lease-time Statement.....	54
4.4.7 The Max-lease-time Statement.....	54
4.4.8 The Min-lease-time Statement.....	54
4.4.9 The ddns-domainname Statement.....	54
4.4.10 The ddns-rev-domainname Statement.....	54
4.4.11 DDNS-update-style.....	55
4.4.12 The ddns-ttl Statement.....	55
4.4.13 NTP Server.....	55
4.4.14 Mixing Static and Dynamic Allocation.....	55
4.5 DHCP Messages.....	55
4.5.1 The DHCPDISCOVER Message.....	56
4.5.2 The DHCPOFFER Message.....	56
4.5.3 The DHCPREQUEST Message.....	56
4.5.4 The DHCPACK Message.....	56
4.6 DHCP for DDNS update.....	57
4.6.1 How FQDN formed?.....	57
4.6.2 DNS Update Security.....	58
4.6.3 Hostnames Configuration in Client.....	58
4.6.4 Configuring the Servers for Updates.....	59
4.6.4.1 Configuring the DHCP Server to Do Updates.....	59
4.6.4.2 DNS Server Configuration to Allow Dynamic Updates.....	59
4.7 DNS Record Removal.....	60
4.8 Example of dhcpcd.conf Configuration File with DDNS	60
Chapter 5 Project Design and Implementation.....	62
5.1 Introduction.....	62
5.2 Hardware and Software Used to Implement Project.....	62
5.3 Network Design.....	62
5.3.1 Router Configuration.....	64
5.3.2 BIND Configuration.....	67
5.3.2.1 Domain Configuration.....	68
5.3.2.1.1 Network and DNS Setting for Domain Server.....	70
5.3.2.2 Subdomain Configuration.....	71
5.3.2.1.2 Network and DNS Setting for Domain Server.....	74
5.3.3 DHCP Server Configuration.....	75
5.3.3.1 Network and DNS Setting in DHCP Server.....	77
5.4 Procedure and Operation.....	79
5.5 Performance Measurement.....	94
Chapter 6 Conclusion.....	97
Reference.....	98

List of Figures:

Figure 2.1 Structure of DNS.....	10
Figure 2.2 The Datagardens.com Domain.....	11
Figure 2.3 A Node in Multiple domains.....	12
Figure 2.4 Hierarchy of domain names.....	14
Figure 2.5 Alberta.ca domain is delegated to Province Alberta.....	16
Figure 2.6 Com domain broken into zones.....	18
Figure 2.7 The ibm.com domain broken into more sub domain.....	19
Figure 4.1 Exchanged messages between a client and a server to assign an initial address.....	57
Figure 5.1 Network diagram.....	63
Figure 5.2 IP address and DNS setting in Domain server.....	71
Figure 5.3 IP address and DNS setting in Sub Domain server.....	75
Figure 5.4 IP address and DNS setting in DHCP server.....	78
Figure 5.5 DHCP settings in web server and client.....	79
Figure 5.6 PING reply from all router interfaces.....	82
Figure 5.7 IPCONFIG output of web server.....	83
Figure 5.8 NSLOOKUP output of web server.....	85
Figure 5.9 PING reply from web server.....	86
Figure 5.10 Dynamic update log output.....	87
Figure 5.11 IPCONFIG output of web server (After moved).....	88
Figure 5.12 Syslog output of DHCP server.....	90
Figure 5.13 NSLOOKUP output of web server (After moved).....	91
Figure 5.14 Open webpage of web server (after moved).....	92
Figure 5.15 NSLOOKUP of other residing host on Network.....	93
Figure 5.16 NSLOOKUP of web server when moved to different subnet	95

List of Tables

Table 2.1 Top-Level Domains.....	14
Table 2.2 Country-code top-level domains.....	15
Table 2.3 New generic Top level domains.....	15
Table 5.1 Router information.....	63
Table 5.2 Workstation Information.....	64
Table 5.3 Router interface IP address scheme.....	64
Table 5.4 Time reading of web server when TTL =600.....	95
Table 5.5 Time reading of web server when TTL =300.....	96

Chapter 1 Project Introduction

1.1 Project subject

The project implemented here is a real problem faced by one of the IT Companies, Datagardens. The company is making a variety of software products for IT industries. One of the products is used to move the virtual Database server from one subnet of networks to another in the WAN environment. The basic purpose of migrating Database server is to provide data sharing between local and remote WAN users. The Virtualized Database server can be moved across different physical servers to manage the network or server load, or to respond to failures. Once the Database server has been moved, the existing and any new clients must be pointed to the new IP address so there is no service disruption. That is, somehow the clients must be informed of the new IP address and domain name of the moved Database server. The aim of this project is to implement an automatic system by which the existing and new clients are able to access the moved Database server from anywhere in the WAN.

1.2 Proposed Solution:

Here the project solution is implemented by using the BIND server, the DHCP server and the Dynamic DNS (DDNS) concept. Instead of using virtual software to move the Database server, I used a physical web server which makes no difference in the solution. The BIND server, software running on computer, retains information about IP addresses and names of computers living on network. This is like a central database for all clients in the network. If client want to know about the IP address of neighbour client then it will send queries to the BIND server by domain name or by IP address. For example Client asks the BIND server, “I am looking for the IP address of the host “ualberta.ca”. The BIND server will check in its database and reply a corresponding IP address of the queried host. The project solution is implemented by using this concept of BIND server to locate moved Virtual Database Server (web server) on a WAN link. If the web server is moved from one subnet to another, the BIND database will change according to new assigned IP address and domain name of the web server. For example if web server is moved then I will tell the BIND server that the IP address 192.168.13.5 for host “ualberta.ca” is no longer valid and has been moved to the new IP address 192.168.40.5 so please let me update it. But what will happen if hundreds of web servers are moved from one subnet to another subnet together. It is not suitable to change all entries by hand in the BIND server when web servers migrate frequently. Therefore, an automatic system is needed to update BIND database entries. To solve this problem DDNS is used here. Dynamic DNS is capable of updating the BIND server automatically without any human intervention. However, the BIND server is not capable of updating a DNS entry without the DHCP server. We know the DHCP server assigns an IP address to the client dynamically. In our project if the web server is moved to another subnet, then it will get an IP address from the DHCP server. The DHCP server will also assign some other TCP/IP configuration parameter like a subnet mask, default gateway, DNS server etc to the web server. Once web server has all the network information then DHCP server will send the Dynamic DNS updates to the BIND server informing the assignment of new IP address 192.168.1.1 to host “ualberta.ca” and requesting to update its database. Suppose if the host “ualberta.ca” is moved to another subnet 192.168.2.1 then the DHCP

server will assigns new IP address to host. The DHCP server will send a Dynamic DNS update to the BIND server to remove the existing IP address 192.168.1.1 for the “ualberta.ca” host and add a new IP address 192.168.2.1. As per the DHCP server instructions, the BIND server will remove the existing IP address for host “ualberta.ca” and add a new IP address.

The BIND server also has lots of good features that will be useful for migration of virtual data base server more easily and securely in the WAN environment.

Chapter 2 DNS

2.1 History of DNS

In the late 1960s, the Advanced Research Projects Agency (ARPA) of U.S. (Department of Defence) started project known as ARPAnet to connect research organization in the U.S. to share important resources of Government organization.

During 1970s, the ARPAnet was a very small. Initially there were only around hundred hosts available in network. Information of all host and network address were maintained in the HOSTS.TXT file. Basically HOST.TXT was used for mapping between hostname and address.

In early 1980s new Protocol suite The Transmission Control Protocol/Internet Protocol (TCP/IP) was developed and became the standard host-networking protocol on the ARPAnet. Same time University of California at Berkeley's popular BSD Unix operating system was available virtually free to users. So number of host in Arpanet increase suddenly and known as Internet. The size of HOSTS.TXT grew in proportion to the growth in the number of ARPAnet hosts.

Updating the HOST.TXT with new thousand of host was become impractical. Adding new host in ARPAnet created problems like Traffic and load and Name collisions etc.

Maintaining and updating the HOST.TXT file across an expanding network became harder and harder. Before updated file HOXT.TXT reach to remote host may be new hosts were added or disconnected from network.

A replacement for the HOSTS.TXT file was needed. Some simplified system was needed to solve the problems inherent in a unified host table system. The new simplified system update database locally and available to globally.

In 1984 Paul Mockapetris designed architecture for new system known as DNS domain name system.

2.2 Introduction of DNS

The DNS (Domain Name System) is the globally hierarchical distributed database system used for naming the human readable identity of Internet host. The DNS is an Internet Engineering Task Force (IETF) standard. Internet hosts are known by IP address. The DNS translates meaningful human readable domain names into the IP addresses. It is very important because computer understands only IP address and it is inconvenient to remember millions of host on Internet by the IP address. The DNS Gives global identity to host in Internet so any one can reach to particular host by the domain name. It works as mediator between host domain name and IP address.

For example DNS translates computer name like www.ualberta.ca to numerical IP address such as 129.128.98.86. DNS names are easier for people to read and remember than IP addresses.

The DNS service enables client computers on network to register and resolve DNS domain names. These names are used to find and access resources offered by other computers on Internet or private network. Following are main core component of the DNS:

- ❖ **Domain name space and associated resource records (RRs)** A distributed database of name-related information.
- ❖ **DNS Name Servers:** These are servers that hold the domain name space and RRs, and answer queries from DNS clients.
- ❖ **DNS Resolvers:** Program used by DNS client to query the name server.

The Domain Name System is a distributed database. Small segment of Domain name space is available to global database. It is work on client/server model. Client situated in remote place can updated data in global database.

The structure of the DNS database, shown in Figure 2.1, DNS database looks like inverted tree structure, starting at an unnamed root used for all DNS operations. The root node is written as a single dot (.) and it is on top of the DNS tree. Each node in tree is label with text and it relate with top root node.

2.3 The Domain Namespace

The naming system on which DNS is based is a hierarchical and logical tree structure known as the domain namespace.

The tree has a single root at the top DNS's tree can branch any number of ways at each intersection point, or node. The depth of the tree is limited to 127 levels

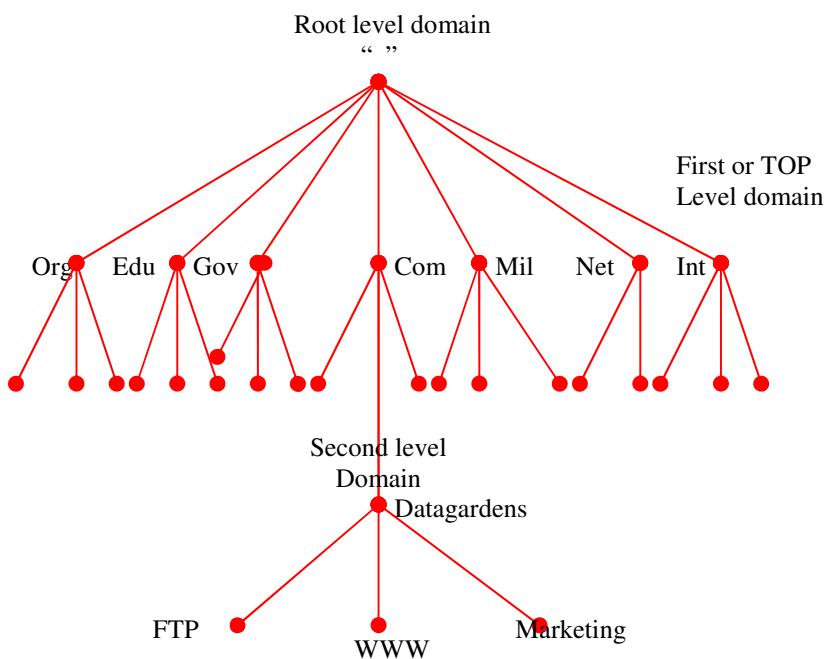


Figure 2.1 Structure of DNS

2.4 Domain Names and Domains

2.4.1 Domain names

The Domain name is assigned by the position of the node in the domain name space. Every domain has a unique name. The Assigned name for each node in the domain space is long up to 63 characters. The Domain name can be made by connecting all labels come in path to root. Dot (.) is used to separate name when connecting name of two different nodes. Domain names start from node and end to root. Every node in the DNS domain tree can be identified by a the fully qualified domain name (FQDN).

2.4.2 Domains

Each node of the root node is worked as the root node of the new sub tree. All these sub trees is part of the DNS database and known as domain. In DNS, each domain can be broken into a number of sub domains, and responsibility for those sub domains can be assign to different organizations. These new domains again divided in the sub domains. A domain is sub tree of the domain namespace. As shown in Figure 2.2 domain name at top of node *Datagardens.com* is same as domain name *Datagardens.com*.

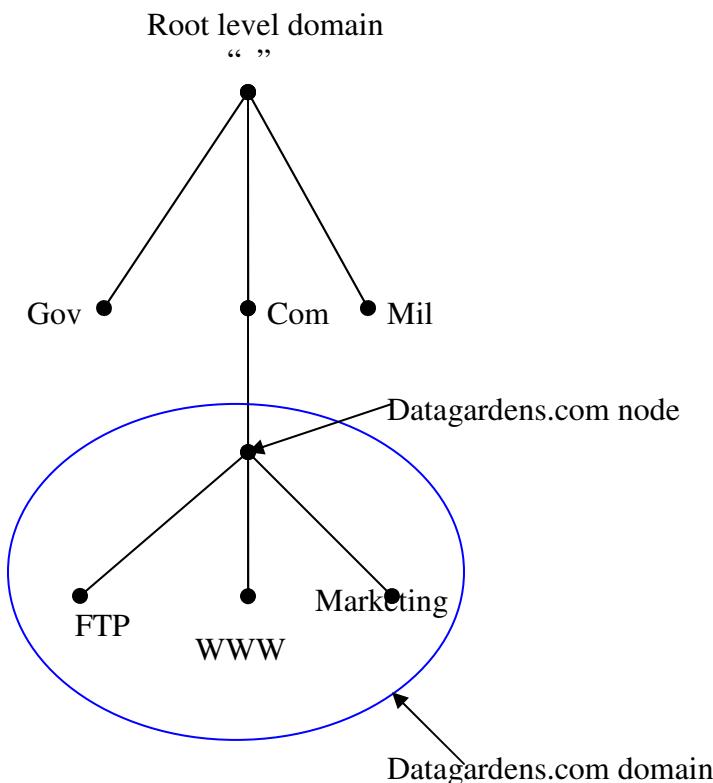


Figure 2.2 The Datagardens.com Domain

Any domain name in the sub tree is considered a part of the domain. The Domain name can result in many sub trees accordingly a domain name can also be in many

domains. For example, as shown in Figure 2.3 the *mint.ualberta.ca* is domain name. It is part of two domain first is the *ualberta.ca* domain and also part of the *ca* domain.

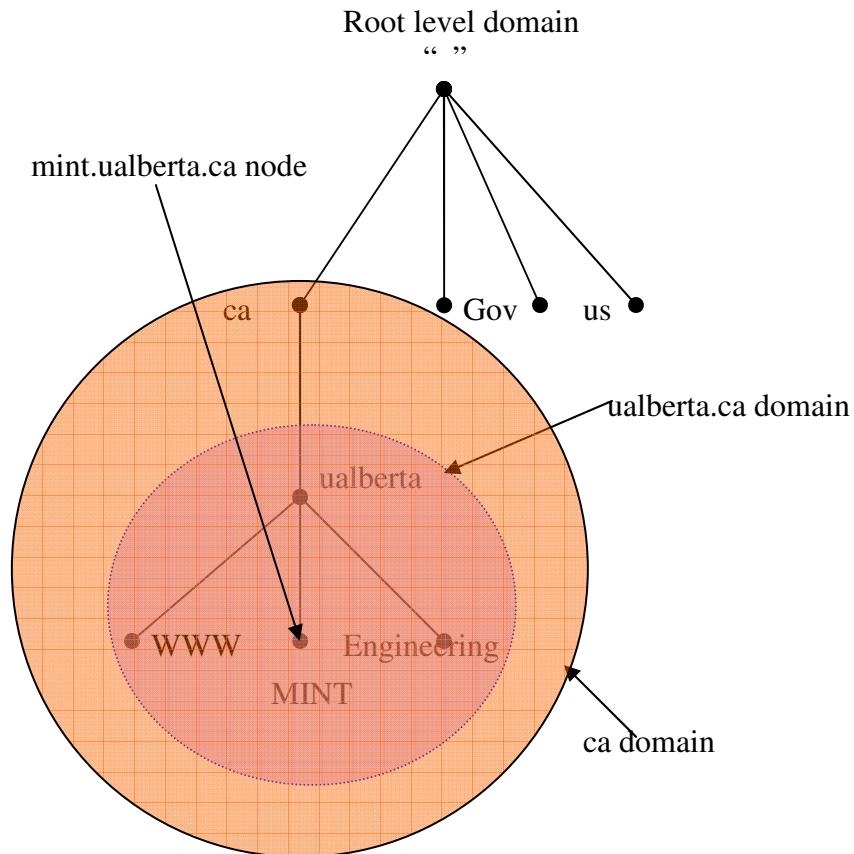


Figure 2.3 A node in multiple domains

The Domain is subtree of the domain namespace. It also includes the host or groups of hosts. Host is also part of domain name. It is used to provide internet service like web server or FTP server to the Internet users. Domain name of the host doesn't depend on geographical location. Suppose domain name *ualberta.ca* may have some host located in other country.

The Domain name assign logically by geographical location or by organization name. Might be 15 different host are in 15 different country but still they can get same domain suffix. If law department of University of Alberta *law.ualberta.ca* is situated in Canada and engineering department *ece.ualberta.ca* may be in U.S. but still both departments are part of *ualberta.ca* domain. So domain name doesn't depend on network address or hardware type.

The Domain names at the leaves of the domain namespace inverted tree generally represent particular host on the network. For example, *ualberta.ca* is both the name of the University of Alberta's domain and a domain name that refers to the hosts that run main web server.

A simple way of determining if a domain is a subdomain of another domain is to compare their domain names. A subdomain's domain name ends with the domain name of its parent domain. For example, the domain *mint.ualbera.ca* must be a subdomain of *ualberta.ca*, because *mint.ualberta.ca* ends with *ualberta.ca*. It's also a subdomain of *ca*, as is *ualberta.ca*.

2.5 Resource Records

The Resource records are used to configure nameserver. It is used to define data associated with the domain name. Internet class (IN) is most popular class. It has different type of resource records like A records, PTR records, SOA records, NS records, CNAME records etc. All are used to configure zone data file of the DNS. Detail description of resource record is given in chapter 3

2.6 The Internet Domain Namespace

There is no any rules and regulation to assign domain name for private network. You can use any combination of letter A to Z anywhere in domain name but the existing Internet domain namespace has some self-imposed structure to it. To assign domain name in Internet domain namespace one has to follow some specified rules and regulation. It is useful to understand the location of host in the domain space, especially in the upper-level domains. These structure help to understand domain names easily.

Domains are often referred to by the level. These terms are very helpful to know the domain's position in the domain namespace:

- ❖ A top-level or a first-level domain is a child of the root.
- ❖ A second-level domain is a child of a first-level domain, and so on.

2.6.1 Hierarchy of domain names

The structure of DNS database is inverted tree. The Root domain name server is situated on top of the hierarchy then after first or top level domains are available in path. The root of the tree has no name. After top level domain second level domain comes in path. Following Figure 2.4 show hierarchy of domains names

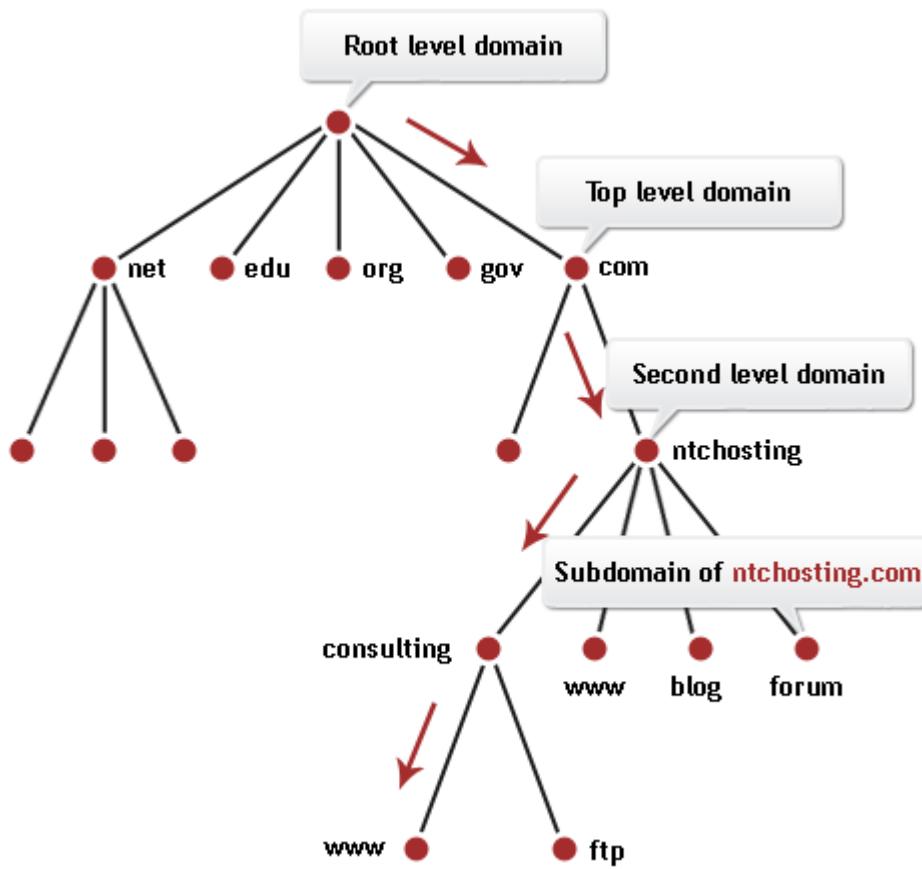


Figure 2.4 Hierarchy of domain names

Figure from <http://www.ntchosting.com/dns/>

2.6.2 Top-Level Domains

The original top-level domains divided the Internet domain namespace organizationally into seven domains:

gTLD	Type of organization
Com	Use for commercial organizations, such as Datagardens (datagardens.com).
Edu	Educational organizations, such as U.C. Berkeley (berkeley.edu)
Gov	Government organizations, such as NASA (nasa.gov) and The National Institutes of Health of the U.S (NIH.gov).
Mil	Military organizations, such as the U.S. Army (army.mil) and Navy (navy.mil).
Net	Allocated for network infrastructure like nsf.net etc but now it can use for commercial purpose Ex. Classical.net for music
Org	Wikipedia.org
Int	International organizations, such as NATO (nato.int).

Table 2.1 Top-Level Domains

These original seven domains are called generic top-level domains, or gTLDs. They are different from the country-code top-level domains, which are specific to a particular country.

2.6.3 Country-code top-level domains

Use of Internet is increased day by day. For better administration of DNS individual domain name is assigned to each country by name of country. All the country has separate domain called country code top –level domains. These domain names followed an existing international standard.

