Ecole Doctorale EDITE

Thèse présentée pour l'obtention du diplôme de
DOCTEUR DE L'INSTITUT NATIONAL DES TELECOMMUNICATIONS

Doctorat délivré conjointement par L'Institut National des Télécommunications èt l'Université Pierre et Marie Curie - Paris 6

Specialité :
INFORMATIQUE et RÉSEAUX

Par
Sandoche BALAKRICHENAN

An Autonomic Simulation Platform for Studying and Optimizing Addressing Issues in Next Generation Networks

Soutenue le 25 Avril 2008 devant le jury Composé de :

Pr. Guy Pujolle (PARIS VI)	Président
Pr. Andrè-Luc Beylot (ENSEEIHT)	Rapporteurs
Dr. Frederica Darema (National Science Foundation)	
Dr. Pierre HOUEIX (Thomson)	Examinateurs
Dr. Jean-Pierre PROST (IBM)	
Pr. Pierre Vincent (MOBKIT)	
Dr. Michel Marot (T & M SudParis)	
Pr. Monique Becker (T & M SudParis)	Directeur de thèse

N ° de thèse : 08INT003

To my Parents and to my Brother

Abstract

An Autonomic Simulation Platform for Studying and Optimizing Addressing Issues in Next Generation Networks

by

Sandoche BALAKRICHENAN

In this thesis we focus on studying the addressing issues in Next Generation Networks; between traditional telephony and the Internet world. Addressing users and services uniquely across network boundaries without exploding the address space is one of the key challenges of NGN. ENUM is one of the main technologies which use a single address identifier for communicating with multiple communication services across different types of networks.

ENUM is a new protocol designed to translate numbers into information by using the Domain Name System architecture. Its success will depend on whether the DNS can achieve a performance similar to the database used in the current classical voice services.

In this thesis we designed a new way to measure and model the local performance of DNS servers of the French ENUM model. We also modeled and measured the IP links connecting the DNS namely; Resolver, Cache and Authoritative servers. Finally we created a simulation model which enables us to simulate ENUM traffic. We then use the parameters obtained from the previous two models as input values into this simulator. The numerical results obtained from the simulation, was compared to real measurements in order to validate the global model. We plan to use the simulator to study different scenarios by varying different parameters.

Difficulty in studying real ENUM implementation involving different network infrastructures, architectures, applications and services used in NGN is a real motivation for us to build a simulation platform. This platform needs to be autonomic, so that it can adapt itself depending on the input configuration. In this thesis work we explain how the simulation model is designed to be an autonomic simulation platform. Different experiments are run using the simulation model, and we present how the solutions obtained from the simulation can be used as recommendations back in real implementations.

Resumé

Une Plateforme de simulation autonomique pour étudier et optimiser la solution des problèmes d'adressage dans les NGN

par

Sandoche BALAKRICHENAN

Cette thèse traite des problématiques d'adressage pour les réseaux du futurs; entre les réseaux téléphonique traditionnels et le monde de l'internet. L'adressage unique des usagers au delà des limites des réseaux, et sans faire exploser le nombre d'adresses utilisées est un enjeu essentiel des NGN. ENUM est une des principales technologies qui utilise une seule adresse pour plusieurs services de communication et à travers différents types de réseaux.

ENUM est un nouveau type de protocole prévu pour permettre l'usage d'une seule adresse pour plusieurs noms de domaines différents, tout permettant à l'architecture DNS de les indexer. Son succès va dépendre de si le DNS peut avoir une performance similaire à celles des bases de données utilisées pour la téléphonie classique.

Dans cette thèse nous proposons une nouvelle façon de mesurer et de modéliser les performances des serveurs DNS de l'implémentation française d'ENUM. Nous modélisons et nous mesurons aussi les performances des liens IP connectant les différentes entités du système DNS : système de recherche, base de donnée, cache et serveur autoritaire. Enfin nous avons développés un modèle de simulation qui nous permet de simuler le trafic ENUM. Nous avons ensuite utilisé les paramètres obtenus avec les deux modèles précédents comme valeurs d'entrée de ce simulateur. Les résultats numériques obtenus avec cette simulation ont été comparés avec des mesures réelles afin de valider le modèle global. Nous avons prévu d'utiliser le simulateur pour étudier différents scénarios en faisant varier les différents paramètres.

La difficulté dans l'étude de l'implémentation réelle d'ENUM est que cette architecture implique différentes infrastructures, architectures, applications et services qui seront utilisés dans les réseaux du futur. C'est donc très intéressant pour nous d'avoir construit une plateforme de simulation. Cette plateforme doit être autonome et donc auto-adaptative à la configuration voulue. Dans ce travail de thèse nous expliquons comment le modèle de simulation a été conçu de façon à ce que ce simulateur soit autonome. Différentes expériences ont été faites avec ce modèle et nous avons présenté comment les solutions obtenues avec la simulation peuvent aider à améliorer les implémentations réelles.