Following table show you some assigned domain name for country

Country	Domain name
Canada	.ca
USA	.us
France	.fr

Table 2.2 Country-code top-level domains

2.6.4 New top-level domains

To fulfil the requirement of certain organization the Internet Assigned Numbers Authority (IANA) has assigned some special purpose domain name. These are different from original seven gTLD. Some domain name from these new top level domain names are also defined as generic top level domain. Example .biz, .info, .name, .pro and .travel domains are generic domain

Table show you some new gTLDs:

Aero	Sponsored; for the aeronautical industry
Biz	Generic For business
Coop	Sponsored; for cooperatives
Info	Generic
Museum	Sponsored; for museums
Name	for individuals
Pro	for professionals

Table 2.3 New generic Top level domains

2.7 Delegation

Delegation means divide domain to more sub domain to assign responsibility of some part of domain space to other organization. It is difficult to manage domain if it grow more. By using concept of delegation domain name is divided in sub domain to give responsibility of group of host to other administration. Sub domain has just point to the parent domain to connect Internet. In below figure *alberta.ca* domain is delegated to one of the province of Canada, Alberta.

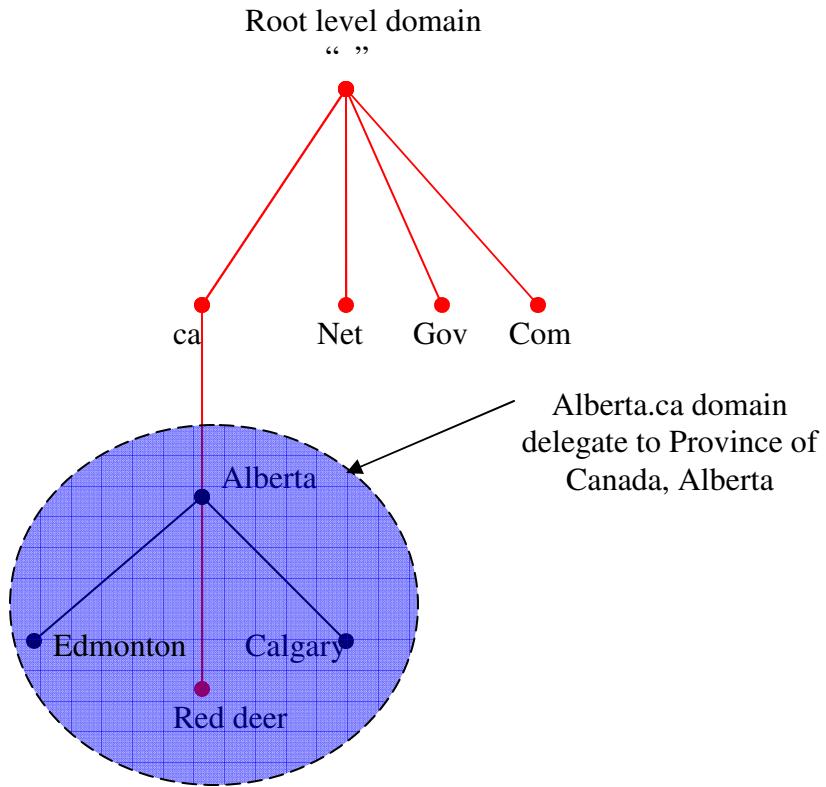


Figure 2.5 alberta.ca domain is delegated to province Alberta

2.8 Name servers

Program running on computer which store information about the domain name space is known as nameserver. Nameserver is responsible for some part of domain namespace. Nameserver can also be responsible for more than one zone. It loads data for zone from other authoritative server for that zone or by dynamic updates. Nameserver is responsible to solve query done by client. If nameserver doesn't know the answer of query, it will refer neighbour nameserver to get answer.

2.8.1 Types of name server

There are total five types of name server available. Type of the nameserver is depending on the configuration of name server.

2.8.1.1 Authoritative Name Servers

The Authoritative name server is same as primary and secondary name server. Zone has at least one authoritative server either served by primary or secondary name server. Authoritative server is responsible to reply information to client who did the query for particular domain. For redundancy each zone served by more than two authoritative server

2.8.1.2 The Primary Master

Type of authoritative server where the original data of zone file is stored and available to client is called the primary master server or simply the primary. Primary master is using zone data file edited by human or updated dynamically through DDNS.

2.8.1.3 Slave Servers

Slave server is also known as secondary server. It is also type of authoritative servers. Slave server gets the zone data file from other source like primary server or secondary slave. Zone data file is transferred from primary server to slave server by using zone transfer technique. Where the slave sends NOTIFY request to master server for a particular zone and primary server sends information if the slave is authorized to receive the data. Slave server works as authoritative server for those zones which is defined as slave. We can say that slave server work as master to a subordinate slave server.

Slave name servers are good for redundancy. It is important when service by primary server is interrupted. Also it has important role to spread the load around network. Administration work is easy by using slave name servers. Secondary servers are recommended in larger network setups

2.8.1.4 Caching Name Servers

Caching server stored the lookups in local database obtained from another name server for specified period of time. Purpose of caching server is to speeding up name resolution. If client queried for same information again then caching name server reply with information stored in local database. Network administrator used TTL to define time for cache to store in nameserver. Once TTL is expired nameserver automatically discard data of remote nameserver.

2.8.1.5 Multiple or hybrid name server

The DNS name server can simultaneously act as Caching and Primary server, caching and secondary server, primary and secondary server by changing configuration. This kind of server is recommended where resources are less.

2.9 Zones

Top level domain is broken into more domains. Here as shown in bellow Figure 2.6 *com* domain is divided in *cisco.com*, *Nortel.com* and *juniper.com* are known as zones.

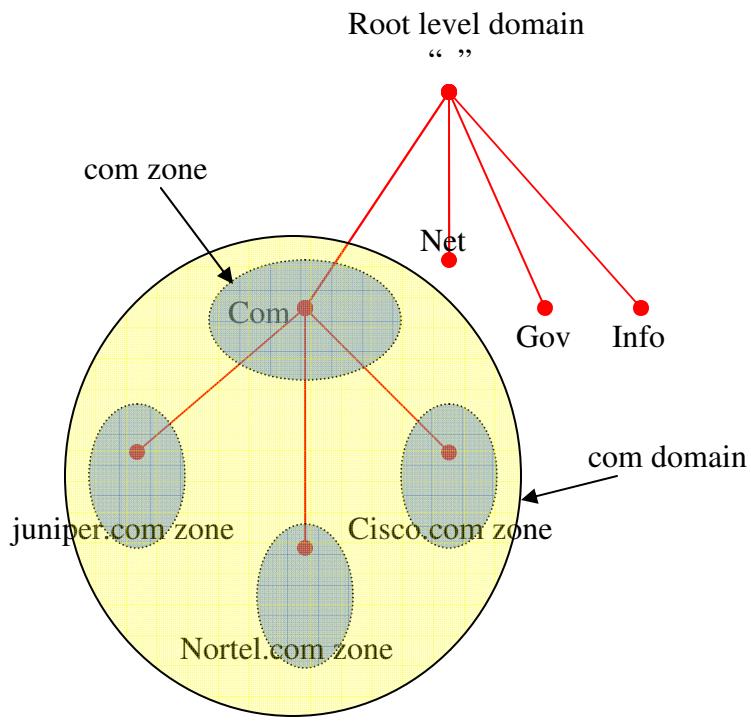


Figure 2.6 Com domain broken into zones

As shown in below Figure 2.7 *ibm.com* subdomain can also delegate to different department of IBM Company for better administration of *ibm.com* sub domain. Here *ibm.com* zone is further divided in to zone like *srv.ibm.com*, *rpr.ibm.com*, *mkt.ibm.com* etc. all zone can have their separate name server or may be share with some other zone.

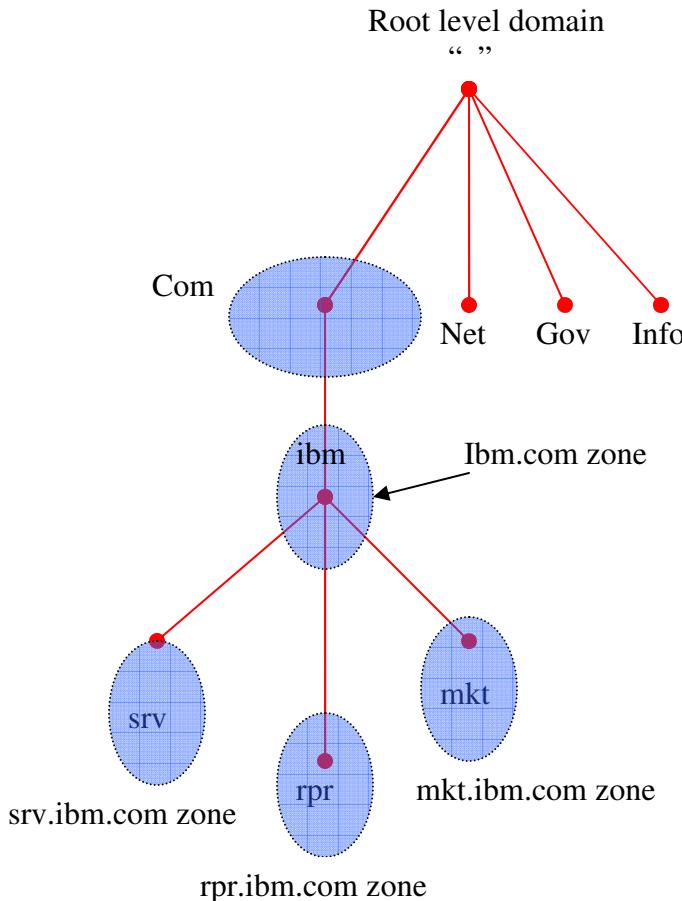


Figure 2.7 The ibm.com domain broken into more sub domain

2.10 Resolvers

Resolver is software running on client machine. It is used by host to query nameserver. It is also interpret responses from nameserver and give the information to the programs who requested.

Resolver is smart, they place query and wait until get answer. If they didn't get answer they resend query.

2.11 Resolution

Nameserver is not only responsible to retrieve data from the authoritative zone but also smart to go through domain namespace for which it is not responsible to find data to answer query done by client. This process is called name resolution. If nameserver doesn't get answer from its zone data then it ask root nameserver to answer query

2.12 Root Nameservers

Root name server has important role in name resolution process. Root name server has information about respective top level authoritative name server for zone. In

some cases Root name server is also authoritative for some generic top level domain. For given query to root name server it reply with name and address of nameserver that are authoritative for top level domain name. If top level domain doesn't have any information then it point to the second level domain. If second level nameserver has answer then it replies otherwise again it will refer to closer authoritative nameserver for query.

Only possibility of not getting resolution is if client is not able to contact root nameserver. To avoid this problem 13 root nameserver spread across the Internet. So same root server available more than one place on internet.

2.13 Caching

When client query to the nameserver, nameserver process it and reply back to client but if nameserver doesn't know the answer then it does recursive query. In recursive query nameserver refer another nameserver to resolve query. Nameserver refer to another namserver until finds answer of query. During this process namserver meet lots of new server and at the end when nameserver got answer, it will store the information like address of nameserver, authoritative zone for nameserver etc. next time when nameserver receive query for same nameserver then it will used cache data. By using this data nameserver speed up to resolve query.

2.14 Time to Live

Nameserver can cache data for some specified amount of time but they can't cache it forever. If they cache it forever then remote server never get updated data of authoritative server. To solve this problem DNS server can be configure with TTL option. TTL specifies time in seconds. Name server can cache data for the time specified in TTL option. Once this time is elapsed nameserver discard stored data and get the new data from authoritative server.

The administrator of the zone is responsible to decide time to live (TTL) for the zone. Deciding time to live for domain is important. There are some advantage and disadvantage of keeping TTL higher or lower. DNS performance and consistency depend on TTL. A small TTL helps to discard data of remote nameserver quickly and force authoritative server for new data. On the other hand, TTL increases the network traffic, load on nameservers and increase the average resolution time for query. A large TTL help to reduce the average resolution time for query because the data can be cached in nameserver for longer. The drawback is if any changes in authoritative nameserver's data then it will not reached to remote server until TTL is expired in remote server.

The nameserver sends information of TTL back to client in query responses. This means client or namserver cache data until TTL is expired. It is good to keep default TTL for some days if authoritative server data doesn't changed frequently. Never set TTL to zero because it increases too much load on server if it replies thousands of queries per day.

Chapter 3 BIND

3.1 Introduction of BIND

The Berkeley Internet Name Domain (BIND) also known as named is open-source software that implements the Domain Name System (DNS) protocols for the Internet. It is the most commonly used than any other DNS server software on the Internet. It is widely used for UNIX-like systems. It is a de facto standard and provides a robust and stable platform to implement domain name system, support the entire standard required by DNS protocol. It is appropriate for high reliability and high volume applications. Currently BIND is maintained and developed by Internet Systems Consortium.

The DNS protocols are part of the core Internet standards. DNS is used for mapping between internet host and address. So it is easy to find host on internet by using human readable name. BIND is basically implemented to fulfill all the function needed by DNS protocols. BIND software distribution contains all of the software needed to perform task explained in DNS.

The BIND software distribution contains three parts:

- 1) Domain Name System server
- 2) Domain Name System resolver library
- 3) Tools for managing and verifying the proper operation of the DNS server

In early 1980s, The Berkeley Internet Name Domain package was originally implemented at University of California at Berkeley by four graduate students (Douglas Terry, Mark Painter, David Riggle and Songnian Zhou) as a graduate project under a grant from the Advanced Research Projects Administration (DARPA). Up to Versions 4.8.3 of BIND were maintained by CSRG at UC Berkeley.

In the mid-1980s, DEC (Digital Equipment Corporation) employees took over BIND development. BIND versions 4.9 and 4.9.1 were released by DEC employees

One of the employees, Paul Vixie of DEC who continued to work on BIND after leaving DEC. He released BIND Version 4.9.2 and became BIND's principal architect/programmer.

BIND versions from 4.9.3 onward have been developed and maintained by ISC with support being provided by ISC's sponsors.

Bob Halley and Paul Vixie worked together and released the first production-ready version of BIND version 8 in May 1997.

BIND version 9 was released in September 2000 and lots of new application implemented in BIND9.

BIND9 included some important features like DNS Security: DNSSEC and TSIG, DNS Notify, nsupdate, IPv6, remote name daemon control, IXFR, DDNS, EDNS0, Views, Multiprocessor Support, and an Improved Portability Architecture.

ISC stop development for older BIND version 4 or BIND version 8 other than security related patches.

3.2 Installation of BIND server:

The BIND 9 currently support following system with an ANSI C compiler, basic POSIX support, and a 64 bit integer type.

COMPAQ Tru64 UNIX 5.1B
Fedora Core 6
FreeBSD 4.10, 5.2.1, 6.2
Mac OS X 10.5
NetBSD 3.x and 4.0-beta
OpenBSD 3.3 and up
HP-UX 11.11
Slackware Linux 8.1
Solaris 8, 9, 9 (x86), 10
Ubuntu 7.04, 7.10
Windows XP/2003

There are two different way to install BIND9 in Ubuntu.

First option is download the latest version of BIND from website <http://ftp.isc.org/isc/bind9/>

Unpack the Source Code by using following command.

```
# tar zxvf bind-9.6.1.tar.gz
```

Move to the bind-9.6.1 directory where you want and run the following commands:

```
# cd /tmp/bind-9.6.1  
# ./configure  
# make  
# make install
```

"make install" command will install "named" and BIND 9 libraries. By default, "make install" will install installation into /usr/local directory. Default installation can be changed by using "--prefix" option when running "configure".

Specify the option "--sysconfdir" to set the directory where configuration files like named.conf, named.conf.local etc. go by default, for default parent directory of "run/named.pid" use "--localstatedir".

BIND 8 was used to install by default in /etc and /var directory. To compatible with BIND 9 used sysconfdir defaults to "\$prefix/etc" and localstatedir defaults to "\$prefix/var".

To see additional configure options, run "configure --help".

Use the following command to check installed BIND version.

```
# named -v
```

Second option is appropriate when you want to use shipped version of BIND9 by operation system. Just enter following command in command line input

```
#sudo apt-get install bind9
```

Above command will install BIND9 version.
Use `# named -v` to see BIND is installed successfully.

BIND9 Configuration files are stored in /etc/bind/ directory.

Following are by default installed file in: /etc/bind directory.

```
kaki@kaki-laptop:/etc/bind$ ls  
db.0 db.127 db.255 db.empty db.local db.root named.conf named.conf.local  
named.conf.options rndc.key zones.rfc1918
```

The main configuration is stored in the following files:

- ❖ /etc/bind/named.conf
- ❖ /etc/bind/named.conf.local
- ❖ /etc/bind/named.conf.options
- ❖ /etc/bind/rndc.key

The primary configuration file is /etc/bind/named.conf.
`named.conf` file which controls the behaviour and functionality of BIND.
named.conf is the only file which is used by BIND during first time running. If file is not available then it gives error message.

If BIND installed from the source code, you will have to edit the file named.conf but Ubuntu provides pre-configured BIND files.

Two more file also required to create. That is used to stored forward and reverse zone data file.

/etc/bind/zones/cust.com.db

/etc/bind/zones/in-addr.arpa

We will create these two file later.

3.2.1 `named.conf` default configuration file.

```
include "/etc/bind/named.conf.options";  
zone "." {  
    type hint;  
    file "/etc/bind/db.root";  
};  
zone "localhost" {  
    type master;  
    file "/etc/bind/db.local";  
};  
  
zone "127.in-addr.arpa" {  
    type master;
```

```

        file "/etc/bind/db.127";
};

zone "0.in-addr.arpa" {
    type master;
    file "/etc/bind/db.0";
};

zone "255.in-addr.arpa" {
    type master;
    file "/etc/bind/db.255";
};

include "/etc/bind/named.conf.local";

```

The `named.conf.local` file is blank and it is used to edit local configuration about zone. It is used to specified configuration like logging, define location of `rndc.key` file, specification about where zone data file (forward and reverse map) is stored, configuration about zone type like primary or secondary, configuration about Dynamic update etc.

The `named.conf.local` configuration file for our experiments is explained in chapter 5. The file named `/etc/bind/db.root` describes the root nameservers in the world. The servers change over time, so the `/etc/bind/db.root` file must be maintained now and then. It is generally updated when new version of BIND available in market.

The `include` statement inserts the specified file at the point where the `include` statement is encountered. It can appear anywhere in a `named.conf` file either inside or outside a clause. `Include` statement is use to hide important configuration like security key etc from world.

3.3 The Zone Datafiles

The Zone data file is heart of the DNS. Zone files consist of Comments, Directives and Resource Records. Zone files contain information about a forward and reverse mapping of host and address, domain name, responsible authoritative server, information for slave server, TTL etc. Zone data files are stored in the separate directory. In zone data files most entries are resource records. All directives and resource records entered in individual lines. Zone data files can be edited by hand or updated automatically from other authoritative nameserver. Any combination of uppercase, lowercase, or mixed case are used to configure zone data file because DNS queries are not case sensitive.

3.3.1 Zone File Directives

The Directives are always start with the character \$. Name of the directive start after \$. Directives usually appear at the top of the zone data file.

The following directives are widely used in zone data file

3.3.1.1 \$INCLUDE

Syntax: \$INCLUDE filename [origin] [comment]

It is used to add zone configuration file from different directory to current zone configured data file. Useful to enhance security of zone data file.

3.3.1.2 \$ORIGIN

Syntax: \$ORIGIN domain-name [comment]

It is used to append the domain name to unqualified records like hostname.

For example, a zone file may contain the following line:

\$ORIGIN cust.com.

www CNAME dbserver10

is equivalent to

www.cust.com cname dbserver10.cust.com.

Any names used in resource records that do not end in a trailing period (.) will have cust.com appended to them.

3.3.1.3 \$TTL

Syntax: \$TTL default-ttl [comment]

Range 0-2147483647 seconds.

BIND allows to define TTL in different time unit even in day and week format. All units are case sensitive

1s = 1seconds = 1 seconds

1m = 1minutes = 60 seconds

1h = 1hours = 3600 seconds

1d = 1day = 86400 seconds

1w = 1week = 604800 seconds

TTL1d is equivalent to 86400 seconds

```
; cust.com zone
$TTL 2d ; zone default time to live
        4d IN MX 10 192.168.1.2 ; overrides default
bhatudi 5w IN A      192.168.1.3 ; overrides default
www      IN A      192.168.1.4 ; uses zone default = 2 days
```

The time-to-live is also component of the RR. It Sets the Time to Live (TTL) value for the zone. TTL specifies time to keep cache data in local database of nameserver. Once the time defined by TTL option is elapsed, nameserver will removed cache data automatically. By default TTL unit is second and it is 32-bit integer. Stored Lookup in client cache is dependent on TTL value. The TTL tells how long a RR can be cached in client before it should be discarded.

Setting the Zone's Default TTL

A TTL role in DNS is important. Performance of nameserver is dependent on used value to define TTL. When client query nameserver it stored lookups data until TTL expired in zone data file. If TTL do not expires then clients or nameservers never able to

reach to authoritative nameserver of the domain. Nameserver used cache data in local database for lookups which did before and queried again.

The time to live is the amount of time that any nameserver is allowed to cache the data. Once the time to live expires, the nameserver must discard the cached data and get new data from the authoritative nameservers. This also applies to negatively cached data: a nameserver must time out a negative answer after a period in case new data has been added on the authoritative nameservers.

The administrator of the zone is responsible to decide time to live (TTL) for the data. Deciding a time to live for domain is important. There are some advantage and disadvantage of keeping TTL higher or lower. DNS performance and consistency depend on TTL. A small TTL helps to discard data of remote nameserver quickly and force authoritative server for new data. On the other hand, TTL increases the network traffic, load on nameservers and increase the average resolution time to response query.

A large TTL help to reduce the average resolution time in because the data can be cached in nameserver for longer. The drawback is if any changes in authoritative nameserver data then it will not reached to remote server until TTL is expired in remote server.

The nameserver sends information of TTL back to client or nameserver with query responses. This means client or namserver cache data until TTL is expired. It is good to keep default TTL for some days if authoritative server data do not change frequently. Never set TTL to zero because it increases too much load on server if replies thousands of queries per day.

The \$TTL directive should be present and appear before the first Resource Record.

3.3.2 Comments

The zone datafiles are easier to understand if they contain comments and blank lines.

Comments always start with ';' (semicolon).it is assumed to continue to the end of the line. Comments can occupy a whole line or part of a line as shown in the above example.

3.3.3 Zone File resource records

The Resource record is used to configure zone's datafile. Resource records need to be start in the first column of a line. Some specific rules are defines in DNS RFC for order of resource record but doesn't matter. Here I explain only few components of RRs which used widely and enough for our zone datafile configuration.

3.3.3.1 SOA Record

The Start of Authority record defines global parameters for the zone (domain). There is only one SOA record allowed in a zone file. The first Resource Record must be the SOA record. It starts after the directives.

The following example shows the basic structure of an SOA resource record:

```
@ IN SOA #primary-name-server# #hostmaster-email# (
```

```
#serial-number#
#time-to-refresh#
#time-to-retry#
#time-to-expire#
#minimum-TTL#)
```

@ Symbol replaces the \$ORIGIN directive. If \$ORIGIN is doesn't set then it will use zone's name.

Primary nameserver field is use to define name of the authoritative name server which is responsible for this domain.

Hostmaster-email field is use for e-mail address of administrator who is responsible for this zone to solve problems and error.

Serial-number field is a numerical value and incremented each time when the zone configuration file is change. Serial number can be filling by any unsigned 32 bit value in range 1 to 4294967295. The most popular format for serial number is YYYYMMDDSS where YYYY = year, MM = month, DD = day and SS = a sequence number of day. Other timings are use by secondary nameserver to update zone datafiles. Indicates the time when the slave will try to refresh the zone from the master.

Time-to-refresh is waiting time for slave server before asking master for any changes in zone data. If the serial number of master and slave is not match than slave update zone data file from master. Refresh time will be any value in range of 32 bit and in second unit

Time-to-retry Time used by slave server to contact master once refresh time is expired. Slave will contact to master continuously after time defined in time-to-retry field expired and once it reached to **time-to-expire** time slave server will stop to work as authoritative server. Because slave server has no any contact with primary server.

The **minimum-TTL** is time other name servers keep cache of zone's information.

As explained in TTL directive, SOA is also configured in other time unit like minutes (M), hours (H), days (D), and weeks (W).

Following zone data file is configured with SOA record.

```
$TTL 86400
@ IN SOA ns.cust.com. pakodi.cust.com. (
        41 ; Serial
        604800 ; Refresh
        86400 ; Retry
        2419200 ; Expire
        86400 ) ; Negative Cache TTL
;

cust.com.      IN  NS      ns.cust.com.
cust.com.      IN  A       192.168.1.4
ns             IN  A       192.168.1.4
```

3.3.3.2 NS record

It is used to lists a authoritative name server for the zone

Syntax: IN NS <nameserver-name>

The <nameserver-name> should be

- 1) A Fully Qualified Domain Name (FQDN) e.g. cust.com. (Ends with a dot)
- 2) An unqualified name (does not end with a dot)
- 3) An '@'
- 4) A 'space' or 'blank' (tab) - previous value of the name field is used if not defined. If previously name not defined then \$ORIGIN is used

Here two nameservers are listed as authoritative for the domain.

NAME	TTL	CLASS	RR	NAME
cust.com.		IN	NS	ns1.cust.com.
cust.com.		IN	NS	ns2.cust.com.

3.3.3.3 A Record

Name-to-address mapping.

It is used to map host name to IPv4 address.

Syntax :	<host>	IN	A	<IP-address>
		IN	A	192.168.1.4
	dgrdn	IN	A	192.168.1.3

The IP address is not terminated with a '.' (Dot). If host name is not specified (or space) then the last valid name (or label) is used. Host Name is non-FQDN.

Consider above A record examples for the cust.com zone file:

Requests for cust.com are pointed to 192.168.1.4, while requests for dgrdn.cust.com are pointed to 192.168.1.3.

3.3.3.4 PTR Record

Address-to-name mapping.

Pointer records are used in reverse map zone file. It works the opposite of A and AAAA RRs. Used to map an IP address (IPv4 or IPv6) to FQDN.

PTR record syntax:

Syntax:	last-IP-digit	IN	PTR	FQDN-of-system
	15	IN	PTR	www.cust.com.

If someone query by above IP address then name server will give answer with www.cust.com.

3.3.3.5 CNAME

Canonical name (for aliases).

A CNAME record maps an alias name to the real name.

Syntax	<alias-name>	IN	CNAME	<real-name>
	dbserver	IN	A	192.168.1.3
	www	IN	CNAME	dbserver

In the above example any requests sent to the www (alias-name) it will point to the dbserver <real-name>. www commonly used for Web servers.

3.4 Configuration of BIND server

Following example show you how to configure BIND server for caching nameserver, primary master and secondary master.

3.4.1 Caching Nameserver

Default configuration of BIND is act as caching Nameserver.

Just it needs some changes in /etc/bind/named.conf.options configuration file. It needs real IP address of DNS server provided by ISP.

Simply uncomment and add the following statement in /etc/bind/named.conf.options:

```
forwarders {  
    75.154.132.68;  
    75.154.132.100;  
};
```

Now load the new configurations file. Restart the DNS server from a terminal prompt:
sudo /etc/init.d/bind9 restart

Testing

Dig %domain name% command is use to query nameserve about specific domain name. following command ask nameserver about google.com domain.

```
dig google.com
```

if all are good then output like this.

```
user@appl-user-desktop:~$ dig google.com
```

```
; <>> DiG 9.6.1 <>> google.com  
;; global options: +cmd  
;; Got answer:  
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 25780  
;; flags: qr rd ra; QUERY: 1, ANSWER: 3, AUTHORITY: 4, ADDITIONAL: 4  
  
;; QUESTION SECTION:  
;google.com.      IN  A  
  
;; ANSWER SECTION:  
google.com.      5    IN  A   74.125.45.100
```

```
google.com.      5   IN   A    74.125.127.100
google.com.      5   IN   A    74.125.67.100
```

;; AUTHORITY SECTION:

```
google.com.      5   IN   NS   ns4.google.com.
google.com.      5   IN   NS   ns1.google.com.
google.com.      5   IN   NS   ns3.google.com.
google.com.      5   IN   NS   ns2.google.com.
```

;; ADDITIONAL SECTION:

```
ns3.google.com. 5   IN   A    216.239.36.10
ns2.google.com. 5   IN   A    216.239.34.10
ns4.google.com. 5   IN   A    216.239.38.10
ns1.google.com. 5   IN   A    216.239.32.10
```

```
;; Query time: 23 msec
;; SERVER: 192.168.253.2#53(192.168.253.2)
;; WHEN: Wed Jul 29 11:49:20 2009
;; MSG SIZE rcvd: 212
```

```
user@appl-user-desktop:~$
```

If you "dig" a domain name multiple times you should see a drastic improvement in the Query time: between the first and second query. This is due to the server caching the query.

3.4.2 Primary Master

Now we configure BIND9 as primary nameserver. Here I used cust.com as Fully qualified domain name and IP address 192.168.1.4 for BIND server

3.4.2.1 Forward Zone File

The zone file defines most of the information about domain. It is used to map host names to IP address. The name-to-address lookup is called forward mapping.
Add the following lines in named.conf.local file which is located at /etc/bind/named.conf.local:

```
zone "cust.com"{
type master;
file "/etc/bind/zones/cust.com.db";
};
```

First statement specifies the FQDN of the domain name. This domain name is used to query name server.

Second statement is used to specify type of name server whether it is primary or secondary name server.

Third statement specifies the directory where our zone data file is stored.

Now create the directory to store our zone data file's configuration.

Use following command to create directory.

```
sudo mkdir /etc/bind/zones
```

Next create the zone file which is used to configure forward map data of the domain.

```
sudo gedit /etc/bind/zones/cust.com.db
```

Edit the following configurations in cust.com.db file

```
$TTL 86400
```

```
@ IN SOA ns.cust.com. pakodi.cust.com. (
    41 ; Serial
    604800 ; Refresh
    86400 ; Retry
    2419200 ; Expire
    86400 ) ; Negative Cache TTL
;
```

```
cust.com.      IN  NS      ns.cust.com.
ns            IN  A       192.168.1.4
```

Here ns.cust.com is primary authoritative name server for domain cust.com. Default TTL for this domain is 1 day. Except serial number and A record other configuration is used to update server from other primary name server.

Every time when we change the configuration we will increase serial number in this zone configuration file.

3.4.2.2 Reverse Zone File

It is responsible for mapping IP address to host names, which is exactly the opposite of what the forward zone data file does. The address-to-name lookup is called reverse mapping.

Now Edit the following information in /etc/bind/named.conf.local file

```
zone "in-addr.arpa"{
    type master;
    file "/etc/bind/zones/in-addr.arpa";
};
```

Now create the "/etc/bind/zones/in-addr.arpa";file and add following information for reverse domain.

```
$TTL 604800
@ IN SOA ns.cust.com. pakodi.cust.com.(
    41 ; Serial
    604800 ; Refresh
    86400 ; Retry
    2419200 ; Expire
    604800 ) ; Negative Cache TTL
;
```

```

@           IN   NS    ns.cust.com.
1           IN   PTR    cust.com.

```

As we did in forward zone configuration file we will increase serial number in reverse zone configuration file also. If any one query to nameserver by using IP address 192.168.1.1 then it will reply cust.com domain name.

After creating the reverse zone file restart BIND9 by using following command

```
sudo /etc/init.d/bind9 restart
```

3.4.3 Secondary Master

The Secondary master is needed for redundancy purpose. It is useful when primary master goes down then automatically slave server comes up in network and perform the same task for which primary was responsible.

First we will configure primary server to allow sending configuration to secondary. Add allow transfer statement in both zone files which we edited in /etc/bind/named.conf.local. Others configuration will be same.

```

zone "cust.com"{
    type master;
    file "/etc/bind/zones/cust.com.db";
    allow-transfer { 192.168.1.10; };

};

zone "in-addr.arpa"{
    type master;
    file "/etc/bind/zones/in-addr.arpa";
    allow-transfer { 192.168.1.10; };

};

```

Where 192.168.1.10 is IP Address of Secondary name server.

Same way install BIND server in secondary and add following declaration for Forward and reverse zone in the /etc/bind/named.conf.local

```

zone "cust.com"{
    type slave;
    file "/etc/bind/zones/cust.com.db";
    masters { 192.168.1.4; };

};

zone "in-addr.arpa"{
    type slave;
    file "/etc/bind/zones/in-addr.arpa";
    masters { 192.168.1.4; };

```

```
};
```

Where 192.168.1.4 is IP address of primary server.
Restart BIND9 on the primary and Secondary sever.
`sudo /etc/init.d/bind9 restart`

3.5 Logging

BIND9 has a wide variety of logging configuration options available. There are two main options available 1) Channel and 2) category. The channel option is used to configure where logged data goes such as to specific file, to syslog etc, and the category option determines what information is logged.

Each category of data can be sent to a one location or to multiple locations. Suppose queries are logged to a file while zone transfer data is both logged to a file and to syslog.

Channels are allowed to filter by message severity. Following list tells severities, from most severe to least:

- critical
- error
- warning
- notice
- info
- debug [level]
- dynamic

3.5.1 Logging Statement Grammar

We will use this statement to configure different type of channel and category later.

```
logging {  
    [ channel channel_name {  
        ( file path_name  
            [ versions ( number | unlimited ) ]  
            [ size size spec ]  
            | syslog syslog_facility  
            | stderr  
            | null );  
        [severity (critical | error | warning | notice |  
            info | debug [ level ] | dynamic );]  
        [ print-category yes or no; ]  
        [ print-severity yes or no; ]  
        [ print-time yes or no; ]  
    };]  
    [ category category_name {  
        channel_name ; [ channel_name ; ... ]  
    };]  
...  
}
```

```
};
```

In BIND 9 if no logging option is configured the default option is:

```
logging {  
    category default { default_syslog; default_debug; };  
    category unmatched { null; };  
};
```

By default the default category logs to both syslog and to the debug file.
Now let's configure query category to log in separate query.log file.

```
logging {  
    channel query.log {  
        file "/var/log/query.log";  
        severity debug 3;  
    };  
};
```

Next, configure a category to send all DNS queries to the query file:

```
logging {  
    channel query.log {  
        file "/var/log/query.log";  
        severity debug 3;  
    };  
    category queries { query.log; };  
};
```

Followings are the available categories and brief descriptions of the types of log information they contain.

❖ **Default**

It is used to log all type of category listed here except queries category and undefined category.

❖ **General**

Undefined categories are logged here.

❖ **Database**

Exchanging of Messages relating name server to store zone and cache data.

❖ **Security**

Approval and denial of requests.

❖ **Config**

Configuration files parsing and processing.

❖ **Resolver**

nameserver involved to perform recursive lookups

❖ **Xfer-in**

- Zone transfers update server is receiving.
- ❖ **Xfer-out**
Zone transfers update the server is sending.
- ❖ **Notify**
The NOTIFY protocol.
- ❖ **Client**
Processing of client requests.
- ❖ **Network**
Network operations.
- ❖ **Update**
Dynamic updates.
- ❖ **Update-security**
Approval and denial of update requests.
- ❖ **Queries**
Logs all query transactions.
- ❖ **Dispatch**
Dispatching of incoming packets to the server modules where they are to be processed.
- ❖ **Dnssec**
logged DNSSEC and TSIG protocol processing data
- ❖ **Lame-servers**
Lame servers. These are misconfigurations in remote servers, discovered by BIND 9 when trying to query those servers during resolution.
- ❖ **Delegation-only**
Logged data requested for delegated zone.

3.6 Parenting

When domain should become parents? If domain has any of the following condition then it is good time to delegate sub domain.

- ❖ Divide organization domain according to department affiliation
- ❖ Once domain become large
- ❖ Assign responsibility of domain to separate management

Delegation of domain depends on circumstances like if domain grow more then it is very difficult to manage. Also load on name server is increase. If only one name server is authoritative for large domain then average query response time from name server also increase. If organization has large 4 departments then it is good idea to delegate separate sub domain to each department. Deciding name of the sub domain is also important for domain management. It is recommended to give name according to organization name and geographical location rather than any arbitrary name. How many sub domains? It is also important to figure out number of sub domain you want to create. Avoid any unnecessary delegation.

3.6.1 Domain configuration for parenting

Let's delegate some portion of **cust.com** domain to **dgrdn.cust.com** domain.

First to create the DNS sub domain, we need root access to the our parent domain **cust.com** to edit the parent domain's zone file and to reload zone data file once changes are completed.

First define zone file in named.conf.local configuration file.

Edit following configuration in /etc/bind/named.conf.local

```
zone "cust.com"{
    type master;
    file "/etc/bind/zones/cust.com.db";
};
```

Add more configuration as needed. These are minimum configuration required to delegate subdomain.

Open file by using sudo gedit /etc/bind/zones/cust.com.db and edit following configuration. Configuration shown in following example with the bold letter is added to define sub domain's nameserver IP address and authoritative server. Other configurations are just regular configuration to define zone.

Here bold configuration is NS record of name server residing in sub domain server. It glues NS records of subdomain server to NS records of domain nameserver.

```
$TTL 86400
@    IN  SOA   ns.cust.com. pakodi.cust.com. (
                41           ; Serial
                604800        ; Refresh
                86400         ; Retry
                2419200       ; Expire
                86400)        ; Negative Cache TTL
;
cust.com.      IN  NS    ns.cust.com.
cust.com.      IN  A     192.168.1.4
ns             IN  A     192.168.1.4
```

```
dgrdn.cust.com. IN  NS  ns.dgrdn.cust.com.
ns.dgrdn.cust.com. A  192.168.1.3 ; glue record
```

By declaring above configuration authority of sub domain name server **ns.dgrdn.cust.com** is given to **ns .cust.com** name server.

3.6.2 Configuration for sub domain

Edit following configuration file in named.conf.local file

```
zone "dgrdn.cust.com"{
    type master;
    file "/etc/bind/zones/dgrdn.cust.com.db";
};
```

Add more configurations in named.conf.local file according to your requirement like configuration for reverse lookup or dynamic DNS configuration or TSIG configuration. Complete configuration is declared in chapter 5

Zone data file for sub domain

```
$TTL 86400
@ IN SOA ns.dgrdn.cust.com. pakodi.dgrdn.cust.com. (
        42           ; Serial
        604800       ; Refresh
        86400        ; Retry
        2419200      ; Expire
        86400 )      ; Negative Cache TTL
;
dgrdn.cust.com.   IN  NS    ns.dgrdn.cust.com.
ns              IN  A     192.168.1.3
```

Before you reload the parent's domain name server reload child nameserver first and make sure it works fine. Use following command to reload the nameserver

```
# rndc reload
```

Now sub domain server is ready to accept query from parent domain and can talk to outside world through parent domain server.

3.7 DNS features

3.7.1 Control and rndc

Controls statement determines how the nameserver listens for control messages. BIND 9 support port specifications. By default nameserver uses port 953 for listening.

```
controls {
    inet * allow { any; } keys { "rndc-key"; };
};
```

key specified in the keys sub statement must be defined in a key statement:

```
key "rndc-key" {
    algorithm hmac-md5;
    secret "YBt4XkZhgLjFKR4zcUVZuQ==";
};
```

It is good to use key by include statement in named.conf.local file to keep key unreadable from world

```
include "/etc/rndc.key";
```

To use rndc, you need to create an rndc.conf file to tell rndc which authentication keys to use and which nameservers to use them with. rndc.conf usually lives in /etc. Use rndc-confgen command in command line to generate rndc.conf and rndc-key or use rndc-

confgen -a to set up a rndc.key. It will create rndc.key file in /etc directory so no need to modify named.conf.local file for key configuration.

Just run the following command in command line prompt and output like:

```
user@appl-user-desktop:/etc$ rndc-confgen
# Start of rndc.conf
key "rndc-key" {
    algorithm hmac-md5;
    secret "YBt4XkZhgLjFKR4zcUVZuQ==";
};

options {
    default-key "rndc-key";
    default-server 127.0.0.1;
    default-port 953;
};
# End of rndc.conf

# Use with the following in named.conf, adjusting the allow list as needed:
# key "rndc-key" {
#     algorithm hmac-md5;
#     secret "YBt4XkZhgLjFKR4zcUVZuQ==";
# };
#
# controls {
#     inet 127.0.0.1 port 953
#         allow { 127.0.0.1; } keys { "rndc-key"; };
# };
# End of named.conf
user@appl-user-desktop:/etc$
```

As stated second half of this file is part of /etc/named.conf. Move these lines to named.conf file. Change configuration as needed.

Following command are used with rndc frequently.

Reload: Reload configuration file and zones data file.

Reload zone [class [view]]: Reload the given zone.

Refresh zone [class [view]]: Schedule zone maintenance for the given zone.

Retransfer zone [class [view]]: Retransfer the given zone from the master.

Stats: Write server statistics to the statistics file.

Query log: Toggle query logging.

Dumpdb [-all-cachel-zone] [view]: Dump the server's caches (default) and/or zones to the dump file for the specified views. If no view is specified, all views are dumped.

Flush: Flushes the server's cache.

3.7.2 DDNS

Now a day's most of the IP address assign to computer is dynamically. When you connect computer to Internet or private network, ISP assigns IP address dynamically to computer. This address is change frequently by ISP. Temporary IP addresses don't help to run web server or FTP server. When web server restart it gets new IP address and it is different from previously assigned. Every time when web server gets new IP address from DHCP server, DNS server should point new IP address for assign FQDN for web server.

Traditional methods of updating Zone data file of DNS server is manually editing the zone file and then restart the name server to reload the changes. It is painful for network administrator to update FQDN and IP address of client in DNS server once client get new IP.

Addition and deletion to zone data file by manually is unacceptable when updates available in bulk. When volume of changes for zone data file is too much then traditional method of updating zone is not better solution. There is some automatic technique needed to update zone data files with out any human intervention. To solve problem IETF introduce mechanism, called DDNS (dynamic DNS), a standard mechanism for managing the identity of dynamic networks created through the DHCP server. Dynamic DNS updates a DNS server's zone data files with new or changed records for IP addresses without the need for human intervention. The Dynamic update mechanism useful to add and delete resource records of zone data file. Updaters can add or delete some part or entire resource record of domain name. It can add or delete domain name of host, IP address etc.

The Dynamic update is only possible by using DHCP servers that assign IP addresses automatically to computers and send update to the DNS server to register the resulting name-to-address and address-to-name mappings. Detailed configuration of DHCP server for dynamic update is explained in chapter 4

3.7.2.1 Dynamic Update and Serial Numbers

DHCP server is responsible to update zone data file dynamically. In addition to host name and IP address, it also update (increment) the serial number in zone data file. Updated serial number is useful to update slave server. Nameserver doesn't responsible to increment the serial number for each dynamic update.

3.7.2.2 Dynamic Update and The journal file

The Dynamic updates can not be stored directly in zone data file but first stored to journal file. This journal file created automatically when first time nameserver gets update from other name server or DHCP server. It is created in directory where zone's datafile is stored. .jnl extension is appended to the zone data file name if any other name is not specified.

The journal file can not be edited manually. After 15 minutes journal file write the data to zone data file. Instead of updating zone data file immediately journal file waiting for 15 minutes to update any large zone data file. If server is crash before updating zone data file from journal file then next time when server restart journal file compare update with zone file and if zone file is not updated then it transfer the update. It is not practical to edit zone data file which is updated by dynamic update because there is no any

guaranty that files in the zone data file is most updated from journal file. The only way to ensure that the zone file of a dynamic zone is up to date is to run rndc stop.

Disable dynamic updates to the zone using rndc freeze zone command to edit zone file manually. This will also remove the zone's .jnl file and update zone data file with most updated data. After editing zone data file run rndc thaw zone command to reload zone data file and start dynamic update again.

3.7.2.3 BIND configuration for Dynamic updates.

By default BIND 9 name servers do not allow dynamic updates to authoritative zones. To configure zone data files to accept dynamic updates use allow-update or update-policy sub statement to the zone statement.

allow-update statement matches the key stored in specified file. Here in following example to update *dbserver10.com* zone the updater must have the key signed by TSIG. When DHCP server sends update to BIND server, it sends TSIG key also. BIND server matches the TSIG key sent by DHCP server with its own key stored in separate file. Once the key is matched it will allow updating. If you want to update both forward and reverse map of zone data file then specify allow-update statement with rndc-key in both file.

Edit the following configuration in named.conf file.

```
controls {
    inet 127.0.0.1 allow {127.0.0.1; 192.168.1.3; 192.168.1.2;} keys {"rndc-key";};
};
```

“Controls”: This statement tells the bind server, “Hey these are nice people allow them to update zone data file”

“inet 127.0.0.1”: where 127.0.0.1 is IP address of local BIND server.

“allow {127.0.0.1; 192.168.1.3; 192.168.1.2;}”: IP address specified in brace are allowed to update BIND server. You can add more IP address if you want. In example IP address 192.168.1.3, 192.168.1.2 and 127.0.0.1(local server) are allowed to update BIND server.

“Keys {“rndc-key”;};”: This is the name of a key that is generated to authorize only those computer who has same key allowed to modify the DNS.

The key specified in the keys substatement must be defined in a key statement:

```
key "rndc-key" {
    algorithm hmac-md5;
    secret "wwmyGnn59ZRXXxdCWO5ILg==";
};
```

Above key statement directly goes to named.conf, if named.conf file is world-readable then it's safer to put it in a different file that's not world-readable and include that file in named.conf:

```
include "/etc/rndc.key";
```

Add the following configuration in named.conf.local file

```
key "rndc-key" {
```

```

algorithm hmac-md5;
secret "wwmyGnn59ZRXXdCWO5ILg==";

zone "dgrdn.cust.com"{
type master;
file "/etc/bind/zones/dgrdn.cust.com.db";
allow-update { key "rndc-key"; };
};

zone "in-addr.arpa"{
type master;
notify no;
file "/etc/bind/zones/in-addr.arpa";
allow-update { key "rndc-key"; };
};

```

3.7.3 DNS notify:

When the primary name server is updated manually by editing zone data file or automatically by dynamic DNS then its serial number is incremented. When a primary nameserver notices that the serial number of a zone has increased then it informs all the slave nameservers for designated zone. After restarting primary nameserver, it sends notification message to all slave servers come under that primary name server. Once slave server receive NOTIFY message from primary server it send reply to primary server that I received notification message so stop to sending any more notification message. Now slave server compare the serial number of zone data file, if it is higher than it self then it will update zone data file from primary server.

In BIND 9, DNS NOTIFY is on by default, it can be turn off by using following global statement in named.conf.local file:

```

options {
    notify no;
};

```

If you don't want to define notify statement globally then it can also define for particular zone. Here in following example *dgrdn.cust.com* zone will not allow to send useless notification message to all slave servers comes under the primary server.

```

zone "dgrdn.cust.com" {
    type master;
    file "/etc/bind/zones/dgrdn.cust.com.db";
    notify no;
};

```

Here Zone NOTIFY configuration overrides the global configuration.