Contents

List of Figures

List of Tables

Acknowledgements

It has been three years since i started my dissertation at Université Pierre et Marie Curie and Institut Nationale des Télécommunications. When i sit back and think about the number of people who influenced me and helped me complete this dissertation, i am overwhelmed! There is no doubt that this would have been impossible without their help. I hope that i can remember everyone who helped me through this difficult yet rewarding process.

First and foremost, i would like to thank my advisor, Professor Monique Becker (Director of research laboratory CNRS SAMOVAR), where without her advice and help i will not be here defending my PhD. It was an extraordinary piece of good fortune that led to my becoming her student. She has been an ideal advisor in every respect, both in terms of technical advice on my research and in terms of professional advice. I again thank her for her encouragement, guidance and assistance, not only during my PhD but also from my graduate studies where she was my course coordinator and also while doing internship under her.

I would like to thank Professor Guy Pujolle, (Professor at Paris VI university) who had accepted to be the president of my thesis jury.

I would like to thank Dr. Frederica Darema (Senior Scientist at NSF) and Professor André-Luc Beylot, (Professor at ENSEEIHT, Toulouse) who had accepted to be the reporters of my thesis and also be part of the dissertation committee.

I would like to thank Dr. Pierre HOUEIX (at Thomson), Dr. Jean-Pierre PROST (at IBM) and Professor Pierre Vincent(who has started his own enterprise after heading the LOR department at INT) who had accepted to be the examinaters of my thesis and also be part of the dissertation committee.

I would like to thank Dr. Michel Marot who had always been helpful with his suggestions and technical advice. I had a great time with him working in the Mediatvcom project and i learnt a lot from him during my extended association with Mediatvcom.

I would like to thank Dr. Thomas BUGNAZET who has been very helpful in the earlier part of my thesis in explaining me about the complexities of modeling a system and developing a simulator based on a real system.

The Performance group set the stage for this dissertation and numerous discussions i had with its members greatly improved the quality of my work. I thank its members, past and present - Vincent Gauthier, Mahendiran Prathaban, Ashish Gupta, Armando garcia, Pierre Delannoy, Alexandre Delye, Carlos Moreno, Mabel VASQUEZ BRISENO and Cherif Diallo for an intense and fun intellectual environment.

The Bureaucracy at INT can make life difficult for graduate students. I would like to thank Chantal Lemaire and Françoise ABAD for their friendly attitude and approachability which made it easy to get all administrative problems solved.

I owe a special debt of gratitude to my parents and family. They have, more than anyone else, been the reason i have been able to get this far. Words cannot express my gratitude to my parents and brother Sateesh who give me their support and love from across the seas. They instilled in me the value of hard work and taught me how to overcome life's disappointments. My wife Sandhya, gives me her selfless support and love that make me want to excel.

Curriculum Vitæ

Sandoche BALAKRICHENAN

Education

1999	University of Madras B.E., Computer Science Engineering
2005	Institut Nationale des Télécommunications M.Sc, Computer and Communication Networks
2008	Institut Nationale des Télécommunications and Université pierre et marie curie Ph.D., Information Technology and Networks

Chapter 1

Introduction

I kept six honest serving men. They taught me all I knew. Their names are "What" and "Why" and "When" and "How" and "Where" and "Who".

- Rudyard Kipling

The last thing one discovers in writing a book is what to put first.

- Blaise Pascal, Pensees, I., Chap. 19

1.1 Motivation

The most important inventions in networking and communication can be classified into three categories:

1. Public Switched Telephone Network
2. Wireless Cellular Telephony and
3. Internet

Public Switched Telephone Network (PSTN) has been in use for more than a century. While PSTN growth in developed countries is getting reduced, and as it is also slowly reaching a saturation point in other parts of the world, there have been other trends in networking and communication over the last twenty years which is having a rapid growth. Two stand out prominently; the rapid global expansion and use of the Internet for information access and the massive growth in the use of wireless cellular telephony for communication.

The first trend is the meteoric growth of World Wide Web (WWW) [30] and large amount of information that one can find on it. The second trend, towards wireless connectivity has been fueled by advances in communication hardware and cellular technologies that have enabled large scale wireless telephone networks. Figure 1.1 shows the growing trend for this three categories. In order to get connected either in the telecommunication (PSTN and Wire-