3.7.3.1 also-notify

The *also-notify* is used for only 'type master' server and it defines a list of IP address whom to send NOTIFY message. Doesn't matter whether specified IP address server is under the primary server or not.

Also-notify statement overrides any global configuration for particular zone.

```
options {
    also-notify {192.168.1.1; 192.168.1.2;} // all zones
};

zone "cust.com"{
    also-notify {192.168.1.3;} // only this host
};

zone "cust.net"{
    notify no; // no NOTIFY for zone
};
```

Below substatements tell the nameserver to send NOTIFY messages only to the slave at 192.168.1.1:

```
options {
    also-notify {192.168.1.1;};
    notify explicit;
};
```

3.7.3.2 allow-notify

The *allow-notify* substatement is used to tell slaves to accept NOTIFY messages from specified nameservers by IP address other than just the configured master nameservers for a zone:

```
options {
    allow-notify {192.168.1.1;} // will allow 192.168.1.1 to send NOTIFY message
};
```

If it is defined in options substatement, allow-notify affects all slave zones and when define in zone substatement, allow-notify overrides global configuration for particular zone.

3.7.4 Incremental Zone Transfer (IXFR)

The Traditional method of updating the slave server (secondary) from the primary nameserver is work like this. Once primary server's zone data file is updated manually it sends notification message to slave includes serial number. Slave compares its serial number with primary server's serial number. If primary server's serial number is higher than slave's serial number than primary server transfer the entire zone data file to slave's zone data. Even if minor update in primary server's zone datafile, slave server force primary to send whole zone data files which keeps primary server busy to sending zone data and consume most of the bandwidth if zone data file is bigger. The way of updating slave server is not feasible when very small changes in primary server like change in IP address of resource records taken place. This method of updating slave server is not acceptable when primary server is updated frequently by DDNS.

The IXFR solve the above problem by sending only updated data in primary server to the slave instead of sending entire zone data. The IXFR works only with Dynamic updates.

The IXFR is work well if zone data is changed automatically by dynamic update instead of by hand. Dynamic update also updates serial number in zone data file and it is known as serial version. Primary server sends serial version number with notification. Slave compare serial version sent by primary with it self and computes the difference in version. Now slave request to primary to send only data updated after difference in version. But if you want to update by hand then slave server has to compute difference between updated zone and previous zone's serial number difference not version difference.

To update master zone manually and transfer update to the slave server use turn on ixfr-from-differences substatement. It can be used within an options or zone statement.

```
options {  
    directory "/var/named";  
    ixfr-from-differences yes;  
};  
also stop update dynamically by using rndc freeze zone  
#rndc freeze zone
```

To start dynamic update use rndc thaw zone command

```
# rndc thaw zone
```

3.7.4.1 BIND 9 IXFR Configuration

By default BIND 9 is configured for IXFR. Use following sub statement to turn off the IXFR for particular server. *provide-ixfr* is yes by default.

```
server 192.168.1.1 {  
    provide-ixfr no;  
};
```

The *provide-ixfr* can also be used as an options sub statement; in that case it applies to all slaves that don't have an explicit provide-ixfr sub statement of their own in a server statement.

3.7.5 TSIG

Type of security implemented by one-way hash mathematical function for securing DNS message known as transaction signatures (TSIG). It is used to send secure DNS messages for zone transfer, notify, recursive query messages and dynamic updates. When the nameserver sends DNS messages TSIG record add with messages. It add cryptographic key with messages. Receiver must have same key to get information from sender nameserver.

3.7.5.1 Automatic Generation of TSIG

The following command will generate a 128-bit (16 byte) HMAC-MD5 key. Longer keys are better, but shorter keys are easier to read.

```
dnssec-keygen -a hmac-md5 -b 128 -n HOST server1-server2.
```

The key is in the file Kserver1-server2.+157+00000.private. After opening file we will get following secret key:

Key: La/E5CjG9O+os1jq0a2jdA==

3.7.5.2 Manual Generation

```
# mmencode
```

By using above command we can generate key which we want.

The *rndc tool* also support to generate TSIG key as explained in 3.7.1

3.7.5.3 Configuring TSIG

If we want to use TSIG for secure zone transfer between primary and secondary then we have to use secret key in both primary server and secondary server.

```
key server1-server2. {
    algorithm hmac-md5;
    secret " La/E5CjG9O+os1jq0a2jdA==";
};
```

Instructing the Server to Use the Key

```
server 192.168.1.1 {
keys { server1-server2. ;};
```

If the 192.168.1.2 is IP address for server2 then when server1 sends dynamic update or zone transfer to server 2, server 1 force to sign DNS message by TSIG. For reliable communication between server1 and server2 both have common key.

3.8 Diagnostic Tools

The dig, host, and nslookup programs are all command line tools. They are used to querying nameserver manually. Outputs of all the tools are different from each other.

3.8.1 nslookup

The *nslookup* (Name System Lookup) is a tool for querying a domain name server in order to get information regarding a domain or host, and diagnosing any configuration problems that may have arisen on the DNS.

When it is used without any arguments, the command nslookup displays the name and IP address of the primary domain name server, as well as a command prompt for making queries.

Here in example I used nslookup without any argument

```
Microsoft Windows [Version 6.0.6001]
Copyright (c) 2006 Microsoft Corporation. All rights reserved.
```

```
C:\Users\BHATUDI>nslookup
Default Server: ns1.telus.com
Address: 192.168.1.254
```

```
>
```

Type the domain name at the prompt to get information of particular domain name. We can also get information of domain name by using:

```
> nslookup host.name
```

By default, the nslookup queries the primary domain name server installed on the machine. However, it is also possible to query a particular DNS server by specifying it, preceded by a minus sign, after the command:

```
>nslookup host.name -server.name
```

You can also change the query mode for nslookup by using the argument set:

The nslookup in general returns A or PTR records but specific options can be used to override the default.

Example:

```
Microsoft Windows [Version 6.0.6001]
Copyright (c) 2006 Microsoft Corporation. All rights reserved.
```

```
C:\Users\BHATUDI>nslookup
Default Server: UnKnown
Address: 192.168.1.254
```

```
> www.example.com
Server: UnKnown
Address: 192.168.1.254
```

Non-authoritative answer:

```
Name: www.example.com
Address: 208.77.188.166
```

```
Returns the A record for www.example.com using the default DNS server
Format 1 - Reverse MAP IP lookup
```

```
> 208.77.188.166
Server: UnKnown
Address: 192.168.1.254
```

```
Name: www.example.com
Address: 208.77.188.166
```

>

It Returns the PTR record for 208.77.188.166 using the IN-ADDR.ARPA domain hierarchy.

Use nslookup –all command to list all default values used by nslookup, including the DNS server.

```
Microsoft Windows [Version 6.0.6001]
Copyright (c) 2006 Microsoft Corporation. All rights reserved.
```

```
C:\Users\BHATUDI>nslookup -all
Default Server: (null)
```

```
Set options:
nodebug
defname
search
recurse
nod2
novec
noignoretc
port=53
type=A+AAAA
class=IN
timeout=2
retry=1
root=A.ROOT-SERVERS.NET.
domain=domain.invalid
MSxfr
IXFRversion=1
srchlist=domain.invalid
```

```
Default Server: UnKnown
Address: 192.168.1.254
```

```
v
```

3.8.2 dig

The domain information groper (dig) is the most versatile tool and has all option available for lookup. The dig command is available in two mode 1) command line and 2) a batch mode. As compare to nslookup dig is more powerful. All query options are accessible from the command line.

Usage

```
dig [@server] domain [query-type] [query-class] [+query-option]
[-dig-option] [%comment]
```

Following are most useful option.

```
get the A record for any record without a label
# but will always return the SOA record for the domain
dig example.com
# get the MX record for the domain
dig example.com mx
```

```
# get the A record for the host
dig www.example.com
# get all domain records if allowed
dig example.com axfr
# get all records with no label for the domain
dig example.com any
# returns SOA, NS, MX and domain SPF if defined
```

dig's Output Format

```
user@appl-user-desktop:~$ dig www.ualberta.ca

;; <>> DiG 9.6.1 <>> www.ualberta.ca
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 30451
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 3, ADDITIONAL: 3

;; QUESTION SECTION:
;www.ualberta.ca.      IN  A

;; ANSWER SECTION:
www.ualberta.ca.    5    IN  CNAME  web1.srv.ualberta.ca.
web1.srv.ualberta.ca. 5    IN  A    129.128.98.86

;; AUTHORITY SECTION:
srv.ualberta.ca.    5    IN  NS   NOM.ualberta.ca.
srv.ualberta.ca.    5    IN  NS   MENAIK.CS.ualberta.ca.
srv.ualberta.ca.    5    IN  NS   NAME.ualberta.ca.

;; ADDITIONAL SECTION:
NAME.ualberta.ca.    5    IN  A    129.128.5.233
MENAIK.CS.ualberta.ca. 5    IN  A    129.128.4.241
NOM.ualberta.ca.    5    IN  A    129.128.76.233

;; Query time: 42 msec
;; SERVER: 192.168.253.2#53(192.168.253.2)
;; WHEN: Wed Jul 15 10:36:38 2009
;; MSG SIZE rcvd: 181
```

```
user@appl-user-desktop:~$
```

Example of Host Query

```
# simple host lookup - defaults to an A RR
dig www.example.com
```

```
# or could have been written as - order important
dig www.example.com a
# identified option format - order not important
dig -t a www.example.com
# use the dns at 192.168.2.224 for the query
dig @192.168.2.224 www.example.com a
# use the dns at ns1.example.com for the query
dig @ns1.example.com www.example.com a
# reverse map query - returns PTR RR
dig -x 192.168.2.224
```

3.8.3 Host

It is used to see domain name and internet address. It is also used with options.

Usage

```
host [-aCdlnrsTwv] [-c class] [-N ndots] [-t type] [-W timeout] [-R
retries] [-m flag] [-4] [-6] hostname [server]
```

Output of host command

```
user@appl-user-desktop:~$ host 129.128.98.86
86.98.128.129.in-addr.arpa domain name pointer web1.srv.ualberta.ca.
user@appl-user-desktop:~$ host www.ualberta.ca
www.ualberta.ca is an alias for web1.srv.ualberta.ca.
web1.srv.ualberta.ca has address 129.128.98.86
user@appl-user-desktop:~$
```

3.9 Administrative tools

The *named-checkconf* and *named-checkzone* are available in BIND 9 to check configuration. These tools reside in /usr/local/sbin.

The named-checkconf is used to check configuration file for syntax errors.

The named-checkzone checks a zone file for syntax errors.

3.9.1 named-checkconf

The named-checkconf checks /etc/named.conf by default. If named.conf file is not configured then it gives message like named.conf file not found and if it has error then it show error line and other information.

```
user@appl-user-desktop:/etc$ named-checkconf
none:0: open: /etc/named.conf: file not found
```

If you have an error, named-checkconf displays an error message, such as this one:

```
/etc/named.conf:14: zone '!': missing 'file' entry
```

```
If there are no errors, you won't see any output.
```

3.9.2 named-checkzone

Run named-checkzone for each of zone files:

```
user@appl-user-desktop:/etc$ named-checkzone dgrdn.cust.com.db
```

zone dgrdn.cust.com/IN: loaded serial 4

OK

Chapter 4 DHCP3 server

4.1 Why DHCP?

Some years before the computer networks were very small with few fixed computers. It was easy for the network administrator to configure TCP/IP one by one. By the time technology was growing and demand of the computers increase day by day. New computers added to existing networks as a result networks become large. Everyday thousands of new computers and mobility devices are adding in network. Businesses and academic institute are growing more, demand of computers are increased. It is not easy for network administrator to configure TCP/IP of all new users one by one. To communicate for all new computers with existing network's devices some kind of automatic plug and play system was required to configure TCP/IP. For solution the IETF developed the DHCP server. The DHCP server provides automated, managed configuration of computers and other devices that use TCP/IP. Through DHCP, a network administrator can assign a network address, subnet mask, DNS server address, domain name and default gateway etc. automatically to new newly added computer to network.

4.2 DHCP operation

The Dynamic Host Configuration Protocol (DHCP) is a network protocol that assigns IP address automatically to the network devices to communicate with other existing network devices. It replaces the network administrator's task. It provides configuration required by the TCP/IP networking protocol. The DHCP server runs on a central server that dynamically assigns an IP address to individual PCs or workstations. The Dynamic Host Configuration Protocol (DHCP) is a set of rules for dynamically allocating IP addresses and configuration options to client on a network. The DHCP server worked on server/client model. When new client added in network it gets all necessary configurations to communicate with other network devices automatically. Network parameters like network IP address, default gateway, DNS address, domain name, subnet mask etc are pre configured by network administrator in DHCP server.

When any DHCP enable computer or any other network devices connect to network it sends broadcast message to the DHCP server. The DHCP server interpret message and figure out the origin of network segment of client and accordingly decide IP address, default gateway, subnet mask, DNS server address, domain name, lease time etc for clients. Client and the DHCP server exchange total 4 messages to assign IP address and other network parameter. Thus, the DHCP server acts as the network administrator's agent for managing the configurations of DHCP clients. By DHCP server we can assign IP addresses and the configuration of other TCP/IP protocol parameters in whatever way is appropriate for the network and the organization. Administrator can decide allocation of IP address. It might be any one or combination from following option. It may be dynamic allocation or automatic allocation or static allocation. Administrator will decide according to requirement of network.

4.3 Installation of DHCP server

There are two different methods available to configure DHCP server

- 1) Download the latest DHCP server version from <http://ftp.isc.org/isc/dhcp/> website

```
user@appl-user-desktop:~/Desktop$ tar zxvf dhcp-4.1.1b1.tar.gz
```

UNPACKING DHCP server:

To build the DHCP Distribution, unpack the compressed tar file using the tar utility and the gzip command:

```
user@appl-user-desktop:~/Desktop$ gunzip dhcp-4.1.1b1.tar.gz
user@appl-user-desktop:~/Desktop$ tar xvf dhcp-4.1.1b1.tar
```

CONFIGURING

Now, cd to the dhcp-4.1.1b1 subdirectory just we created and configure the source tree by typing:

```
user@appl-user-desktop:~/Desktop/dhcp-4.1.1b1$ ./configure
```

BUILDING IT

Once you've run configure, just type "make"

```
user@appl-user-desktop:~/Desktop/dhcp-4.1.1b1$ make
```

INSTALLING THE DHCP DISTRIBUTION

Once DHCP Distribution is successfully built to build, install it by typing "make install" command.

Use following command to see version of installed DHCP server

```
user@appl-user-desktop:~/Desktop/dhcp-4.1.1b1$ dhcpcd -t
Internet Systems Consortium DHCP Server 4.1.1b1
Copyright 2004-2009 Internet Systems Consortium.
```

2) Vendor-Supplied Versions of the ISC DHCP Distribution

If DHCP server is shipped with operating system then just type following command in command prompt:

```
sudo apt-get install dhcp3-server
```

Ubuntu 7.10 is shipped with DHCP version 3.0.3.

Default configuration file is located at /etc/dhcp3/dhcpd.conf.

To Specified interface edit following configuration in /etc/default/dhcp3-server file. Specify interface will listen DHCP server. By default it listens to eth0.

```
INTERFACES= "eth1"
```

4.4 Configuring a DHCP server:

Configuration of the DHCP server is depending on the network topology. Before starting configuration of the DHCP server first design the network architecture. Once you have list of network requirement then it is easy to configure. List all the features you want to implement by DHCP server because now a day wide ranges of features available. To configure DHCP server open etc/dhcp3/dhcpd.conf file.

First I will explain use of syntaxes individually. Once we have idea about all syntaxes then after I will put all configurations together for dhcpd.conf file

4.4.1 Authoritative configuration

The DHCP server must know whether it is authoritative or not. The DHCP server must be authoritative if it is controlled by network administrator. Most of the DHCP servers are authoritative. If the DHCP server is not authoritative, it will not tell the clients when network is changed, so the clients that change networks will not get new IP addresses. To configure the ISC DHCP server to be authoritative write an authoritative.

Statement syntax:

```
authoritative;
```

4.4.2 Individual Subnet configuration

The DHCP server configuration file needs at least one subnet declaration. You can specify more than one subnet as per network requirements. If the network is small and it has just one subnet 192.168.1.0 and subnet mask 255.255.255.0 then it can be declare by using following configuration syntax:

```
subnet 192.168.1.0 netmask 255.255.255.0 {  
}
```

This declaration simply informs the server that the 192.168.1.0 subnet exists. By declaring this subnet the DHCP server is capable to process request done by DHCP enable client. If the DHCP client sends broadcast request to DHCP server and ask to assign particular IP address 192.168.2.1 then the DHCP server realise that the client request for wrong subnet's IP. The DHCP Server sent back DHCPNAK message to the client stating requested IP address is not valid.

4.4.3 Address Allocation

```
subnet 192.168.1.0 netmask 255.255.255.0 {  
range 192.168.1.61 192.168.1.67;  
}
```

The Network administrator must define the IP address range for clients. Client will get valid IP address from the defined range. In the above example our subnet is 192.168.1.0. So according to subnet declaration address range must be with in the defined subnet.

IP address 192.168.1.0 and 192.168.1.255 are used for broadcast address. So exclude these two IP address from range. Here address 192.168.1.61 to 192.168.1.67 is available for client.

Network administrator will define range of IP address according to network topology. It may be some computer needs static address from the above IP addresses range. Some will get any IP dynamically when reconnect to network.

4.4.4 Client Option Configuration

Only IP address assigned to client is not enough to do anything useful. It also needs more information like the default gateway and the DNS server address to reach Internet.

It can be defined by using following syntax:

4.4.4.1 DNS option

Clients need address of DNS server for name resolution. The domain name option specifies the IP address or name of DNS server for resolving names in the Domain Name System (DNS).

```
option domain-name-servers 192.168.1.4;
```

4.4.4.2 Router option

The Router option is used to define default gateway for client.

```
option routers 192.168.1.1;
```

4.4.4.3 Broadcast option

Some clients also require the broadcast address option, although many compute it themselves from the defined subnet of IP address and subnet mask.

```
option broadcast-address 192.168.1.255;
```

4.4.4.4 Subnet mask option

The Network administrator can also specify the subnet mask for the clients. If he forgets to specify then the DHCP server computes subnet mask using subnet declaration. Below example show you how to declare subnet-mask option.

```
option subnet-mask 255.255.255.0;
```

4.4.5 The server-identifier Statement

```
server-identifier IP address;
```

The Server identifier option gives IP address of the DHCP server to client. Client use this address to send DHCPREQUEST message to the DHCP server. If the DHCP server is configured on multi interface machine then client uses the IP address of the interface on which it received the DHCP message. Use of the server-identifier statement is not recommended.

4.4.6 The default-lease-time Statement

```
default-lease-time time;  
default-lease-time 300;
```

The default-lease-time statement specifies the duration of the lease time that the DHCP server assigns to client for assigned IP address if client does not request for specific lease time. Duration of time in seconds and by default lease time is 12 hours.

4.4.7 The max-lease-time Statement

```
max-lease-time time;  
max-lease-time 300;
```

It specifies the maximum duration of the lease that the DHCP server assigns to the DHCP client if default lease time is not defined. If the defined max lease time is 1000 seconds and client want lease time for 2000 seconds then the DHCP server do not allow client to lease for more than 1000 seconds. Here time is the maximum duration of time a client will get for lease in Seconds and by default lease time is 24 hours.

4.4.8 The min-lease-time Statement

```
min-lease-time time;  
min-lease-time 300;
```

Client must accept minimum lease for specified time. If client want lease for 100 second and minimum lease time specified by DHCP server is 200 seconds then client must take minimum lease of 200 seconds. By default minimum lease time is 0 second. Basically DHCP server is forcing client to sign lease for at least minimum time specified in statement.

4.4.9 The ddns-domainname Statement

```
ddns-domainname "name";  
ddns-domainname "dgrdn.cust.com";
```

The DHCP server will use specified domain name to send dynamic update on behalf of client. DHCP server will update A record for client in specified domain name. This also known as forward map files for zone data file.

Ex. ddns-domainname "dgrdn.cust.com";

As shown in syntax DHCP server will update client's A record in domain "dgrdn.cust.com". DHCP server controlled the update for DNS server.

4.4.10 The ddns-rev-domainname Statement

```
ddns-rev-domainname "name";  
ddns-rev-domainname "in-addr.arpa";
```

Function of this statement is same as "ddns-domainname" syntax. Instead of updating client's A record it will update PTR record. It will useful for reverse mapping of

client. If any client send query by IP address then updated PTR record help to solve query.

It gives correspond host name for queried IP address.

4.4.11 ddns-update-style

ddns-update-style interim;

By default, clients send DNS updates to DNS server. To send the Dynamic updates to the DNS server by the DHCP server use “ddns-update-style interim” statement in the configuration file.

4.4.12 The ddns-ttl Statement

ddns-ttl ttl;

ddns-ttl 300;

The ddns-ttl statement tells the DHCP server to send specified TTL value to DNS server with dynamic update. A records of Client updated in DNS server with specified TTL. ttl is specified as a number of seconds.

4.4.13 NTP server

The NTP server option gives the addresses of Network Time Protocol servers available to the client. All clients will use same clock provided by NTP server to synchronise application of the DHCP server

4.4.14 Mixing Static and Dynamic Allocation

It is good practice to assign the fixed IP address for clients who run web server or FTP server and dynamic IP address for others. The Network administrator decides, which clients need static and dynamic IP address? If fixed address host is restarted, it will get automatically same IP address from DHCP server every time.

Here in following example shows you how to assign fixed address to host:

```
subnet 192.168.1.0 netmask 255.255.255.0 {  
range 192.168.1.1 192.168.1.251;  
option routers 192.168.1.254;  
}  
host dhcp-server.genericstartup.com {  
fixed-address 192.168.1.252;  
hardware ethernet 2U:AH:78:11:F9:14;  
}
```

In above example host with Ethernet address 2U:AH:78:11:F9:14 always get 192.168.1.252 IP address even if rebooting.

4.5 DHCP Messages

All Dynamic Host Configuration Protocol (DHCP) messages include a FIXED and a VARIABLE format section. The fixed format section consists of several fields that are the same in every Dynamic Host Configuration Protocol (DHCP) message. The variable format section in the Dynamic Host Configuration Protocol (DHCP) contains

"OPTIONS", which carry additional configuration parameters. Variable format of message is change according to available option in message.

Fixed format message carry following information

- ❖ Control information
- ❖ IP address of server
- ❖ Network and link layer address of client

Obtaining an Initial Configuration

Initially when the DHCP enabled client connect to the network it doesn't have any valid IP address. If it has no valid IP then it is in INIT state. To get first time IP address from DHCP server, client and DHCP server have to exchange four type of message to pass all necessary configurations to the client.

4.5.1 The DHCPDISCOVER Message

First time when client doesn't have any IP address it broadcast DHCPDISCOVER message to all DHCP server in the network to get IP address and other TCP/IP configuration. Client use 0.0.0.0 IP address as source address and 255.255.255.255 as broadcast address. The DHCP message type option identifies this message as a DHCPDISCOVER message, and the client does not include additional DHCP options.

4.5.2 The DHCPOFFER Message

Once the DHCP server receive DHCPDISCOVER message from the client, it start to figure out IP address for client. It also determines other information for client like default gateway, subnet mask, DNS server, domain name by using configuration file. Once server has all the information, it will put in DHCPOFFER message and send back to client.

The DHCP Server send this message to the Ethernet broadcast address FF:FF:FF:FF:FF:FF. The DHCP server use transaction identifier sent by client to include in DHCPOFFER massage.

4.5.3 The DHCPREQUEST Message

Once client receive the DHCPOFFER message from the DHCP server it send back DHCPREQUEST message. Client use link layer address FF:FF:FF:FF:FF:FF and IP address 255.255.255.255 as broadcast address. Client requests the DHCP server to send information about IP address and other configuration parameters which server offered in DHCPOFFER message.

4.5.4 The DHCPACK Message

Once server received DHCREQUEST message, it reply client with the DHCPACK message. The DHCPACK message includes final configuration parameter for the DHCP client. The DHCP server first verifies offered IP address is available or it is used. Once it is available for client, it records and sends to the client with DHCPACK message.

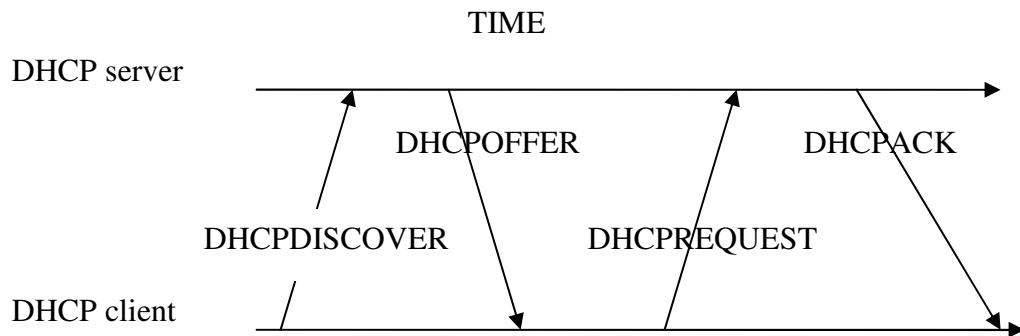


Figure 4.1 Exchanged messages between a client and a server to assign an initial address

When client receive DHCPACK message, it will remove the configuration from message and use it to configure the TCP/IP protocol. Client use the same IP address for all communication until lease is expired. Above diagram show you how DHCP server and DHCP client exchange the message.

4.6 DHCP for DDNS update

Traditionally DNS server is updated by the network administrator but now DHCP enable client and DHCP server is capable to update the DNS database. They can directly change the A and PTR records of client without any intervention of network administrator.

The DHCP server has important role to update DNS server dynamically. DHCP server needs to be configured correspond to the DNS server for the dynamic update. Once the DHCP server assign IP address to the client it will get more basic configuration for TCP/IP communication like default gateway, subnet mask, DNS server address, domain name etc and will write lease time assign to client in lease file. On the other side DHCP send update to primary name server defined in configuration. DHCP server will update the forward and reverse map of the zone data file. The DNS server will allow the DHCP server to update zone data file only after once both server's TSIG key is matched. For secure update of Dynamic DNS, BIND server has feature to create secure key by using algorithm known as TSIG key. BIND server will not allow DHCP server to update zone if TSIG key is unmatched. The BIND server is not store update in zone data files directly but it will create new journal file for temporary storage of data.

4.6.1 How FQDN formed?

The DHCP server append host name sent by client with specified domain name in zone data file. Suppose if client is sending hostname rack2 and the DHCP server is responsible to update domain dgrdn.cust.com of DNS server then resulting FQDN is rack2.dgrdn.cust.com.

4.6.2 DNS Update Security

To authorise only DHCP server and perhaps client to update DNS server and block any unauthorised update some kind of security policy must required.

There are two basic mechanisms available for authentication

- 1) IP address based authentication
- 2) Cryptographic authentication.

From these two schemes cryptographic authentication is easy to use and more protected than other. IP address based authentication is very hard to implement and less secure.

There are four different ways of using cryptographic authentication in DHCP updates:

- ❖ TSIG-based shared secret
- ❖ SIG(0)-based public key
- ❖ TKEY shared secret
- ❖ GSS-TSIG

The simplest of all these is TSIG. The DNS server generates the key by using algorithm. Install the same key in both the DHCP server and DHCP client whom you want to allow updating the DNS server. Client use same key when they send update to the DNS server. DNS server matches its own TSIG key with key received with dynamic update. If the key is match then DNS server allows updating otherwise deny. It is better to update DHCP client through the DHCP server because once key is changed then you need to update key in all the clients.

How to generate key is explained in chapter 3 BIND

Once you generate the key it will be like this:

```
key "rndc-key" {
    algorithm hmac-md5;
    secret "wwmyGnn59ZRXXdCWO5ILg==";
}
```

Responsibility for Performing DNS Updates

Before configuring the DHCP server network administrator must decide update policy for DNS server. Both DHCP server and DHCP client are able to update DNS server directly. Both can change A and PTR record of zone data file. Client must have its own FQDN to update DNS server. Also DHCP server must allow client to send update for DNS server. Client can use any arbitrary FQDN for updates.

4.6.3 Hostnames configuration in client

The Windows DHCP client and apple Macintosh always send host name to the DHCP server. But some operating system like linux never send host name. Network administrator has to configure hostname to update client in DNS server.

Edit /etc/dhcp3/dhclient.conf and set:
send host-name "<hostname>";
send host-name "rack2";

4.6.4 Configuring the Servers for updates

To configure DHCP server for dynamic update, DNS server is also required. Both server need to be configured for dynamic updated.

4.6.4.1 Configuring the DHCP Server to Do Updates

In order to update a domain name, the DHCP server must know the IP address of the primary nameserver and domain name of zone whom DHCP server sends update. If zone is configured for reverse mapping then reverse map zone data file name is also required. It is optional if you don't want to update reverse zone data file. To send only authorised update to DNS server by DHCP server the security TSIG security key is also required.

Following configuration is required in DHCP server to send updates to DNS server

```
key "rndc-key" {
    algorithm hmac-md5;
    secret "wwmyGnn59ZRXxXdCWO5ILg==";

zone dgrdn.cust.com{
primary 192.168.1.3;
key "rndc-key";
}

zone in-addr.arpa {
primary 192.168.1.3;
key "rndc-key";
}
```

In addition to above configuration following configuration is also required to configure DHCP server for Dynamic updates.

The BIND server does not do DNS updates by default. It needs to be enabled. Use ddns-update-style statement to enable the Dynamic DNS update. BIND server supports three styles: none, ad-hoc and interim. Interim style is recommended for DDNS updates.

```
ddns-update-style interim;
```

By default the BIND server allows clients to send update directly to DNS server. To update DNS server by DHCP server rather than by the client then use following statement in configuration.

```
deny client-updates;
```

The DHCP server should know which DNS zone file it will update. Use the ddns-domainname statement to specify domain name.

```
ddns-domainname "dgrdn.cust.com";
```

4.6.4.2 DNS Server configuration to Allow Dynamic Updates

You must configure the DNS server to allow updates from the DHCP server. This involves configuring the DNS server with the same key you used with the DHCP server and configuring the DNS server to allow updates using that key.

Following example of a DNS server configuration that corresponds to the DHCP server configuration shown in above example.

```
key "rndc-key" {
    algorithm hmac-md5;
    secret "wwmyGnn59ZRxxXdCWO5ILg==";
};

zone "dgrdn.cust.com"{
type master;
file "/etc/bind/zones/dgrdn.cust.com.db";
allow-update { key "rndc-key"; };
};

zone "in-addr.arpa"{
type master;
notify no;
file "/etc/bind/zones/in-addr.arpa";
allow-update { key "rndc-key"; };
};
```

Explanation of above configuration is given in chapter 3

4.7 DNS Record Removal

What will happen if the DHCP client suddenly crash or client's lease is expired or client's IP is assigned to other machine? Suppose webserver's A record is being used by any network services to locate and contact the webserver, but actually webserver is crashed and still some network service try to contact webserver by IP address available in DNS database.

The BIND server automatically deletes the A and PTR records from the DNS server when the lease is expire or when the client releases the lease with the help of DHCP server. Not all other DNS server is capable to remove A and PTR records from the DNS server.

4.8 Example of dhcpd.conf configuration file with DDNS

```
authoritative;
ddns-update-style interim;
ddns-updates on;
ddns-domainname "dgrdn.cust.com";
ddns-rev-domainname "in-addr.arpa";
option domain-name "cust.com";
option domain-name-servers 192.168.1.4;
allow unknown-clients;
key "rndc-key" {
    algorithm hmac-md5;
```

```
        secret "wwmyGnn59ZRXXdCWO5ILg==";  
    }  
  
default-lease-time 300;  
max-lease-time 300;  
log-facility local7;  
  
zone dgrdn.cust.com{  
primary 192.168.1.3;  
key "rndc-key";  
}  
  
zone in-addr.arpa {  
primary 192.168.1.3;  
key "rndc-key";  
}  
  
subnet 192.168.1.0 netmask 255.255.255.0 {  
range 192.168.1.61 192.168.1.67;  
option subnet-mask 255.255.255.0;  
option broadcast-address 192.168.1.255;  
option routers 192.168.1.1;  
default-lease-time 300;  
max-lease-time 300;  
}
```

Chapter 5 Project Design and Implementation

5.1 Introduction

This chapter will introduce a basic operation of the project. It will give a detail operation of how to assign an IP address from the DHCP server to the web server and its update in the BIND sub domain server. There is also in-depth detail about hardware and software used to implement the project. We will see how all the equipments are connected in the network and the responsibility of each device. It will also cover the detail Configurations of a Router, BIND domain and sub domain server, and DHCP server.

5.2 Hardware and software used to implement project

- ❖ 4 - Cisco router 2600 series
- ❖ 1 - Cisco switch 3700 series
- ❖ 1 - DHCP server running on Ubuntu Desktop
- ❖ 1 - Web server running on windows XP
- ❖ 2 - BIND DNS server running on Ubuntu system
- ❖ 1 - Windows client
- ❖ 1 - TFTP server

5.3 Network Design:

As shown in **Figure 5.1** 4-Cisco series 2600 routers are connected back to back using serial cables. The host name and position of all the routers and switches are explained below in table 5.1. All routers are accessed by a consol port. R3 is connected to switch (S1) by an Ethernet cable. The following table gives an idea about the hostname of the router, consol IP address and position of the router and switch in the network.

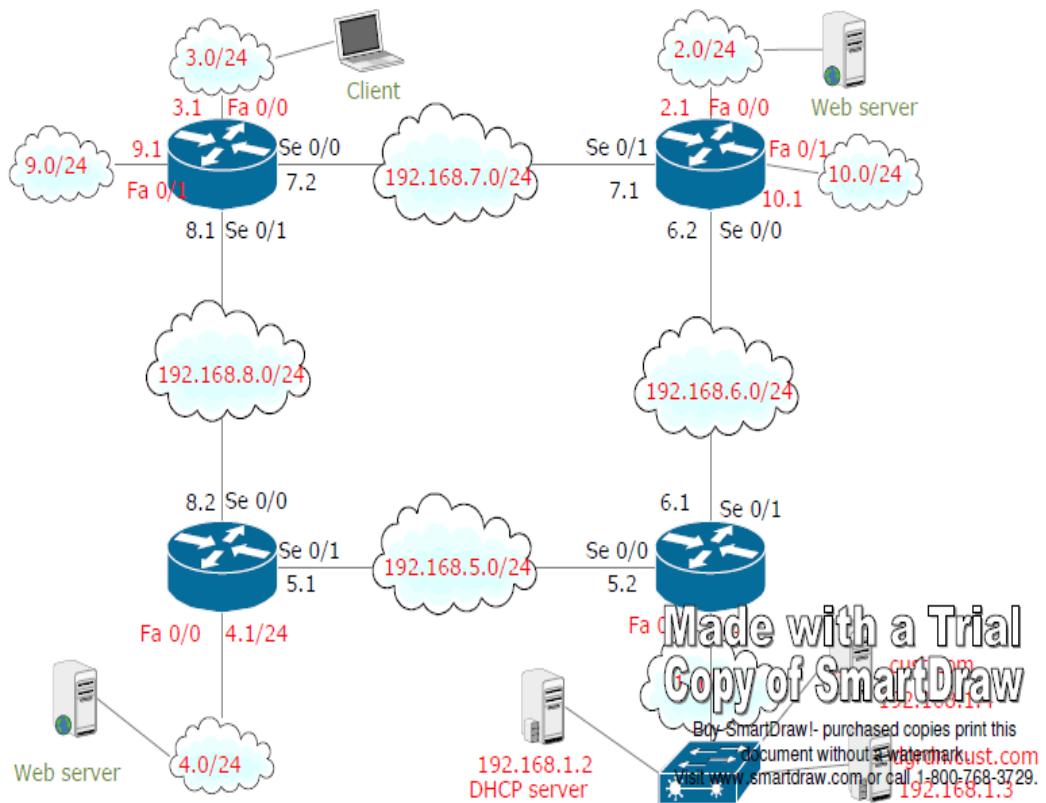


Figure 5.1 Network diagram

Naming and assign IP address of router

Hardware	Host name	Consol port IP address	Position in MINT lab
Router 1	R1	10.3.31.74	Rack4 top 2600
Router 2	R2	10.3.31.75	Rack4 mid 2600
Router 3	R3	10.3.31.76	Rack4 low 2600
Router 4	R4	10.3.31.94	Rack3 top 2600
Switch	S1	10.3.31.72	Rack4 3700

Table 5.1 Router information

10 different IP subnets are used to configure all the equipments.

Following IP subnets are used to configure router's serial link.

- ❖ 192.168.5.0/24
- ❖ 192.168.6.0/24
- ❖ 192.168.7.0/24
- ❖ 192.168.8.0/24

R1 and R2 are connected by subnet 192.168.8.0/24. IP address 192.168.8.1 is assigned to R1's serial 0/1 interface and 192.168.8.2 assigned to R2's serial 0/0 interface.

R2 and R3 are connected by subnet 192.168.5.0/24. IP address 192.168.5.1 is assigned to R2's serial 0/1 interface and 192.168.5.2 assigned to R3's serial 0/0 interface.

R3 and R4 are connected by subnet 192.168.6.0/24. IP address 192.168.6.1 is assigned to R3's serial 0/1 interface and 192.168.6.2 assigned to R4's serial 0/0 interface.

R4 and R1 are connected by subnet 192.168.7.0/24. IP address 192.168.7.1 is assigned to R4's serial 0/1 interface and 192.168.7.2 assigned to R1's serial 0/0 interface.

BIND domain server, BIND sub domain server and DHCP server are directly connected to S1 through Fast Ethernet. Web server is connected to R2's Fast Ethernet and client is connected to R4's Fast Ethernet. Following table explain assigned IP address and host name to all connected server and client in network. Naming system of server and router will helpful to understand operation of project.

Work station	Host name	IP address
BIND domain server (cust.com)	Team4	192.168.1.4
BIND sub domain server (dgrdn.cust.com)	ashish	192.168.1.3
DHCP server	Team3	192.168.1.2
web server	Rack2	Get IP from DHCP server
Client	Rack3	192.168.2.41

Table 5.2 Workstation Information

Remaining 6 IP subnets are used to configure Router's Fast Ethernet interface.

Subnet	Router interface	IP address
192.168.1.0/24	R3's Fa 0/0	192.168.1.1
192.168.2.0/24	R4's Fa 0/0	192.168.2.1
192.168.3.0/24	R1's Fa 0/0	192.168.3.1
192.168.4.0/24	R2's Fa 0/0	192.168.4.1
192.168.9.0/24	R1's Fa 0/1	192.168.9.1
192.168.10.0/24	R4's Fa 0/1	192.168.10.1

Table 5.3 Router interface IP address scheme

5.3.1 Router configuration:

Define following configuration features in all routers. These configurations are common for all routers

- ❖ Assign IP address and subnet masks for Fast Ethernet and Serial interface as explain above

- ❖ Encapsulation-PPP
- ❖ IP helper address
- ❖ Routing protocol-OSPF
- ❖ Clock rate
- ❖ Host name

Following configuration commands are common for all routers:

```
ip helper-address 192.168.1.2
router ospf 1
network 192.168.0.0 0.0.255.255 area 0
encapsulation ppp
clock rate 56000
subnet mask 255.255.255.0
```

R1 configuration

Configure R1 by using `telnet 10.3.31.74` command in command prompt of the team4 machine. Same way access the rest of routers by using consol IP address listed in Table 5.1. Once configuration is done for R1 it will look like.

```
!
version 12.3
!
interface FastEthernet0/0
ip address 192.168.3.1 255.255.255.0
ip helper-address 192.168.1.2
duplex auto
speed auto
!
interface Serial0/0
ip address 192.168.7.2 255.255.255.0
encapsulation ppp
clock rate 56000
no fair-queue
!
interface FastEthernet0/1
ip address 192.168.9.1 255.255.255.0
ip helper-address 192.168.1.2
duplex auto
speed auto
!
interface Serial0/1
ip address 192.168.8.1 255.255.255.0
encapsulation ppp
clock rate 56000
!
```

```
router ospf 1
log-adjacency-changes
network 192.168.0.0 0.0.255.255 area 0
!
end
```

R2 configuration

```
!
version 12.3
!
interface FastEthernet0/0
ip address 192.168.4.1 255.255.255.0
ip helper-address 192.168.1.2
duplex auto
speed auto
!
interface Serial0/0
ip address 192.168.8.2 255.255.255.0
encapsulation ppp
!
interface Serial0/1
ip address 192.168.5.1 255.255.255.0
encapsulation ppp
clock rate 56000
!
router ospf 1
log-adjacency-changes
network 192.168.0.0 0.0.255.255 area 0
!
end
```

R3 configuration

```
!
version 12.3
!
interface FastEthernet0/0
ip address 192.168.1.1 255.255.255.0
ip helper-address 192.168.1.2
duplex auto
speed auto
!
interface Serial0/0
ip address 192.168.5.2 255.255.255.0
encapsulation ppp
```

```

no fair-queue
!
interface Serial0/1
ip address 192.168.6.1 255.255.255.0
encapsulation ppp
clock rate 56000
!
router ospf 1
log-adjacency-changes
network 192.168.0.0 0.0.255.255 area 0
!
end

```

R4 configuration

```

!
version 12.3
!
interface FastEthernet0/0
ip address 192.168.2.1 255.255.255.0
ip helper-address 192.168.1.2
duplex auto
speed auto
!
interface Serial0/0
ip address 192.168.6.2 255.255.255.0
encapsulation ppp
no fair-queue
!
interface FastEthernet0/1
ip address 192.168.10.1 255.255.255.0
ip helper-address 192.168.1.2
duplex auto
speed auto
!
interface Serial0/1
ip address 192.168.7.1 255.255.255.0
encapsulation ppp
!
router ospf 1
log-adjacency-changes
network 192.168.0.0 0.0.255.255 area 0
!
end

```

5.3.2 BIND configuration

Two separate systems are required to configure the BIND server. One server acts as domain (parent) server and the other is a sub domain (child) server. Why are there two different BIND servers? I will explain later in this chapter.

As shown in the network diagram IP address 192.168.1.4 is used for the Ubuntu system in which is responsible for domain **cust.com** and 192.168.1.3 is assigned for domain **dgrdn.cust.com** zone which is sub domain of cust.com domain. Both BIND configured server are directly connected to switch S1 through the Ethernet cable.

A detail installation procedure of BINDv9 is explained in chapter 3. The BIND package is installed in both systems and the installation directory is located at /etc/bind. Also create folder by name *zones* in both systems to store forward and reverse zone data file by using sudo mkdir /etc/bind/zones command.

As explained in the table above, system with IP address 192.168.1.4 is for team4 host and 192.168.1.3 is for ashish host

Main configuration file of both BIND domain server and sub domain server are located at:

- ❖ /etc/bind/named.conf
- ❖ /etc/bind/named.conf.local
- ❖ /etc/bind/named.conf.option
- ❖ /etc/rndckey

The primary configuration file for both BIND servers is /etc/bind/named.conf. BIND use named.conf file to run the server. If it is not present then it will give you an error message.

5.3.2.1 Domain configuration

Forward map configuration

Do following configuration in host **team4** and assigned IP address is 192.168.1.4

Create sudo gedit /etc/bind/zones/cust.com.db file and edit following configuration

```
$TTL 86400
@ IN SOA ns.cust.com. pakodi.cust.com. (
        41 ; Serial
        604800 ; Refresh
        86400 ; Retry
        2419200 ; Expire
        86400) ; Negative Cache TTL
;
```

cust.com.	IN	NS	ns.cust.com.
cust.com.	IN	A	192.168.1.4
ns	IN	A	192.168.1.4
www	IN	A	192.168.1.4
team4	IN	A	192.168.1.4

```
dgrdn.cust.com. IN NS ns.dgrdn.cust.com.
ns.dgrdn.cust.com. A 192.168.1.3 ; glue record
```

A detailed explanation of the configuration is given in chapter 3. The last line of configuration is very important. It is pointing to the sub domain server (192.168.1.3-dgrdn.cust.com) 192.168.1.3 is address of sub domain server. When any client queries for host residing in sub domain server then the domain server uses the last line of configuration to point the sub domain server to resolve the query.

Reverse map configuration

Same way create sudo gedit /etc/bind/zones/ in-addr.arpa file and edit following reverse map zone.

```
$TTL 604800
@ IN SOA ns.cust.com. pakodi.cust.com.(
        41           ; Serial
        604800       ; Refresh
        86400        ; Retry
        2419200      ; Expire
        604800       ; Negative Cache TTL
;
        IN NS ns.cust.com.
```

NOTE: Don't forget to increment the zone file's serial number in both forward and reverse map before saving the file and reloading the zone

❖ Named.conf configuration file for domain

Open file /etc/bind/named.conf and add following configuration lines:

```
include "/etc/bind/named.conf.options";
zone "." {
    type hint;
    file "/etc/bind/db.root";
};

zone "localhost" {
    type master;
    file "/etc/bind/db.local";
};

zone "127.in-addr.arpa" {
    type master;
    file "/etc/bind/db.127";
};

zone "0.in-addr.arpa" {
    type master;
```

```

        file "/etc/bind/db.0";
};

zone "255.in-addr.arpa" {
    type master;
    file "/etc/bind/db.255";
};

include "/etc/bind/named.conf.local";

```

We don't need the BIND domain server to accept dynamic updates from the DHCP server so there is no control statement in named.conf configuration file.