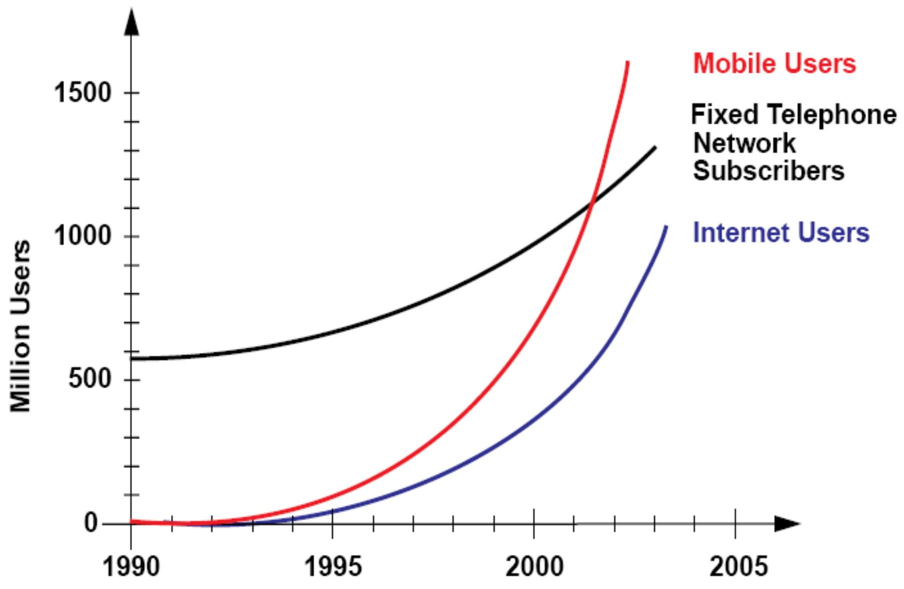


Figure 1.1. User Numbers for three major trends of communication {*Source:ITU News 2/2002*}

less cellular telephony) or in the Internet sectors, the end points must be identified. Names and numbers are how these end points are identified and they are also called *identifiers*. Names and numbers represent the resources that enable providers of telecommunications and Internet services to identify effective endpoints (an e-mail address for example is an endpoint, a telephone number is also an endpoint). For a telephone call to be delivered, two endpoints of the connection must be identified. Can we imagine making a telephone call or sending a mail without a number or a name? This gives us the idea how important, addressing schemes like names and numbers are for communication.

A single person can have different Personal Communication Identifiers(PCI) such as phone number, mobile number, email address, instant messaging etc., by which he can be connected to. PCI such as telephone and mobile number were initially used for voice services and these services were provided by a circuit switched network infrastructure, while email and instant messaging services were used for data services and provided by a packet switched network infrastructure. In today's anywhere/anytime communication world one wants to be connected through all three means of communication technologies (PSTN, Internet and Cellular) and also should be interoperable between the two different network infrastructures.

One would therefore expect that the combination of these two infrastructures - *packet switched* on one hand and *circuit switched* on the other - would result in a potent combination which would dominate the marketplace. This combination is given by Next Generation Network (NGN) which provides a migration path from traditional network to an Internet Protocol (IP) based network, while maintaining existing services. The general idea behind

NGN is that there is one network which transports all information and services such as voice, data and video and this network is built on the top of the IP.

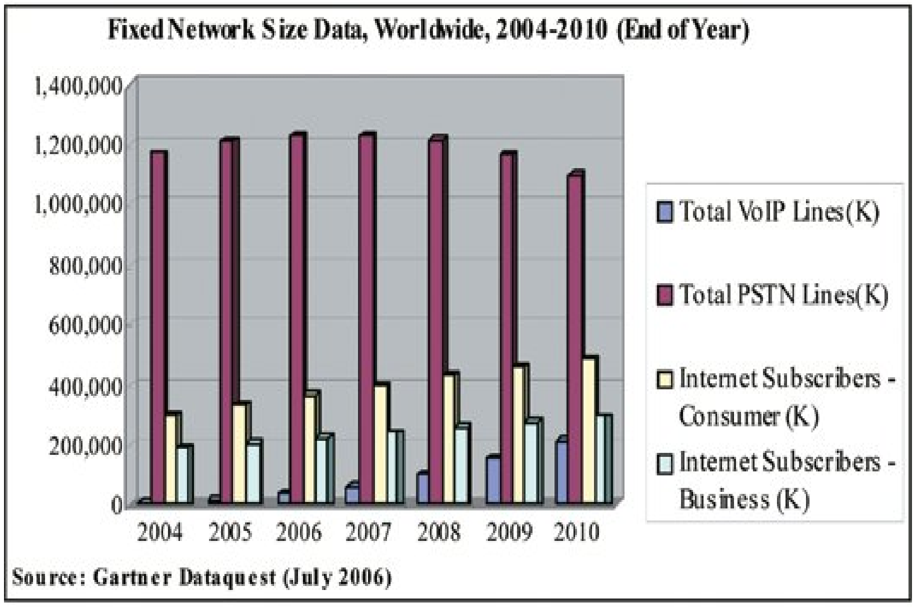


Figure 1.2. Worldwide fixed network size data {*Source:Gartner Dataquest 7/2006*}

Figure 1.2 indicates that the number of global PSTN subscribers will reach 1.3 billion and as indicated previously it will reach its saturation point and then register a negative growth. On the other hand we can see from figure 1.1, there is going to be a rapid growth in voice and data communication over IP. It is easy to integrate these new services in NGN. But is it possible to ignore the huge penetration in user numbers in the communication world already made by PSTN. Migration of traditional PSTN to all IP NGN is an attractive option, but the question is it really feasible?

So the only answer is *NGN convergence*. Here convergence can be defined as "*the process of integration of previously independent industries of telecommunication, Information technology and media*". Integration of these independent technologies is a very broad topic. Our focus is only on one issue; addressing issue between the two different network infrastructures - Packet switched and Circuit switched.