❖ named.conf.local configuration file for Domain

Edit following configuration in /etc/bind/named.conf.local

```

key "rndc-key" {
    algorithm hmac-md5;
    secret "EGfbL8Lo7JWZUQbcWyBmlQ==";
};

zone "cust.com"{
type master;
file "/etc/bind/zones/cust.com.db";
};

zone "in-addr.arpa"{
type master;
file "/etc/bind/zones/in-addr.arpa";
};

```

5.3.2.1.1 Network and DNS setting for Domain server

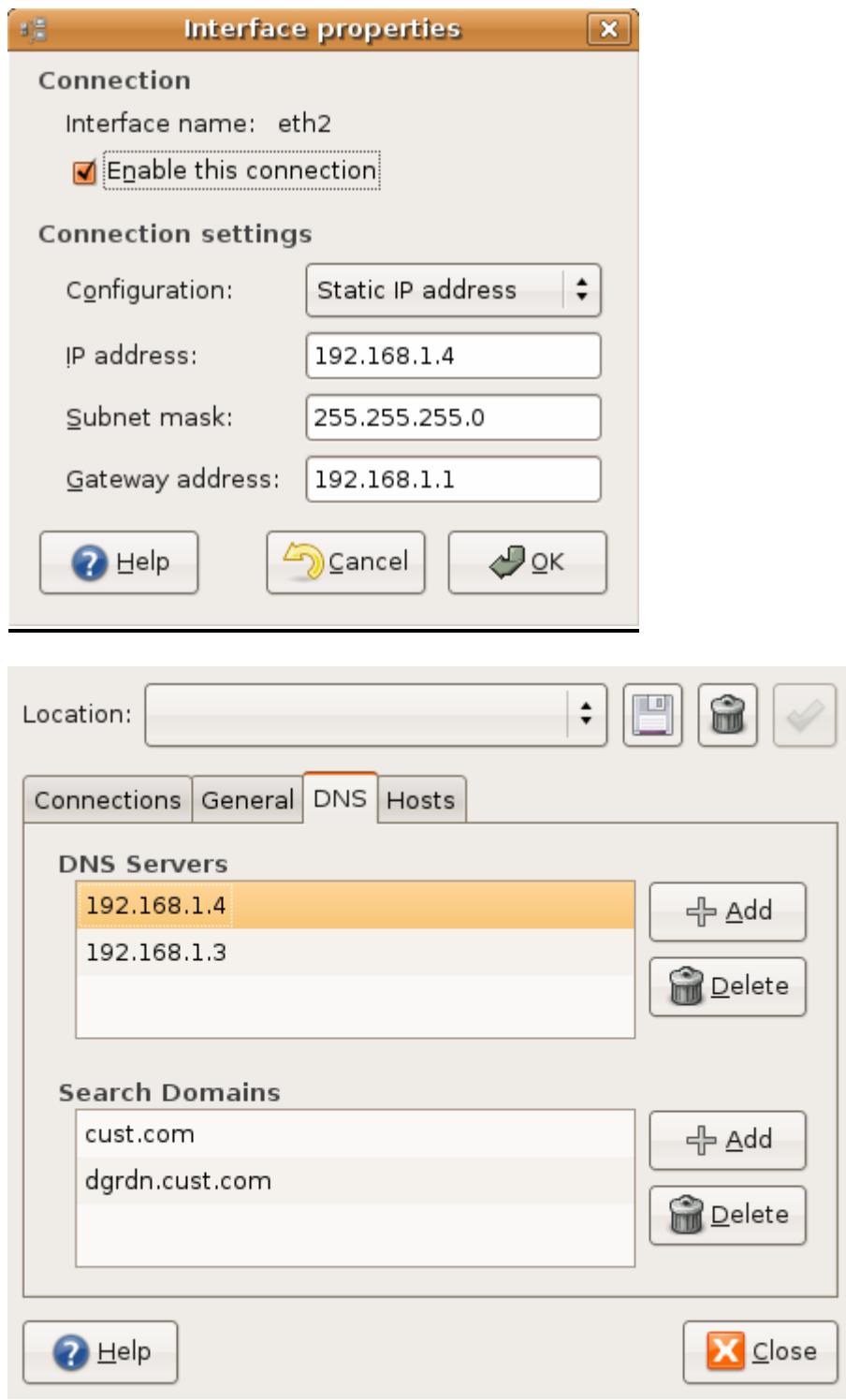


Figure 5.2 IP address and DNS setting in Domain server

5.3.2.2 Sub Domain configuration

Do following configuration in host **ashish** and assigned IP address 192.168.1.3
Open sudo gedit /etc/bind/zones/dgrdn.cust.com.db file and edit following configuration

Initial Forward zone configuration file for dgrdn.cust.com sub domain

```
$TTL 86400
@ IN SOA ns.dgrdn.cust.com. pakodi.dgrdn.cust.com. (
        42 ; Serial
        604800 ; Refresh
        86400 ; Retry
        2419200 ; Expire
        86400 ) ; Negative Cache TTL
;

dgrdn.cust.com. IN NS ns.dgrdn.cust.com.
ns IN A 192.168.1.3
ashish IN A 192.168.1.3
www IN A 192.168.1.3
```

Initial Reverse zone configuration file for dgrdn.cust.com sub domain

Same way Open sudo gedit /etc/bind/zones/in-addr.arpa file and edit following configuration lines.

```
$TTL 604800
@ IN SOA ns.dgrdn.cust.com. pakodi.dgrdn.cust.com.(
        42 ; Serial
        604800 ; Refresh
        86400 ; Retry
        2419200 ; Expire
        604800 ) ; Negative Cache TTL
;
IN NS ns.dgrdn.cust.com.
```

These are initial configuration file for forward and reverse map. Keep in mind we are using BIND sub domain server to accept dynamic updates from the DHCP server. Once BIND sub domain server is updated with dynamic updates we will not see above configuration in configuration files. Both forward and reverse map file will change because of dynamic updates.

No need to update serial number because DHCP server sends update to BIND server with increased serial number

❖ Named.conf configuration file:

Edit following configuration file in /etc/bind/named.conf

```
include "/etc/bind/named.conf.options";
zone "." {
```

```

        type hint;
        file "/etc/bind/db.root";
};

zone "localhost" {
    type master;
    file "/etc/bind/db.local";
};

zone "127.in-addr.arpa" {
    type master;
    file "/etc/bind/db.127";
};

zone "0.in-addr.arpa" {
    type master;
    file "/etc/bind/db.0";
};

zone "255.in-addr.arpa" {
    type master;
    file "/etc/bind/db.255";
};

controls {
inet 127.0.0.1 allow {127.0.0.1; 192.168.1.3; 192.168.1.2;} keys {"rndc-key";}
};

include "/etc/bind/named.conf.local";

```

❖ named.conf.local configuration file

Edit following configuration file in /etc/bind/named.conf.local

```

logging {
    channel query.log {
        file "/var/log/query.log";
        // Set the severity to dynamic to see all the debug messages.
        severity dynamic;
    };
    channel query.log {
        file "/var/log/query.log";
        // Set the severity to dynamic to see all the debug messages.
        severity debug 3;
    };
    category queries { query.log; };
    channel dupdate.log {
        file "/var/log/dupdate.log";
        // Set the severity to dynamic to see all the debug messages.
        severity dynamic;
    };
    channel dupdate.log {

```

```

file "/var/log/dupdate.log";
// Set the severity to dynamic to see all the debug messages.
severity debug 3;
};

category update { dupdate.log; };
};

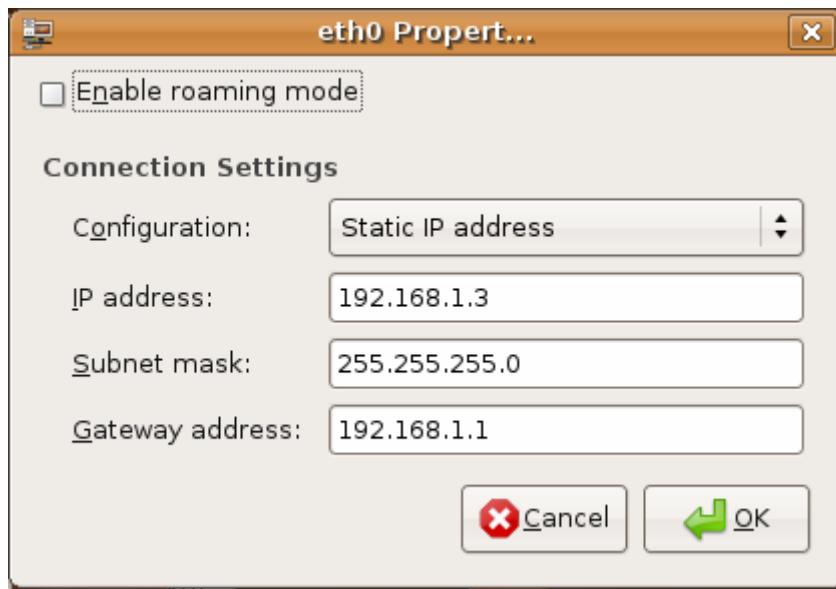
key "rndc-key" {
    algorithm hmac-md5;
    secret "wwmyGnn59ZRXXdCWO5ILg==";
};

zone "dgrdn.cust.com"{
type master;
file "/etc/bind/zones/dgrdn.cust.com.db";
allow-update { key "rndc-key"; };
};

zone "in-addr.arpa"{
type master;
notify no;
file "/etc/bind/zones/in-addr.arpa";
allow-update { key "rndc-key"; };
};

```

5.3.2.1.2 Network and DNS setting for BIND Sub Domain server



DNS setting:

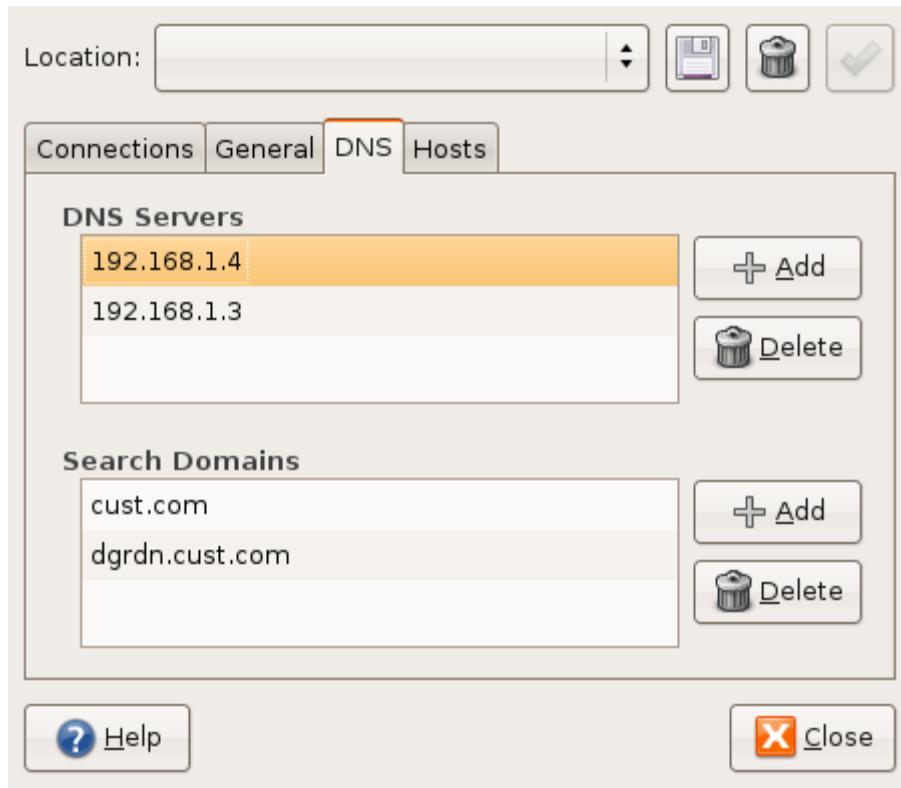


Figure 5.3 IP address and DNS setting in sub Domain server

Once all the configuration is done restart the BIND domain and subdomain server by using following command.

```
sudo /etc/init.d/bind9 restart
```

It is recommended that the child subdomain's nameserver is configured and functions properly before reloading the parent domain's zone file on the parent nameservers.

Use sudo /etc/init.d/bind9 restart command to start the BIND subdomain server first then after using the same command start the BIND domain server.

5.3.3 DHCP server configuration

As shown in network diagram, the DHCP server is connected with switch S1 in the subnet 192.168.1.0/24.

The installation procedure is explained in chapter 4.

The installation file is stored in /etc/dhcp3/dhcpd.conf

Edit following configuration in above file

```

authoritative;
ddns-update-style interim;
ddns-updates on;
ddns-domainname "dgrdn.cust.com";
ddns-rev-domainname "in-addr.arpa";
option domain-name "cust.com";
option domain-name-servers 192.168.1.4;
allow unknown-clients;
key "rndc-key" {
    algorithm hmac-md5;
    secret "wwmyGnn59ZRXXdCWO5ILg==";
}

default-lease-time 300;
max-lease-time 300;
log-facility local7;

zone dgrdn.cust.com{
primary 192.168.1.3;
key "rndc-key";
}

zone in-addr.arpa {
primary 192.168.1.3;
key "rndc-key";
}

subnet 192.168.1.0 netmask 255.255.255.0 {
range 192.168.1.36 192.168.1.41;
option subnet-mask 255.255.255.0;
option broadcast-address 192.168.1.255;
option routers 192.168.1.1;
default-lease-time 300;
max-lease-time 300;
}

subnet 192.168.2.0 netmask 255.255.255.0 {
range 192.168.2.36 192.168.2.41;
option routers 192.168.2.1;
option subnet-mask 255.255.255.0;
option broadcast-address 192.168.2.255;
default-lease-time 300;
max-lease-time 300;
}

```

```
subnet 192.168.3.0 netmask 255.255.255.0 {  
range 192.168.3.36 192.168.3.41;  
option routers 192.168.3.1;  
option subnet-mask 255.255.255.0;  
option broadcast-address 192.168.3.255;  
default-lease-time 300;  
max-lease-time 300;  
}
```

```
subnet 192.168.4.0 netmask 255.255.255.0 {  
range 192.168.4.36 192.168.4.41;  
option routers 192.168.4.1;  
option subnet-mask 255.255.255.0;  
option broadcast-address 192.168.4.255;  
default-lease-time 300;  
max-lease-time 300;  
}
```

```
subnet 192.168.9.0 netmask 255.255.255.0 {  
range 192.168.9.36 192.168.9.41;  
option routers 192.168.9.1;  
option subnet-mask 255.255.255.0;  
option broadcast-address 192.168.9.255;  
default-lease-time 300;  
max-lease-time 300;  
}
```

```
subnet 192.168.10.0 netmask 255.255.255.0 {  
range 192.168.10.36 192.168.10.41;  
option routers 192.168.10.1;  
option subnet-mask 255.255.255.0;  
option broadcast-address 192.168.10.255;  
default-lease-time 300;  
max-lease-time 300;  
}
```

Once it is configured start the DHCP server by using sudo /etc/init.d/dhcp3-server start command.

5.3.3.1 Network and DNS setting in DHCP server

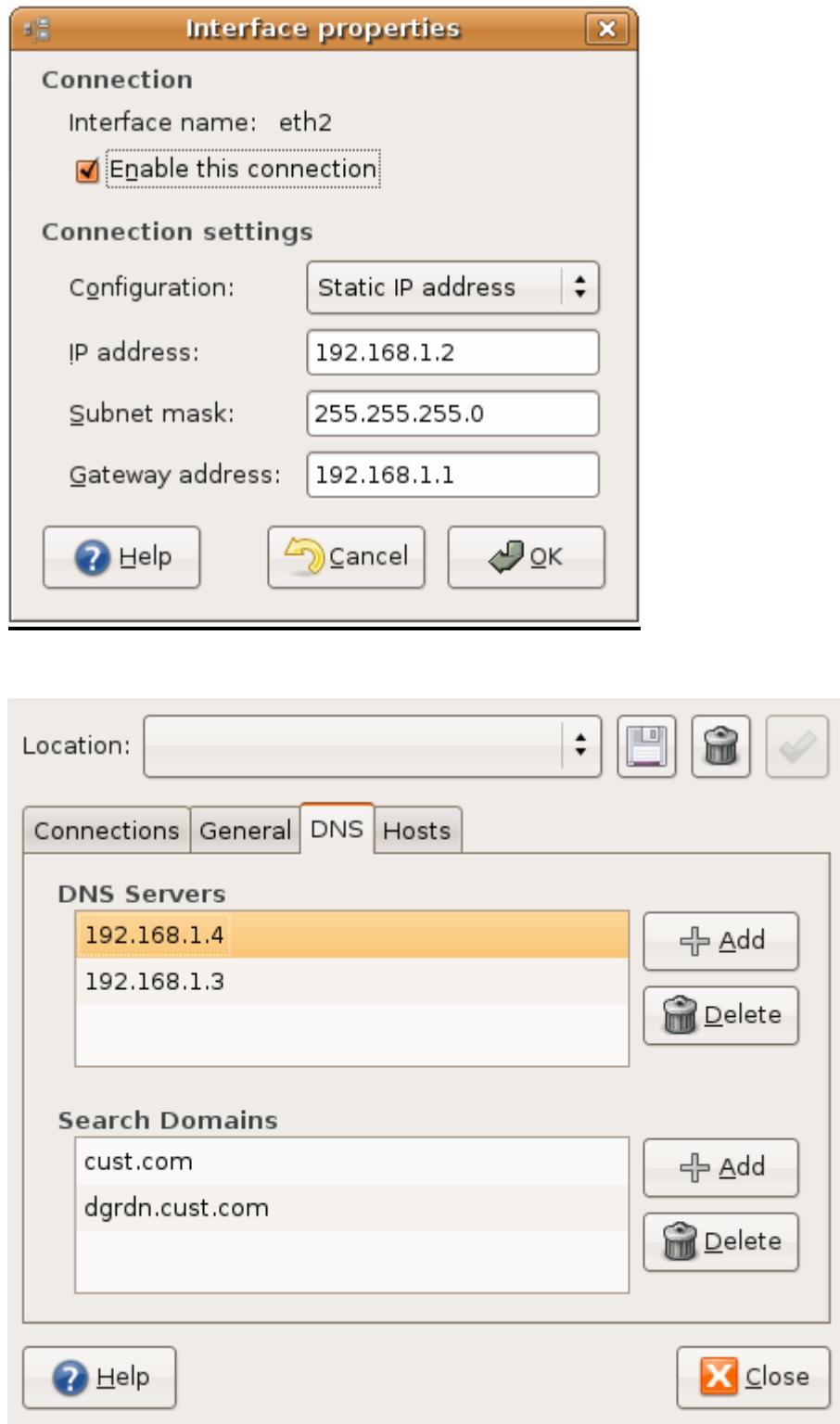


Figure 5.4 IP address and DNS setting in DHCP server

Web server and client network setting configuration

The web server and client are using the DHCP server to get IP address dynamically. No static configuration is required. Just enable the DHCP in both machines. Open the TCP/IP properties screen as shown in below and select “obtain an IP address automatically” option.

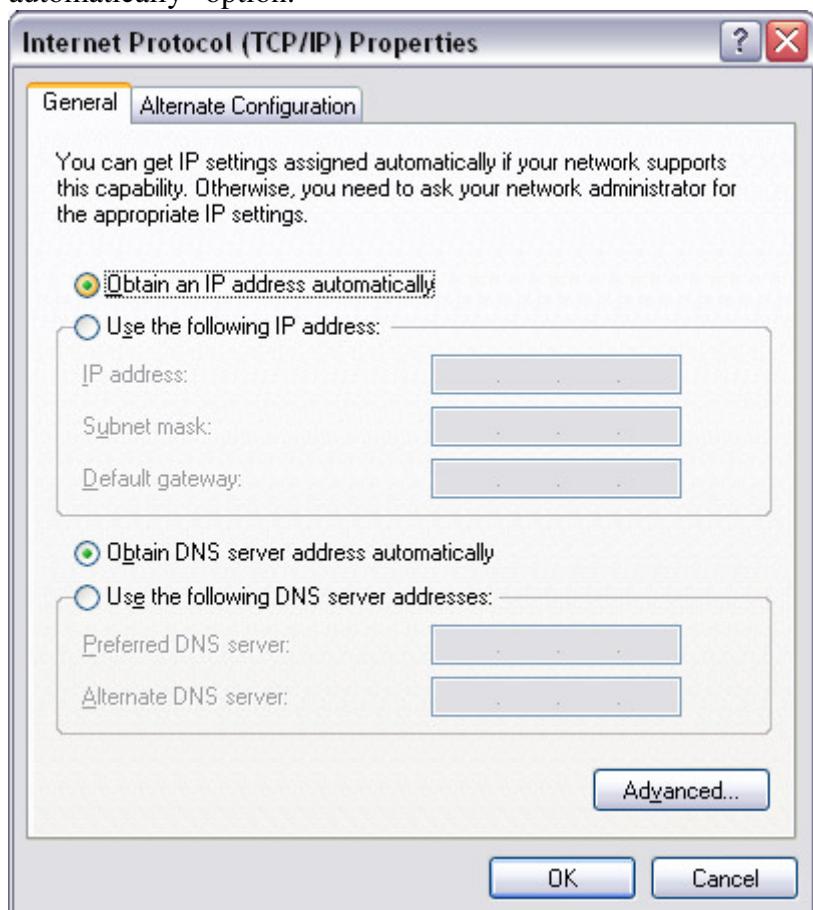


Figure 5.5 DHCP settings in web server and client

5.4 Procedure and operation

As shown in network diagram connect all the equipment in the MINT lab. Use serial cable to connect all routers back to back. Connect R3's Fast Ethernet 0/0 to S1's Fast Ethernet 0/0 using CAT-5 Ethernet cable. The DHCP server, the BIND domain server and the BIND sub domain server are connected to S1's Fast Ethernet. Connect client to 192.168.2.0/24 subnet. Initially web server is connected 192.168.4.0/24 subnet and later we will move web server to 192.168.3.0/24 and 192.168.9.0/24 subnet.

Configure all the routers, the BIND domain server, the BIND sub domain server and the DHCP server as shown above. Use consol IP address to access router as specified in one of the above tables. Assign correct IP address to interfaces as shown in network diagram. Use network and DNS setting shown above for DHCP server, BIND server and BIND domain server. Run all the server ones they are configured.

To check network connectivity run PING in DHCP server for all router's interface. Here we are getting reply from all router interfaces.

The screenshot shows a Linux desktop environment with five terminal windows open, each displaying the output of a ping command to a specific host. The terminals are arranged vertically, each with its own title bar and window controls.