Each of this different network infrastructures has different addressing structures not to mention the terminologies. PSTN addressing is done by E.164 numbers (E.164 number is used to specify telephone numbers) while in the IP world it is done by IP addresses. To achieve the vision of *anywhere/anytime* communication we need to have a seamless convergence between these two different addressing formats. We believe that there are two important and closely related reasons that hamper the realization of this vision.

Heterogeneity When we talk about the NGN convergence between circuit switched and packet switched, its not only the two different network infrastructures, but there is a tremendous diversity in architectures, applications, services etc. Interworking between these two different infrastructures requires that applications and services involved will need to address issues such as numbering, naming and addressing. They should be able to address users and services uniquely and across network boundaries without exploding the address space.

Quality of Service (QoS) There are a number of QoS-related aspects that need to be addressed as NGN is deployed [81]. From a user perspective, quality of service may relate to the communication service itself, eg voice quality, picture quality, delay, speed, etc. While considering the existing PSTN network, the performance of real time databases in the telecom world is quite good. PSTN call establishment is expected to take less than 200 milliseconds [3]. Customers used to PSTN performance will have the same expectations in the IP world also. Another important issue is the reliability. PSTN connection set up are considered to be quite reliable. Unfortunately use of reliable data transport protocols for connection establishment will result in an increase in response time performance, which become too large compared to reliability. It is required that identification and routing over traditional PSTN and new NGN architecture should be accelerated to obtain optimum performance and reliability compared to telephony world.

The vision of anywhere/anytime communication in NGN poses several challenges. In this thesis, we identify a protocol proposed by Internet Engineering Task Force (IETF), which helps to solve the problems in the addressing issues in NGN. Even though the protocol (ENUM) which we have identified, proposes an innovative idea to solve the addressing issues, it has its drawbacks. Our methodology is classified in three phases; *analysis, simulation* and *optimization*, helps us to to study scenarios using ENUM for address resolution between the packet switched and circuit switched infrastructures. The end result is an autonomous simulation platform which can be used for studying addressing issues in NGN and propose promising solutions to improve the performance.

The rest of this section gives a functional overview of NGN architecture and elaborates the issues of heterogeneity and QoS in NGN. In section 1.2 we discuss the three fundamental challenges to build an autonomic simulation platform to study and solve the address issues in NGN. Section 1.3 presents an overview of our contributions and describes the organization of the rest of the thesis.

1.1.1 NGN Architecture

NGN concept defines telecommunications network architectures and technologies. It describes networks that cover conventional PSTN type of data and voice communications as well as new types of service such as video. All information is carried in packet switched form, as is done in the Internet. Packets are labeled according to their type (data, voice, video, etc) and forwarded in the network based on their QoS and security parameters. The functionality of NGN has been divided into different functions (see Fig. 1.3) namely; Service functions, Transport functions, Management functions and End-user functions. The NGN makes a clear separation between transport and service layer. This is being done to allow

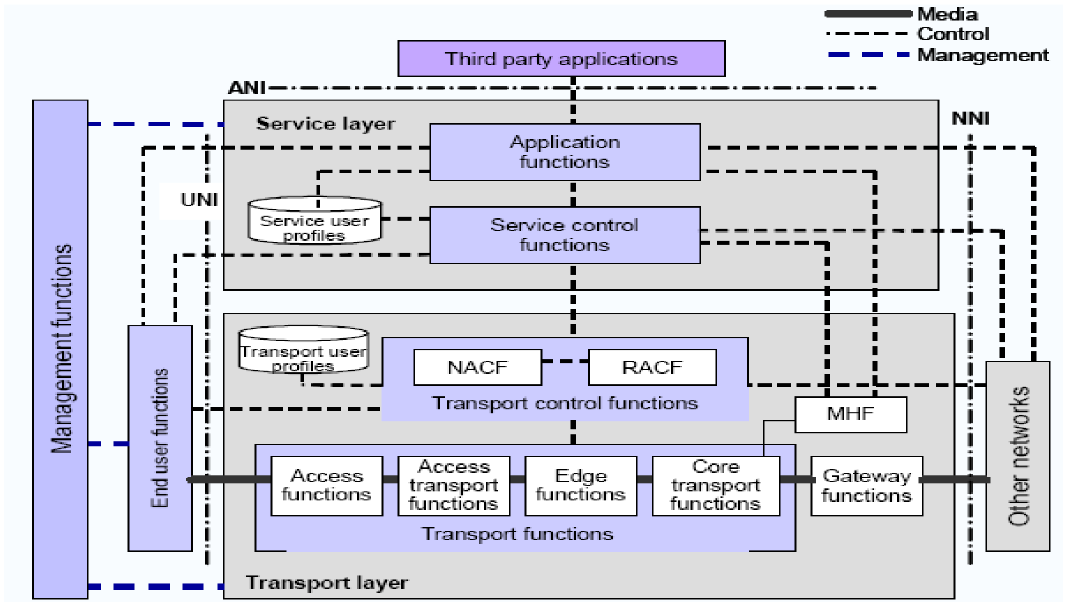


Figure 1.3. NGN Architecture Overview *(ITU-T Y.2012)*

smooth introduction of new services. When a new service need to be provided it is defined at the service layer without considering the transport layer, thus making services independent of the transport technology [26] [90]. The transport layer functions provides connectivity for all components and physically separated functions within an NGN. The transport technology that is used is IP. The service functions are used to give session based (IP telephony and video conferencing) and non session based (video streaming and broadcasting) services. Session layer includes *application functions* to support open Application Programming Interfaces (APIs) to enable third party service providers to use NGN capabilities.