- Terminal 1:** apatell@team3: ~
Shows ping results to 192.168.5.1:
64 bytes from 192.168.5.1: icmp_seq=335 ttl=254 time=28.0 ms
64 bytes from 192.168.5.1: icmp_seq=336 ttl=254 time=28.1 ms
64 bytes from 192.168.5.1: icmp_seq=337 ttl=254 time=28.0 ms
64 bytes from 192.168.5.1: icmp_seq=338 ttl=254 time=28.0 ms
64 bytes from 192.168.5.1: icmp_seq=339 ttl=254 time=28.0 ms
64 bytes from 192.168.5.1: icmp_seq=340 ttl=254 time=34.6 ms
- Terminal 2:** apatell@team3: ~
Shows ping results to 192.168.5.2:
64 bytes from 192.168.5.2: icmp_seq=348 ttl=255 time=1.22 ms
64 bytes from 192.168.5.2: icmp_seq=349 ttl=255 time=1.25 ms
64 bytes from 192.168.5.2: icmp_seq=350 ttl=255 time=1.26 ms
64 bytes from 192.168.5.2: icmp_seq=351 ttl=255 time=1.22 ms
64 bytes from 192.168.5.2: icmp_seq=352 ttl=255 time=1.16 ms
64 bytes from 192.168.5.2: icmp_seq=353 ttl=255 time=1.26 ms
- Terminal 3:** apatell@team3: ~
Shows ping results to 192.168.6.1:
64 bytes from 192.168.6.1: icmp_seq=302 ttl=255 time=1.27 ms
64 bytes from 192.168.6.1: icmp_seq=303 ttl=255 time=1.21 ms
64 bytes from 192.168.6.1: icmp_seq=304 ttl=255 time=1.20 ms
64 bytes from 192.168.6.1: icmp_seq=305 ttl=255 time=1.16 ms
64 bytes from 192.168.6.1: icmp_seq=306 ttl=255 time=1.24 ms
64 bytes from 192.168.6.1: icmp_seq=307 ttl=255 time=1.20 ms
- Terminal 4:** apatell@team3: ~
Shows ping results to 192.168.6.2:
64 bytes from 192.168.6.2: icmp_seq=325 ttl=254 time=27.9 ms
64 bytes from 192.168.6.2: icmp_seq=326 ttl=254 time=28.0 ms
64 bytes from 192.168.6.2: icmp_seq=327 ttl=254 time=28.0 ms
64 bytes from 192.168.6.2: icmp_seq=328 ttl=254 time=28.0 ms
64 bytes from 192.168.6.2: icmp_seq=329 ttl=254 time=27.9 ms
64 bytes from 192.168.6.2: icmp_seq=330 ttl=254 time=27.9 ms
- Terminal 5:** webserver ping
Shows ping results to 192.168.3.41:
64 bytes from 192.168.3.41: icmp_seq=36 ttl=125 time=63.6 ms
64 bytes from 192.168.3.41: icmp_seq=37 ttl=125 time=59.5 ms
64 bytes from 192.168.3.41: icmp_seq=38 ttl=125 time=55.6 ms
64 bytes from 192.168.3.41: icmp_seq=39 ttl=125 time=54.2 ms
64 bytes from 192.168.3.41: icmp_seq=40 ttl=125 time=54.2 ms
64 bytes from 192.168.3.41: icmp_seq=41 ttl=125 time=55.6 ms
64 bytes from 192.168.3.41: icmp_seq=42 ttl=125 time=55.7 ms
64 bytes from 192.168.3.41: icmp_seq=43 ttl=125 time=55.0 ms

The desktop interface includes a menu bar at the top with options like Applications, Places, System, and a system tray icon. The taskbar at the bottom shows other open applications: Terminal (16) and gedit [dhcpd.conf (/etc/dhcp3)].

```

apatell@team3: ~
File Edit View Terminal Tabs Help
64 bytes from 192.168.8.1: icmp_seq=271 ttl=253 time=59.1 ms
64 bytes from 192.168.8.1: icmp_seq=272 ttl=253 time=57.9 ms
64 bytes from 192.168.8.1: icmp_seq=273 ttl=253 time=57.2 ms
64 bytes from 192.168.8.1: icmp_seq=274 ttl=253 time=57.1 ms
64 bytes from 192.168.8.1: icmp_seq=275 ttl=253 time=59.6 ms
64 bytes from 192.168.8.1: icmp_seq=276 ttl=253 time=59.8 ms

apatell@team3: ~
File Edit View Terminal Tabs Help
64 bytes from 192.168.8.2: icmp_seq=260 ttl=254 time=28.0 ms
64 bytes from 192.168.8.2: icmp_seq=261 ttl=254 time=28.1 ms
64 bytes from 192.168.8.2: icmp_seq=262 ttl=254 time=28.0 ms
64 bytes from 192.168.8.2: icmp_seq=263 ttl=254 time=28.2 ms
64 bytes from 192.168.8.2: icmp_seq=264 ttl=254 time=28.1 ms
64 bytes from 192.168.8.2: icmp_seq=265 ttl=254 time=28.3 ms

apatell@team3: ~
File Edit View Terminal Tabs Help
64 bytes from 192.168.7.2: icmp_seq=281 ttl=253 time=55.1 ms
64 bytes from 192.168.7.2: icmp_seq=282 ttl=253 time=55.0 ms
64 bytes from 192.168.7.2: icmp_seq=283 ttl=253 time=55.0 ms
64 bytes from 192.168.7.2: icmp_seq=284 ttl=253 time=55.1 ms
64 bytes from 192.168.7.2: icmp_seq=285 ttl=253 time=55.0 ms

apatell@team3: ~
File Edit View Terminal Tabs Help
64 bytes from 192.168.7.1: icmp_seq=289 ttl=254 time=28.0 ms
64 bytes from 192.168.7.1: icmp_seq=290 ttl=254 time=28.2 ms
64 bytes from 192.168.7.1: icmp_seq=291 ttl=254 time=28.3 ms
64 bytes from 192.168.7.1: icmp_seq=292 ttl=254 time=28.0 ms
64 bytes from 192.168.7.1: icmp_seq=293 ttl=254 time=28.0 ms
64 bytes from 192.168.7.1: icmp_seq=294 ttl=254 time=28.0 ms
64 bytes from 192.168.7.1: icmp_seq=295 ttl=254 time=28.1 ms

webserver ping
File Edit View Terminal Tabs Help
64 bytes from 192.168.2.41: icmp_seq=60 ttl=126 time=27.4 ms
64 bytes from 192.168.2.41: icmp_seq=61 ttl=126 time=27.4 ms
64 bytes from 192.168.2.41: icmp_seq=62 ttl=126 time=27.5 ms
64 bytes from 192.168.2.41: icmp_seq=63 ttl=126 time=27.4 ms
64 bytes from 192.168.2.41: icmp_seq=64 ttl=126 time=27.5 ms
64 bytes from 192.168.2.41: icmp_seq=65 ttl=126 time=27.5 ms
64 bytes from 192.168.2.41: icmp_seq=66 ttl=126 time=27.5 ms
64 bytes from 192.168.2.41: icmp_seq=67 ttl=126 time=27.4 ms
64 bytes from 192.168.2.41: icmp_seq=68 ttl=126 time=27.4 ms

```

Figure 5.6 PING reply from all router interfaces

By observing above ping reply we can say our network is running fine. It is ready to do experiment.

Initially the web server and the client don't have any IP address. They are sending broadcast request to all DHCP servers in the network. Once DHCP server is running well, it assigns the IP address to all clients by exchanging messages with client. The detail operation of assigning IP address is discussed in chapter 4. The Web server will get IP address, DNS server, default gateway, subnet mask, DHCP server IP address and domain name from DHCP server. Very first web server is connected to subnet 192.168.9.0/24.

Defined range in DHCP configuration file for subnet 192.168.9.0/24 is from 192.168.9.36 to 192.168.9.41. The DHCP server starts to assign IP address from high range end and decrease to lower order of range. As shown in below Figure 5.7 the web server gets the IP address 192.168.9.41. Web server will get IP address for 300 second and will renew lease after 150 second. The DHCP server writes assigned leased in separate leased file located at `/var/lib/dhcp3/dhcpd.leases`. Every time leased file is updated when client's lease is renew or assigned.

```
C:\ Command Prompt
Tunnel adapter Automatic Tunneling Pseudo-Interface:
  Connection-specific DNS Suffix . : cust.com
  IP Address . . . . . : fe80::5efe:192.168.4.41%2
  Default Gateway . . . . . :
C:\Documents and Settings\teamrack>ipconfig
Windows IP Configuration

Ethernet adapter Local Area Connection 6:
  Connection-specific DNS Suffix . : cust.com
  IP Address . . . . . : 192.168.9.41
  Subnet Mask . . . . . : 255.255.255.0
  IP Address . . . . . : fe80::202:b3ff:fed4:c7d2%4
  Default Gateway . . . . . : 192.168.9.1

Ethernet adapter Local Area Connection 3:
  Media State . . . . . : Media disconnected

Ethernet adapter Local Area Connection:
  Media State . . . . . : Media disconnected

Ethernet adapter Local Area Connection 5:
  Media State . . . . . : Media disconnected

Ethernet adapter Local Area Connection 4:
  Media State . . . . . : Media disconnected

Tunnel adapter Teredo Tunneling Pseudo-Interface:
  Connection-specific DNS Suffix . :
  IP Address . . . . . : fe80::ffff:ffff:ffffd%8
  Default Gateway . . . . . :

Tunnel adapter Automatic Tunneling Pseudo-Interface:
  Connection-specific DNS Suffix . : cust.com
  IP Address . . . . . : fe80::5efe:192.168.9.41%2
  Default Gateway . . . . . :

C:\Documents and Settings\teamrack>
```

Figure 5.7 IPconfig output of web server (First time connected)

Now the DHCP server sends dynamic update to the BIND sub domain server for all clients who just got IP address and other TCP/IP configuration parameter.

In our experiment we are using two BIND servers one for *cust.com* domain and another for *dgrdn.cust.com* sub domain. When any client send query, it will ask to *cust.com* domain server first because all client is getting *cust.com* domain as default DNS server. *dgrdn.cust.com* domain server is used only to accept dynamic update from DHCP server. If BIND domain server doesn't have any answer for query then after it will ask to sub domain BIND server. Reason behind using two separate BIND servers is to minimize load on single server, easy administration and increase security for dynamic update.

First time when the DHCP server sends dynamic update to BIND sub domain server, it will create journal file in directory /etc/bind/zones/ with extension *.jnl*. BIND sub domain server will create two separate journal file for forward and reverse map zone file. Journal file keep the update from the DHCP server for 15 minutes. Once this time elapsed updates from journal file are moved to reverse and forward zone data file.

Here are initially updated forward and reverse map zone data file of sub domain BIND server when web server is connected to the network first time. These zone files are updated with the IP address assign to webserver and it's FQDN. The previous original configuration is updated in both zone data files.

Forward map file for sub domain

```
$ORIGIN .
$TTL 86400 ; 1 day
dgrdn.cust.com.      IN SOA      ns.dgrdn.cust.com. pakodi.dgrdn.cust.com. (
                           98      ; serial
                           604800  ; refresh (1 week)
                           86400   ; retry (1 day)
                           2419200 ; expire (4 weeks)
                           86400   ; minimum (1 day)
                         )
NS      ns.dgrdn.cust.com.

$ORIGIN dgrdn.cust.com.
ashish      A      192.168.1.3
ns          A      192.168.1.3
$TTL 150    ; 2 minutes 30 seconds
rack2       A      192.168.9.41
             TXT   "31ec0ea8c148f00cdd8c136ad60ef3d2bc"
rack3       A      192.168.2.41
             TXT   "315a03dee4ef6563b7d39d9d03fd80edc9"
$TTL 86400  ; 1 day
www        A      192.168.1.3
```

Once the webserver is updated in zone data file, it uses sub domain's name *dgrdn.cust.com* as suffix. In the above updated file rack2 and rack3 will get *dgrdn.cust.com* as suffix.

Output of query for rack2

```
> rack2.dgrdn
Server: Unknown
Address: 192.168.1.4

Non-authoritative answer:
Name:   rack2.dgrdn.cust.com
Address: 192.168.9.41

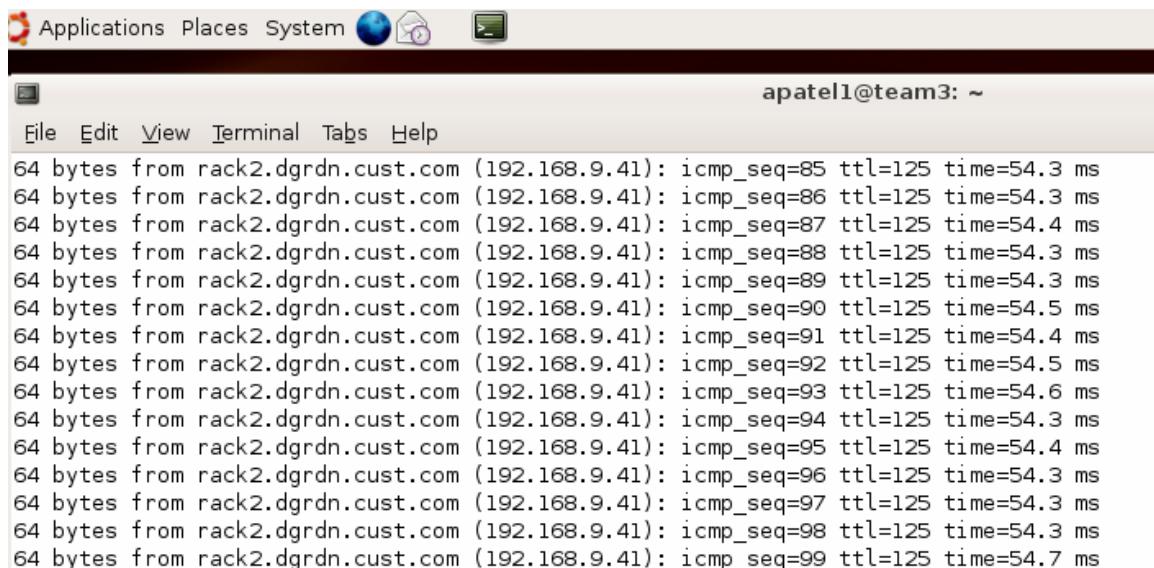
>
```

Figure 5.8 NSLOOKUP output of web server (first time connected)

Reverse map file for sub domain

```
$ORIGIN .
$TTL 604800 ; 1 week
in-addr.arpa      IN SOA      ns.dgrdn.cust.com. pakodi.dgrdn.cust.com. (
                      80      ; serial
                      604800  ; refresh (1 week)
                      86400   ; retry (1 day)
                      2419200 ; expire (4 weeks)
                      604800  ; minimum (1 week)
)
                      NS      ns.dgrdn.cust.com.
$ORIGIN 168.192.in-addr.arpa.
$TTL 150    ; 2 minutes 30 seconds
41.2        PTR    rack3.dgrdn.cust.com.
41.9        PTR    rack2.dgrdn.cust.com.
```

PING reply from web server:



The screenshot shows a terminal window with a dark grey header bar containing icons for Applications, Places, System, and a lock. The main window title is "apatell@team3: ~". The menu bar includes File, Edit, View, Terminal, Tabs, and Help. The terminal window displays a series of ICMP echo replies from the IP address 192.168.9.41. Each reply shows 64 bytes sent, an ICMP sequence number (seq), a TTL of 125, and a time value ranging from 54.3 ms to 54.7 ms.

```
64 bytes from rack2.dgrdn.cust.com (192.168.9.41): icmp_seq=85 ttl=125 time=54.3 ms
64 bytes from rack2.dgrdn.cust.com (192.168.9.41): icmp_seq=86 ttl=125 time=54.3 ms
64 bytes from rack2.dgrdn.cust.com (192.168.9.41): icmp_seq=87 ttl=125 time=54.4 ms
64 bytes from rack2.dgrdn.cust.com (192.168.9.41): icmp_seq=88 ttl=125 time=54.3 ms
64 bytes from rack2.dgrdn.cust.com (192.168.9.41): icmp_seq=89 ttl=125 time=54.3 ms
64 bytes from rack2.dgrdn.cust.com (192.168.9.41): icmp_seq=90 ttl=125 time=54.5 ms
64 bytes from rack2.dgrdn.cust.com (192.168.9.41): icmp_seq=91 ttl=125 time=54.4 ms
64 bytes from rack2.dgrdn.cust.com (192.168.9.41): icmp_seq=92 ttl=125 time=54.5 ms
64 bytes from rack2.dgrdn.cust.com (192.168.9.41): icmp_seq=93 ttl=125 time=54.6 ms
64 bytes from rack2.dgrdn.cust.com (192.168.9.41): icmp_seq=94 ttl=125 time=54.3 ms
64 bytes from rack2.dgrdn.cust.com (192.168.9.41): icmp_seq=95 ttl=125 time=54.4 ms
64 bytes from rack2.dgrdn.cust.com (192.168.9.41): icmp_seq=96 ttl=125 time=54.3 ms
64 bytes from rack2.dgrdn.cust.com (192.168.9.41): icmp_seq=97 ttl=125 time=54.3 ms
64 bytes from rack2.dgrdn.cust.com (192.168.9.41): icmp_seq=98 ttl=125 time=54.3 ms
64 bytes from rack2.dgrdn.cust.com (192.168.9.41): icmp_seq=99 ttl=125 time=54.7 ms
```

Figure 5.9 PING reply from web server

Now let's move the web server from 192.168.9.0/24 subnet to 192.168.4.0/24 subnet and see 1) How DHCP server assigns new IP address? 2) How DHCP server sends update to BIND server 3) How BIND server remove the RRs from zone data file?

Move web server physically by removing CAT 5 Ethernet cable from 192.168.9.0 subnet and attach to 192.168.4.0 subnet.

Once the web server is moved from 192.168.9.0 to 192.168.4.0, web server send broadcast request to the DHCP server with the existing lease. The DHCP server notices that webserver is trying to renew lease by using wrong subnet network. DHCP server remove wrong IP address mapped with webserver's MAC address from lease file. It also sends request to the BIND sub domain server to remove resource record from zone data file. The BIND server remove A and TXT record from forward and reverse map file. Ones record removes from zone data file then BIND server is not a capable to reply query of web server (rack2.dgrdn.cust.com)

Below screen shot of dupdate.log will show you how BIND server is removing RR from dgrdn.cust.com and in-addr.arpa zone data file. It is removing A and TXT record from dgrdn.cust.com forward zone file and PTR record from in-addr.arpa reverse zone file.

The screenshot shows a window titled '/var/log/dupdate.log (monitored) - System Log Viewer'. The window contains a list of log entries from the file '/var/log/dupdate.log'. The entries are as follows:

```
client 192.168.1.2#33116: updating zone 'dgrdn.cust.com/IN': adding an RR at 'rack2.dgrdn.cust.com' A
client 192.168.1.2#33116: updating zone 'dgrdn.cust.com/IN': adding an RR at 'rack2.dgrdn.cust.com' TXT
client 192.168.1.2#33116: updating zone 'in-addr.arpa/IN': deleting rrset at '41.9.168.192.in-addr.arpa' PTR
client 192.168.1.2#33116: updating zone 'in-addr.arpa/IN': adding an RR at '41.9.168.192.in-addr.arpa' PTR
client 192.168.1.2#33116: updating zone 'dgrdn.cust.com/IN': deleting an RR
client 192.168.1.2#33116: updating zone 'dgrdn.cust.com/IN': deleting an RR
client 192.168.1.2#33116: updating zone 'in-addr.arpa/IN': deleting rrset at '41.9.168.192.in-addr.arpa' PTR
client 192.168.1.2#33116: updating zone 'dgrdn.cust.com/IN': adding an RR at 'rack2.dgrdn.cust.com' A
client 192.168.1.2#33116: updating zone 'dgrdn.cust.com/IN': adding an RR at 'rack2.dgrdn.cust.com' TXT
client 192.168.1.2#33116: updating zone 'in-addr.arpa/IN': deleting rrset at '41.4.168.192.in-addr.arpa' PTR
client 192.168.1.2#33116: updating zone 'in-addr.arpa/IN': adding an RR at '41.4.168.192.in-addr.arpa' PTR
client 192.168.1.2#33116: updating zone 'dgrdn.cust.com/IN': deleting an RR
client 192.168.1.2#33116: updating zone 'dgrdn.cust.com/IN': deleting an RR
client 192.168.1.2#33116: updating zone 'in-addr.arpa/IN': deleting rrset at '41.4.168.192.in-addr.arpa' PTR
client 192.168.1.2#33116: updating zone 'dgrdn.cust.com/IN': adding an RR at 'rack2.dgrdn.cust.com' A
client 192.168.1.2#33116: updating zone 'dgrdn.cust.com/IN': adding an RR at 'rack2.dgrdn.cust.com' TXT
client 192.168.1.2#33116: updating zone 'in-addr.arpa/IN': deleting rrset at '41.9.168.192.in-addr.arpa' PTR
client 192.168.1.2#33121: updating zone 'dgrdn.cust.com/IN': deleting an RR
client 192.168.1.2#33121: updating zone 'dgrdn.cust.com/IN': deleting an RR
client 192.168.1.2#33121: updating zone 'in-addr.arpa/IN': deleting rrset at '41.9.168.192.in-addr.arpa' PTR
client 192.168.1.2#33121: updating zone 'dgrdn.cust.com/IN': adding an RR at 'rack2.dgrdn.cust.com' A
client 192.168.1.2#33121: updating zone 'dgrdn.cust.com/IN': adding an RR at 'rack2.dgrdn.cust.com' TXT
client 192.168.1.2#33121: updating zone 'in-addr.arpa/IN': deleting rrset at '41.3.168.192.in-addr.arpa' PTR
client 192.168.1.2#33121: updating zone 'in-addr.arpa/IN': adding an RR at '41.3.168.192.in-addr.arpa' PTR
```

At the bottom of the window, it says '1417 lines (142.7 KB) - last update: Mon Jun 15 17:40:02 2009'. The window has a standard Linux desktop interface with icons for Applications, Places, System, and a user icon.

Figure 5.10 Dynamic update log output

Once the above process is finished Suddenly DHCP server assigns new IP address to the web server. This time web server is in 192.168.4.0 subnet and assigned range in the DHCP server's configuration file for this subnet is 192.168.4.36 to 192.168.4.41. The DHCP server will assign IP address 192.168.4.41 to the webserver. Again web server will get all details like subnet mask, default gateway, DNS server, domain name etc. from the DHCP server.

Output of the IPCONFIG command in the web server is changed from last output.

```

C:\ Command Prompt
Tunnel adapter Automatic Tunneling Pseudo-Interface:
  Connection-specific DNS Suffix . : cust.com
  IP Address . . . . . : fe80::5efe:192.168.4.41%2
  Default Gateway . . . . . :

C:\Documents and Settings\teamrack>
C:\Documents and Settings\teamrack>ipconfig

Windows IP Configuration

Ethernet adapter Local Area Connection 6:
  Connection-specific DNS Suffix . : cust.com
  IP Address . . . . . : 192.168.4.41
  Subnet Mask . . . . . : 255.255.255.0
  IP Address . . . . . : fe80::202:b3ff:fed4:c7d2%4
  Default Gateway . . . . . : 192.168.4.1

Ethernet adapter Local Area Connection 3:
  Media State . . . . . : Media disconnected

Ethernet adapter Local Area Connection:
  Media State . . . . . : Media disconnected

Ethernet adapter Local Area Connection 5:
  Media State . . . . . : Media disconnected

Ethernet adapter Local Area Connection 4:
  Media State . . . . . : Media disconnected

Tunnel adapter Teredo Tunneling Pseudo-Interface:
  Connection-specific DNS Suffix . :
  IP Address . . . . . : fe80::ffff:ffff:ffffd%8
  Default Gateway . . . . . :

Tunnel adapter Automatic Tunneling Pseudo-Interface:
  Connection-specific DNS Suffix . : cust.com
  IP Address . . . . . : fe80::5efe:192.168.4.41%2
  Default Gateway . . . . . :

C:\Documents and Settings\teamrack>

```

Figure 5.11 IPCONFIG output of web server (After moved)

The DHCP server writes new lease file for IP address 192.168.4.41 with MAC address of the webserver and renew lease after every 150 second.

Once the DHCP server wrote lease file, it sends updates to the BIND subdomain server to add A and TXT record for IP address 192.168.4.41 and host *rack2* in forward map and PTR record for 41.4.168.192.in-addr.arpa in reverse map data file.

Output of updated forward and reverse ZONE data file after web server moved to 192.168.4.0/24 subnet is shown below.

Updated Forward zone data file after webserver is moved

```

$ORIGIN .
$TTL 86400 ; 1 day
dgrdn.cust.com      IN SOA      ns.dgrdn.cust.com. pakodi.dgrdn.cust.com. (
                           98      ; serial
                           604800 ; refresh (1 week)
                           86400  ; retry (1 day)
                           2419200 ; expire (4 weeks)
                           86400  ; minimum (1 day)
                         )
NS      ns.dgrdn.cust.com.

$ORIGIN dgrdn.cust.com.
ashish      A      192.168.1.3
ns          A      192.168.1.3
$TTL 150    ; 2 minutes 30 seconds
rack2       A      192.168.4.41
             TXT   "31ec0ea8c148f00cdd8c136ad60ef3d2bc"
rack3       A      192.168.2.41
             TXT   "315a03dee4ef6563b7d39d9d03fd80edc9"
$TTL 86400  ; 1 day
www        A      192.168.1.3

```

Updated Reverse map zone data file after webserver is moved

```

$ORIGIN .
$TTL 604800 ; 1 week
in-addr.arpa    IN SOA      ns.dgrdn.cust.com. pakodi.dgrdn.cust.com. (
                           80      ; serial
                           604800 ; refresh (1 week)
                           86400  ; retry (1 day)
                           2419200 ; expire (4 weeks)
                           604800 ; minimum (1 week)
                         )
NS      ns.dgrdn.cust.com.

$ORIGIN 168.192.in-addr.arpa.
$TTL 150    ; 2 minutes 30 seconds
41.2       PTR   rack3.dgrdn.cust.com.
41.4       PTR   rack2.dgrdn.cust.com.