The management functions enable an NGN operator to manage the network and provide NGN services with the required quality, security and reliability. These functions are distributed to each functional entity and they interact with the network element management, network management and service management functional entities [52] [16].

Interfaces to the end user are either physical or functional as shown in the figure 1.3. The NGN architecture supports all kinds of customer equipment categories from single user telephones to complex corporate networks, irrespective of the end user equipment (mobile or fixed).

1.1.2 Heterogeneity issues in NGN

Figure 1.4 shows the difference between present day networks and NGN. Each of the present day networks (data, voice and video) were designed to run in parallel. NGN aims to go back to the simplicity of one single network and deploying one network platform capable of supporting all traffic types over IP. The difficulty is in how to inter operate between the different heterogeneous networks.

For calls where both endpoints are in the circuit switched network, traditional routing methods for the circuit switched network are used. Similarly when both endpoints are in packet switched network, Domain Name System (DNS) is used. IP telephony has also matured to a stage where most of the fundamental technology, protocols and standards are available. However, in a hybrid IP-Circuit switched network like in the case of NGN, routing is more complex. Before a call between the different networks can be established, an appropriate gateway must be located. The selected gateway must be able to complete the call to the desired destination. There may be several gateways and selecting the most suitable one is important. In the routing of a call or a packet, the *switch* (for circuit switched

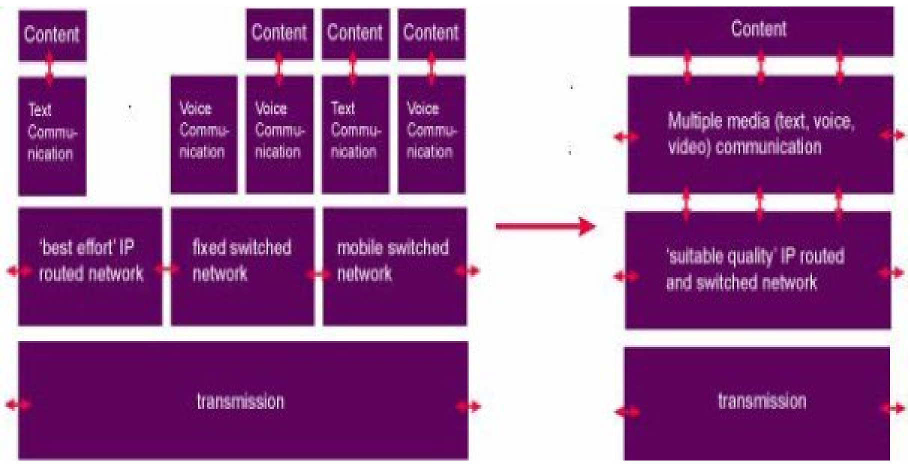


Figure 1.4. Transition from Current Network to NGN networks *(Source: Antelope Consulting)*

networks) performs routing based on the E.164 number in the PSTN. The *router* (for packet switched networks) however performs routing based on the Internet IP address. Besides the IP address for routing, a domain name and a user address are used to identify the target on the Internet. The user address is an identifier with a higher degree of abstraction and takes the form of "user@domain" which is used as the mail address or SIP (Session Initiation Protocol) address.

Internet users who have keyboards used domain names and user address as identifers, which are easier to remember. However, the problem arises when the networks are inter-connected. Callers on the PSTN have no keyboard and a scheme for entering characters using number keypad would be too complicated. This limits the PSTN users to entering numeric names. Consequently, an IP terminal must have a telephone number to be accessible from the PSTN. The problem was recognized by the TIPHON [8] project, which chose to equip IP-terminals with an E.164 number. For calls within the IP network, other types of addressing can be used. Unfortunately, this would require the user to know on what type of network the destination is. When IP telephony is largely deployed, users do not necessarily even know the underlying technology of their own connection.

To establish a call to a terminal on an IP network, the destination IP address must be known. Alternatively the terminal can be identified by a host name, which is translated to an IP address by DNS. As terminals in PSTN are equipped with an E.164 number, a new mapping is required: from an E.164 number to an IP address.

When the circuit switched network and IP telephony networks are interconnected, new call scenarios arise. Since the originating network and destination network can be of two types, there are four basic call scenarios: PSTN-PSTN, PSTN-IP, IP-PSTN and IP-IP. When calls are setup, the first task is to determine the type of the destination network. As explained earlier in this case mapping from E.164 name to IP address will be required. The required mappings could be solved with some type of directory. At a minimum, the mapping from E.164 number to network type and IP address must be supported. The directory must be scaleable to store large amounts of mappings, possibly for all telephones in the world. It must be capable to reply to a high rate of lookups, for each call that is set up. In practice, the directory must therefore be distributed. The directory must also propagate updates rather quickly when the information changes. Additionally the mapping is expected to be used with several different services. In addition to voice calls, the IP network allows for video conferencing and e-mail among others. Some method of locating the available contact modes and services is desired [**?**].