```

Below screen shot of syslog show you how the DHCP server remove resource record for IP 192.168.9.41 and add new recourse record for IP address 192.168.4.41.

```

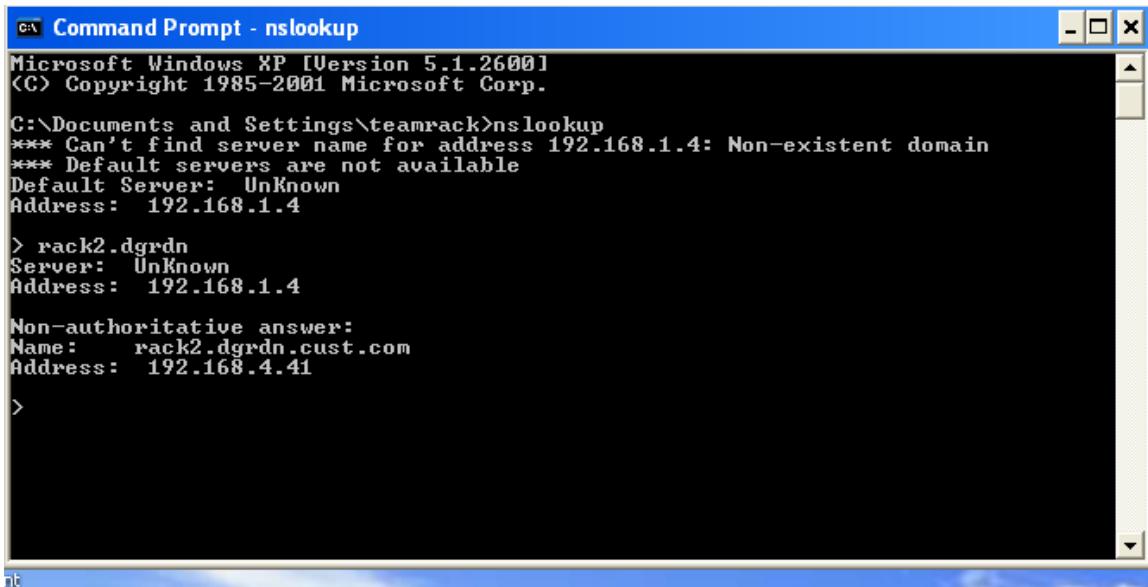
Applications Places System
Mon 15 Jun, 17:30

File Edit View Terminal Tabs Help
Jun 15 17:17:47 localhost dhcpcd: DHCPREQUEST for 192.168.3.41 from 00:02:b3:d4:c7:d2 (rack2) via eth2
Jun 15 17:17:47 localhost dhcpcd: DHCPACK on 192.168.3.41 to 00:02:b3:d4:c7:d2 (rack2) via eth2
Jun 15 17:19:24 localhost dhcpcd: DHCPREQUEST for 192.168.2.41 from 00:02:b3:e7:ec:05 (rack3) via eth2
Jun 15 17:19:24 localhost dhcpcd: DHCPACK on 192.168.2.41 to 00:02:b3:e7:ec:05 (rack3) via eth2
Jun 15 17:21:54 localhost dhcpcd: DHCPREQUEST for 192.168.2.41 from 00:02:b3:e7:ec:05 (rack3) via eth2
Jun 15 17:21:54 localhost dhcpcd: DHCPACK on 192.168.2.41 to 00:02:b3:e7:ec:05 (rack3) via eth2
Jun 15 17:22:14 localhost dhcpcd: DHCPREQUEST for 192.168.3.41 from 00:02:b3:d4:c7:d2 (rack2) via 192.168.9.1: wrong network.
Jun 15 17:22:14 localhost dhcpcd: DHCPNAK on 192.168.3.41 to 00:02:b3:d4:c7:d2 via 192.168.9.1
Jun 15 17:22:15 localhost dhcpcd: DHCPDISCOVER from 00:02:b3:d4:c7:d2 via 192.168.9.1
Jun 15 17:22:15 localhost dhcpcd: DHCPOFFER on 192.168.9.41 to 00:02:b3:d4:c7:d2 (rack2) via 192.168.9.1
Jun 15 17:22:16 localhost dhcpcd: if rack2.dgrdn.cust.com IN TXT "31ec0ea8c148f00cdd8c136ad60ef3d2bc" rrset exists and rack2.dgrdn.cust.com IN A 192.168.3.41
rrset exists delete rack2.dgrdn.cust.com IN A 192.168.3.41: success.
Jun 15 17:22:16 localhost dhcpcd: if rack2.dgrdn.cust.com IN A rrset doesn't exist delete rack2.dgrdn.cust.com IN TXT "31ec0ea8c148f00cdd8c136ad60ef3d2bc": success.
Jun 15 17:22:16 localhost dhcpcd: removed reverse map on 41.3.168.192.in-addr.arpa
Jun 15 17:22:16 localhost dhcpcd: Added new forward map from rack2.dgrdn.cust.com to 192.168.9.41
Jun 15 17:22:16 localhost dhcpcd: added reverse map from 41.9.168.192.in-addr.arpa to rack2.dgrdn.cust.com
Jun 15 17:22:16 localhost dhcpcd: DHCPREQUEST for 192.168.9.41 (192.168.1.2) from 00:02:b3:d4:c7:d2 (rack2) via 192.168.9.1
Jun 15 17:22:16 localhost dhcpcd: DHCPACK on 192.168.9.41 to 00:02:b3:d4:c7:d2 (rack2) via 192.168.9.1
Jun 15 17:24:24 localhost dhcpcd: DHCPREQUEST for 192.168.2.41 from 00:02:b3:e7:ec:05 (rack3) via eth2
Jun 15 17:24:24 localhost dhcpcd: DHCPACK on 192.168.2.41 to 00:02:b3:e7:ec:05 (rack3) via eth2
Jun 15 17:26:43 localhost dhcpcd: DHCPREQUEST for 192.168.9.41 from 00:02:b3:d4:c7:d2 (rack2) via 192.168.4.1: wrong network.
Jun 15 17:26:43 localhost dhcpcd: DHCPNAK on 192.168.9.41 to 00:02:b3:d4:c7:d2 via 192.168.4.1
Jun 15 17:26:45 localhost dhcpcd: DHCPDISCOVER from 00:02:b3:d4:c7:d2 via 192.168.4.1
Jun 15 17:26:45 localhost dhcpcd: DHCPOFFER on 192.168.4.41 to 00:02:b3:d4:c7:d2 (rack2) via 192.168.4.1
Jun 15 17:26:46 localhost dhcpcd: if rack2.dgrdn.cust.com IN TXT "31ec0ea8c148f00cdd8c136ad60ef3d2bc" rrset exists and rack2.dgrdn.cust.com IN A 192.168.9.41
rrset exists delete rack2.dgrdn.cust.com IN A 192.168.9.41: success.
Jun 15 17:26:46 localhost dhcpcd: if rack2.dgrdn.cust.com IN A rrset doesn't exist delete rack2.dgrdn.cust.com IN TXT "31ec0ea8c148f00cdd8c136ad60ef3d2bc": success.
Jun 15 17:26:46 localhost dhcpcd: removed reverse map on 41.9.168.192.in-addr.arpa
Jun 15 17:26:46 localhost dhcpcd: Added new forward map from rack2.dgrdn.cust.com to 192.168.4.41
Jun 15 17:26:46 localhost dhcpcd: added reverse map from 41.4.168.192.in-addr.arpa to rack2.dgrdn.cust.com
Jun 15 17:26:46 localhost dhcpcd: DHCPREQUEST for 192.168.4.41 (192.168.1.2) from 00:02:b3:d4:c7:d2 (rack2) via 192.168.4.1
Jun 15 17:26:46 localhost dhcpcd: DHCPACK on 192.168.4.41 to 00:02:b3:d4:c7:d2 (rack2) via 192.168.4.1
Jun 15 17:26:54 localhost dhcpcd: DHCPREQUEST for 192.168.2.41 from 00:02:b3:e7:ec:05 (rack3) via eth2
Jun 15 17:26:54 localhost dhcpcd: DHCPACK on 192.168.2.41 to 00:02:b3:e7:ec:05 (rack3) via eth2
Jun 15 17:29:16 localhost dhcpcd: DHCPREQUEST for 192.168.4.41 from 00:02:b3:d4:c7:d2 (rack2) via eth2
Jun 15 17:29:16 localhost dhcpcd: DHCPACK on 192.168.4.41 to 00:02:b3:d4:c7:d2 (rack2) via eth2
Jun 15 17:29:25 localhost dhcpcd: DHCPREQUEST for 192.168.2.41 from 00:02:b3:e7:ec:05 (rack3) via eth2
Jun 15 17:29:25 localhost dhcpcd: DHCPACK on 192.168.2.41 to 00:02:b3:e7:ec:05 (rack3) via eth2
Jun 15 17:30:02 localhost dhcpcd: DHCPREQUEST for 192.168.4.41 from 00:02:b3:d4:c7:d2 (rack2) via 192.168.9.1: wrong network.
Jun 15 17:30:02 localhost dhcpcd: DHCPNAK on 192.168.4.41 to 00:02:b3:d4:c7:d2 via 192.168.9.1
Jun 15 17:30:05 localhost dhcpcd: DHCPDISCOVER from 00:02:b3:d4:c7:d2 via 192.168.9.1
Jun 15 17:30:05 localhost dhcpcd: DHCPOFFER on 192.168.9.41 to 00:02:b3:d4:c7:d2 (rack2) via 192.168.9.1
Jun 15 17:30:06 localhost dhcpcd: if rack2.dgrdn.cust.com IN TXT "31ec0ea8c148f00cdd8c136ad60ef3d2bc" rrset exists and rack2.dgrdn.cust.com IN A 192.168.4.41
rrset exists delete rack2.dgrdn.cust.com IN A 192.168.4.41: success.
Jun 15 17:30:06 localhost dhcpcd: if rack2.dgrdn.cust.com IN A rrset doesn't exist delete rack2.dgrdn.cust.com IN TXT "31ec0ea8c148f00cdd8c136ad60ef3d2bc": success.
Jun 15 17:30:06 localhost dhcpcd: removed reverse map on 41.4.168.192.in-addr.arpa
Jun 15 17:30:06 localhost dhcpcd: Added new forward map from rack2.dgrdn.cust.com to 192.168.9.41
Jun 15 17:30:06 localhost dhcpcd: added reverse map from 41.9.168.192.in-addr.arpa to rack2.dgrdn.cust.com
Jun 15 17:30:06 localhost dhcpcd: DHCPREQUEST for 192.168.9.41 (192.168.1.2) from 00:02:b3:d4:c7:d2 (rack2) via 192.168.9.1
Jun 15 17:30:06 localhost dhcpcd: DHCPACK on 192.168.9.41 to 00:02:b3:d4:c7:d2 (rack2) via 192.168.9.1

```

Figure 5.12 Syslog output of DHCP server

Now query for rack2.dgrdn.cust.com and see output of NSLOOKUP. Nslookup must show point new IP address 192.168.4.41 for host rack2.dgrdn.cust.com



```
C:\ Command Prompt - nslookup
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\teamrack>nslookup
*** Can't find server name for address 192.168.1.4: Non-existent domain
*** Default servers are not available
Default Server: Unknown
Address: 192.168.1.4

> rack2.dgrdn
Server: Unknown
Address: 192.168.1.4

Non-authoritative answer:
Name:   rack2.dgrdn.cust.com
Address: 192.168.4.41

>
```

Figure 5.13 NSLOOKUP output of web server (after moved)

Open the website running on the webserver after moved it to 192.168.4.0/24 subnet. I open website in client's internet explorer. Output like:

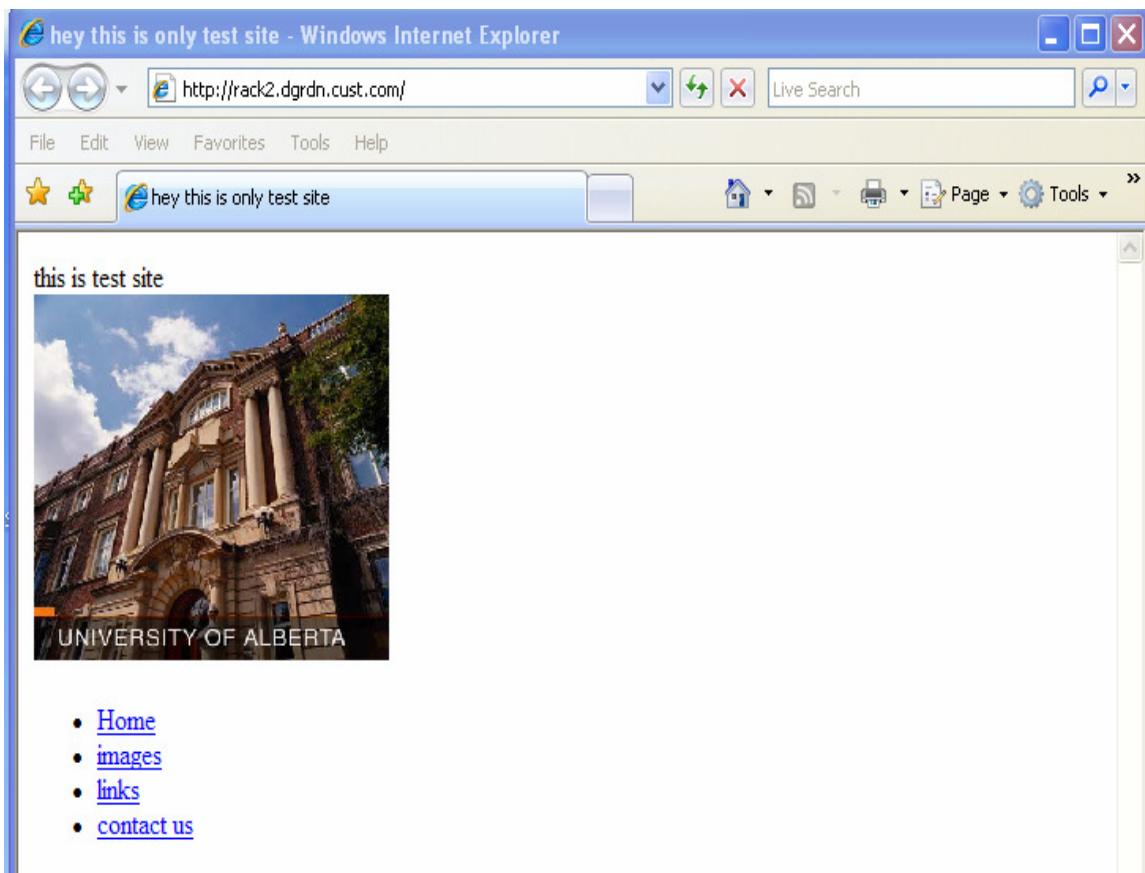


Figure 5.14 Opened webpage of web server (after moved)

This is one more example in addition to nslookup to show successful moving of webserver from one subnet to another subnet.

See how nslookup gives some query's response from domain nameserver and some from sub domain nameserver.

The screenshot shows a Linux desktop interface with a window titled "nslookup". The window contains the output of a DNS query for the domain "ashish.dgrdn". The output shows several entries for "ashish.dgrdn.cust.com" and other hosts like "www.cust.com" and "ns.cust.com". The terminal window has a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". Below the menu is a command line input field. The bottom of the window shows a toolbar with icons for file operations and a status bar with the user's name and session details.

```
> ashish.dgrdn
Server:      192.168.1.4
Address:     192.168.1.4#53

** server can't find ashish.dgrdn: NXDOMAIN
> ashish.dgrdn.cust.com
Server:      192.168.1.3
Address:     192.168.1.3#53

Name:  ashish.dgrdn.cust.com
Address: 192.168.1.3
> rack2
Server:      192.168.1.3
Address:     192.168.1.3#53

Name:  rack2.dgrdn.cust.com
Address: 192.168.9.41
> rack3
Server:      192.168.1.3
Address:     192.168.1.3#53

Name:  rack3.dgrdn.cust.com
Address: 192.168.2.41
> www
Server:      192.168.1.4
Address:     192.168.1.4#53

Name:  www.cust.com
Address: 192.168.1.4
> ns
Server:      192.168.1.4
Address:     192.168.1.4#53

Name:  ns.cust.com
Address: 192.168.1.4
> team4
Server:      192.168.1.4
Address:     192.168.1.4#53

Name:  team4.cust.com
Address: 192.168.1.4
> ashish.dgrdn.cust.com
Server:      192.168.1.3
Address:     192.168.1.3#53

Name:  ashish.dgrdn.cust.com
Address: 192.168.1.3
> www.dgrdn.cust.com
Server:      192.168.1.3
Address:     192.168.1.3#53

Name:  www.dgrdn.cust.com
Address: 192.168.1.3
> 
```

Figure 5.15 NSLOOKUP of other residing host on Network

5.5 Performance measurement

TTL role in DNS operation is important. Performance of the nameserver is dependent on used value to define TTL. When client or nameserver perform lookups they stored lookups data until defined TTL is expired. If TTL doesn't expire then clients or nameservers never able to reach to the authoritative servers of domain. Nameserver use cache data stored in local database for same lookups queried again.

First keep TTL= 600 and move web server between subnet 192.168.3.0, 192.168.4.0 and 192.168.9.0. and measure T1 and T2 time.

Where, T1: This time is actually taken by DHCP server to assign IP address to Web server.

When any DHCP enable host connect to network it broadcast request to all available DHCP server in network. Host will get IP from DHCP server by exchanging some message with DHCP server. This T1 is same time taken by DHCP server to process host request and give valid IP address, DNS server, domain name, subnet mask and default gateway etc.

Where, T2 is the time taken by the webserver to come up in the service after moved.

T2: summation of the time taken to perform following tasks.

- 1) Taken time by the DHCP server to assign IP address to the host.
- 2) Time taken by the DHCP server to send dynamic updates to Sub Domain BIND server.
- 3) Time taken by BIND sub domain server to process dynamic update sent by the DHCP server.

Once the DHCP server assigned IP address to host, it writes lease to the lease file. Now the DHCP server sends the dynamic updates to the BIND sub domain server and writes the data to forward and reverse zone data file. The DHCP server is not smart to write the data directly to zone's forward and reverse data file but first it sends updated data to journal file. This file is created automatically when first time DHCP server sends dynamic update to BIND sub domain server. This file created in same directory where zone data file is stored. Journal file will put extension .jnl to zones forward and reverse map file. In our example created journal file is dgrdn.cust.com.jnl and in-addr.arpa.jnl. Before journal file dump the updated data in zone file it will wait for 15 minutes. But interesting question is how we get reply of query before the journal file dump data into zone data file? Answer is too simple from journal file. Journal file is responsible to respond any query done by client before it dumps data to the zone data file.

```

> rack2.dgrdn
Server: Unknown
Address: 192.168.1.4

Non-authoritative answer:
Name: rack2.dgrdn.cust.com
Address: 192.168.4.41

> rack2.dgrdn
Server: Unknown
Address: 192.168.1.4

Non-authoritative answer:
Name: rack2.dgrdn.cust.com
Address: 192.168.9.41

> rack2.dgrdn
Server: Unknown
Address: 192.168.1.4

Non-authoritative answer:
Name: rack2.dgrdn.cust.com
Address: 192.168.3.41

```

Figure 5.16 NSLOOKUP of webserver when moved to different subnet

Reading with TTL=600 seconds = 10 minutes

T1 (minute: second)	Web server moved to subnet	T2 (minute: second)
0:17	192.168.4.0	2:18
0:19	9.0	3:55
0:15	4.0	4:16
0:24	9.0	4:10
0:26	4.0	4:29
0:23	9.0	4:32
0:15	4.0	4:07
0:23	9.0	4:32
0:18	3.0	3:48
0:17	4.0	4:27
0:28	9.0	1:00
0:13	3.0	3:44
0:15	4.0	4:27
0:20	9.0	4:16
4:08	3.0	4:18
0:24	4.0	2:24
0:29	3.0	2:53
0:20	4.0	3:40
0:20	3.0	4:10
4:32	4.0	4:32

0:25	3.0	4:22
------	-----	------

Table 5.4 Time reading of web server when TTL =600

Average time to come up the webserver in network when TTL= 600 second is 3.47 second

Readings with TTL = 300 seconds

T1 (minute: second)	Web server moved to subnet	T2 (minute: second)
0:16	192.168.3.0	1:03
2:32	9.0	3:45
0:12	4.0	1:30
0:27	3.0	1:58
2:28	9.0	2:28
0:28	4.0	1:27
0:23	3.0	1:16
0:22	9.0	1:49
0:27	4.0	0:36
0:23	3.0	1:53
2:20	9.0	4:18
0:19	3.0	1:39

Table 5.4 Time reading of web server when TTL =300

Average time to come up the webserver in network when TTL= 300 second is 1.58 second

By observing above reading we can say to come up the web server in designed WAN link can be minimized by decreasing value of TTL. Here during TTL= 600 webserver is taking average 3.14 second to come up in service while when TTL=300 second time is 1.58 second. Also one interesting reading is 4.32 second, this is maximum time taken by webserver to come up in network. I haven't got any reading which need more than 4:32 seconds which is appropriate for live environment.

Chapter 6 Conclusion

In this project I successfully implement application to move web server from one network segment of WAN link to another network segment. The moved web server is using DDNS feature to update it self in DNS server. DDNS feature is very useful to set up client machines with a static DNS name even if the IP address is dynamically assigned by DHCP server. It circumvents needs of static IP addresses when user wants to host the webserver or any other internet application. DDNS enables to update a DNS server automatically each time the IP address changes. So anyone who wants to access application running on the *name.dgrdn.cust.com*'s server can reach by using the domain name despite the changing IP addresses.

Reference

BIND 9 Administrator Reference Manual by Internet Systems Consortium
DNS and BIND, 5th Edition by Paul Albitz, Cricket Liu
The DHCP Handbook Second Edition by Ralph Droms and Ted Lemon
<https://www.isc.org/>
<http://www.zytrax.com/books/dns/>
<http://www.bind9.net/dns-links>
<http://tldp.org/HOWTO/DNS-HOWTO.html>
<http://www.madboa.com/geek/soho-bind/>
<http://www.ubuntugeek.com/dns-server-setup-using-bind-in-ubuntu.html>
<http://ulyssesonline.com/2007/11/07/how-to-setup-a-dns-server-in-ubuntu/>
<http://www.debianadmin.com/howto-setup-dhcp-server-and-dynamic-dns-with-bind-in-debian.html>
<http://www.linuxfromscratch.org/blfs/view/svn/server/bind.html>
<https://help.ubuntu.com/community/BIND9ServerHowto>
<https://help.ubuntu.com/8.10/serverguide/C/dns-configuration.html>
<http://www.mirrorservice.org/sites/mirror.centos.org/3/docs/html/rhel-rg-en-3/ch-bind.html>