There are multiple issues in NGN. In the above paragraphs we have tried to explain the issues that could hinder the possibilities of interconnection between a hybrid IP-circuit switched call set up in NGN.

1.1.3 QoS

NGN supports a converged framework using IP based technologies on top of various other technologies like fixed, IP and mobile technologies (see Fig.1.4). There are a number of QoS related aspects that need to be addressed in NGN. From a user's point of view, the QoS may relate to the communication service itself like voice quality in the call, picture quality in multimedia transfers, delay etc.

For voice telephony it will always be important to control delay, jitter, error rate and packet loss, otherwise the user experience is likely to suffer. This is also true for video telephony, which also demands a higher guaranteed bandwidth to maintain a certain quality of service.

The voice planning process in traditional telephony, whilst complex, is relatively straightforward. Transmission, switching and voice communication services are provided by dedicated networks. It is possible to ensure voice quality by allocating dedicated bandwidth for the duration of each call.

In the NGN environment, a single network supports multiple service types (see Fig. 1.4). Voice quality gets worse when a call crosses several different networks. There may be need for conversion to the packets according to the different networks. Between each pair of networks a conversion introduces many milliseconds of delay. In order for NGNs to achieve the same grade voice quality as that of traditional PSTN, there needs to be either a system of network protocol prioritization or allocation of additional capacity.

The reliability and performance of a Voice over IP (VoIP)service depends on a number of elements. Our objective is to look at identifier issues which could degrade the performance. One of the main issue is the *number portability* as discussed below.

When the two network technologies, PSTN and IP, are interconnected, a need for number portability across different networks come forth. The existing number portability solutions in the PSTN provides solutions for numbers which are ported from PSTN domain to IP domain. Limited number portability can be implemented in IP domain by call forwarding.

For the time being, there are no specific solutions for number portability across the network types. A number that is moved to the IP network can be handled in a normal way in the number portability solutions on the PSTN. The number portability databases are updated with a routing number that directs a call to a gateway.

Another important performance issue that arises, is the time taken for the address mapping from E.164 numbers to IP addresses. In case of loss of packets during the address resolution process, this can really impact the performance of the voice calls. The address mapping database should be able to scale its performance under large query loads. Also as new services are integrated, the amount of data to be managed increases. The service providers should be able to add, delete or modify data in real time with no interruption of service. Failure to do so increases the overhead of managing the data and can add undue delay and service interruption, eroding the benefits and efficiencies, the converged NGN can provide.

1.2 Preview: Three Fundamental Challenges

The challenges to addressing in NGN arises primarily because of network heterogeneity, different addressing structures and different characteristics of the underlying technologies involved. These lead to problem that can degrade the performance of communication across different networks in NGN.

Even though there have been different technologies which were proposed to interwork between different networks, identifying the appropriate technology for NGN addressing convergence was the first challenge. As we have already explained NGN is composed of heterogeneous entities. Studying such heterogeneous real networks is quite an impossible task. So there arises a need for a simulation tool. The simulation tool that has to be implemented, need to be benchmarked with an empirical model. The second challenge that arises is how to measure and model a real implementation model. NGN is quite evolving. In order to adapt to the evolving nature of NGN, the simulation model has to be designed to adapt to new applications and conditions, thus the third challenge is to build an autonomic simulation platform and then use the simulation platform to study and improve the performance of NGN convergence scenarios.

The rest of this section summarizes the challenges and outlines our solutions to them.

1.2.1 Challenge #1: Identifying the appropriate technology for NGN addressing

The convergence towards NGN requires that different network users using different network technologies, can communicate with each other and access resources on distinct network infrastructures. This requires the interworking of different Naming, Addressing and Numbering systems.

PSTN and Internet handles different addressing schemes. The basic concept of PSTN is a fully qualified phone number such as +33 1 6076 4000. This is defined by an International Telecommunication Union standard known as E.164. All ITU-T E.164 numbers are 15 digits or fewer and start with a country code (+1 for countries on the North American numbering plan, +33 for France, and so on), suballocated by the corresponding national bodies under nation-specific policies. The Internet on the other hand employs several kinds of identifiers. They may be mail address like mailto@int-edu.eu or a domain name such as www.int-edu.eu. All of these name have to be resolved to find its IP address and this is done by a highly distributed database called DNS.

There are different IP based services such as VoIP and Push-to-talk for voice, IPTV for media, email and instant messaging for data communications. To be able to implement these services across different technology such as Global System for Mobile communications (GSM) and Code Division Multiple Access (CDMA) for wireless telephony and fixed line we need a simplified addressing mechanism which can work across multiple technology platforms.

With multiple PCIs for a single person there is no best way to contact someone. The question arises while contacting, should it be done on ones cell phone or his home land line phone or should it be through a instant message on his cell phone or text message to his computer? Paul Mockapetris says the best way is to let the network figure out how the person to be contacted to [91]. Here comes the problem of signaling which has to keep track of all the potential communicating parties, their equipment and their services and selecting the right combination for each contact.

Our approach to solving this problem is by reviewing the literature for a technology which could solve all the above mentioned problems. Developing on our own a protocol or application to solve these issues is not pragmatic enough. So we proceed by meticulously analyzing the existing technologies for NGN addressing and we have selected with tElephone NUmber Mapping (ENUM). We have put forth the recommendations why we chose ENUM in chapter 2.

ENUM, as a standardized technology, offers solutions for many known problems within an NGN. It provides mechanism for the mapping of E.164 telephone numbers into Uniform Resource Identifiers (URIs) and thereby enables new service features within an IP based network. The deployment of ENUM will support the smooth migration of service discovery and signalling functions from the existing Signaling System 7 (SS7) databases and protocol mechanisms to an ENUM based NGN. It also solves the issues of multiple identifiers for a single user as it uses a single E.164 number to map to different types of URI's for a single user.

1.2.2 Challenge #2: Studying the performance of a real system

ENUM uses DNS for NGN convergence. With so many different addressing look ups in the DNS for address mapping, the challenge arises as what will be the impact of this new address look ups on DNS. The constraints that we look into are

- Load supported by DNS Servers
- Delay caused by DNS queries for address mapping and
- Availability

In the past several methodologies have been used for evaluating the performances of computer systems [50], [67]. The problem is that it is not always possible to study the real system and come up with promising solutions. There are several reasons for this, like the size, studying a new system which is not yet built, etc. The feasible method is to scale down the real systems and experiment via theoretical studies [38].

The analytical studies can be done either by a purely analytical approach or simulation based approach. A purely analytical approach, for instance include, Queuing Networks, Markov Chains and Petri nets [40]. Most often, this allows only to deal with simple models, and forces specific assumptions which are infrequently met by real world systems.

By a simulation based approach, a simulation model is built, which is an algorithmic description of the model of the system under study. The simulation is run, according to various input parameters reflecting the actual environment. Most studies of real systems use both approaches, one way to validate the study lying in a careful comparison and a cross-validation between the two.

Our methodology is to first study the real system i.e a French ENUM system. To study the behavior and performance of this real system we used both analytical and simulation approaches. We used analytical models to study certain behaviors and obtained parameters. We then built a simulation model, which is an algorithmic description of the French ENUM system. The parameters derived by these analytical models were used as input in the simulation model to make the simulation accurate as possible. Finally we validated the results of the simulation as well as the measurements made on the real system, thus making sure that the simulation model built is a good approximation of the real system.

Analysis of the empirical French ENUM model is done in chapter 3 and explanation and validation of the simulation model is explained in chapter 4.

1.2.3 Challenge #3: Constructing an autonomous Simulation Platform to study NGN issues

ENUM cannot be viewed as a technology that is static. Like DNS it is also evolving, maybe more than DNS. Initially it started as a convention to address phone number resolution in the IP domain. Then due to policy and security issues public and infrastructure, ENUM was proposed. Around 2004 mostly in Europe when initial deployments of ENUM started, ideas started popping about how ENUM can be used for providing different services as explained in subsection 4.1.3, which can take advantage of ENUM. To quote from [94] "*ENUM community just knows a fraction of features we might see for ENUM in future*".

For building a simulation model to study the French ENUM scenario as reported in chapter 4, it is quite enough that we build a static model and improve the response time by using different techniques, changing the simulation code wherever needed. But on viewing the conditions that ENUM can be used in different possibilities, a static simulation model will be short sighted.

The simulation model built, is designed in such a way that all the input parameters are obtained from configuration files, thus enabling the simulation model to be modified without understanding the software complexity that is used to develop the model. NGN and in particular ENUM is in nascent stage. There are different versions of ENUM, and ENUM could be used in tandem with other services such as SIP. So we want the model to be a platform where new services, which need IP-PSTN convergence, to be added and removed as needed.

In NGN where numerous entities are involved to realize the service functionalities, our belief is an autonomous approach is needed to deal with the complexity. In order for the simulation system to adjust automatically without any external help, we tried to follow the IBM vision of autonomic computing namely: self-configuring, self-healing, self-optimizing and self-protecting.

1.3 Contributions and structure of thesis

In this thesis, we develop an autonomous simulation model and evaluate an architecture of NGN composition, studying issues of address mapping and proposing solutions for optimization. Because there is tremendous heterogeneity and diversity in NGN, and because there is enormous variation in their characteristics, there is no single solution that cures all the addressing problems. Our methodology is depicted in Figure 1.5.

To study and optimize the NGN address resolution issues we used a combination of analysis, simulation and optimization. In this thesis work, all the three phases were done based on an ENUM model. The contributions are presented in all these three phases.

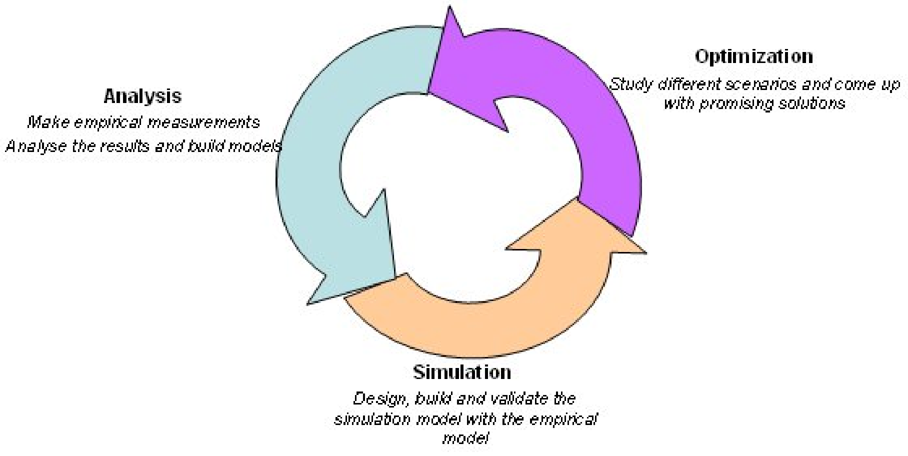

Optimization
Study different scenarios and come up with promising solutions
Analysis
Make empirical measurements
Analyse the results and build models
Simulation
Design, build and validate the simulation model with the empirical model

Figure 1.5. Three phase research methodology involving analysis, simulation and optimization

1.3.1 Analysis

The contribution in this phase is the new methodology we used to study an empirical system. Since ENUM uses DNS we studied the impact of classical DNS in the beginning. Then we studied the impact of DNS with the ENUM extension. For this study we used the real French ENUM empirical model.

We first measured the empirical system, then from these measurements we developed analytical models. Four types of measurements were made:

1. Measurement of the capacity of the different servers used in the system under different loads.
2. Measurement of the loss rate and response time at each of the authoritative servers at a given query rate and under a given DNS environment (Hardware, DNS software type and DNS database configuration). These measurements are needed since the authoritative servers performance depends on the configuration of the system. We wanted to analyse how these servers behave under different loads.
3. Measurements of the two important metrics which impact the performance of the IP links (i.e. delay and loss) connecting the different nodes of the ENUM system. This is required to understand how the network links connecting the system impact the ENUM address resolution
4. Measurement of the global response time of the queries made on the French ENUM model. This measurement is needed to finally benchmark the simulation model built.

For these measurements we have developed and used existing tools.

The analysis of the packet traces obtained from these measurements helped to develop analytical models. These analytical models were verified and validated. Parameters obtained from the analytical models are needed to be used in the simulation tool to emulate the characteristics of the empirical model. At the first phase, our main contribution is the methodology we used and the tools that we developed and modified to study a real system so that finally a simulation model can be built based on the real system.

1.3.2 Simulation

The gathered data and the functional characteristics of the empirical model serve as the starting point for the simulation phase. Contribution in the second phase is designing, building and validating the autonomous simulation platform. The simulation model was verified and validated, so that it can be used as an approximation of the real system. This is the main contribution of this work. According to our knowledge there is not such a simulation platform built to study the addressing issues in NGN.

The simulation model was built keeping in mind the heterogeneity factors of the NGN and the evolving services and applications that need address mapping between different types of networks. All the input parameters were obtained from configuration files, which enables us to try new scenarios, add new applications and technologies and run simulations. An autonomous approach was taken in building the simulation model keeping the mind the four pillars of autonomic computing - self-healing, self-protecting, self-configuring and self-optimizing. We explain in Chapter 4 how the simulation model that is built is autonomous, and we prove it by testing autonomicity on the model from an architectural as well as the functional view. Even though currently the simulation model is based on ENUM, it is designed in such a way that it could be adapted to other applications with similar functionalities, like address mapping between PSTN and IP identifiers using DNS. The simulation platform can also be used to study NGN systems such as IP Multimedia Subsystem (IMS) where Session Initiation Protocol (SIP) registrars are used for identifying the routing destination. The adaptations that would be required here is to add modules for SIP session establishment.

1.3.3 Optimization

Once the simulation model is validated, we study different scenarios and come up with promising solutions in the optimization phase. The objective here is to use the promising solutions obtained from this phase as recommendations back into real implementation. We give some examples for which this simulation platform could be used to answer the questions for a new ENUM implementation such as:

- With which parameters one obtains a satisfactory QoS?
- Which model will give a better QoS? For example can a two tiered model give a better QoS for certain operators or countries.

- What is the influence of algorithms at the cache server? For example does an Adaptive Time to Live (TTL) algorithm obtain a better QoS or is it increasing the processing load at the cache server.
- What is the influence of TTL for the Performance? For e.g. if instead of giving 5 seconds for each request, does 3 seconds give a better performance.
- At how much load does the optimization of the cache server becomes interesting or do we need to increase the capacity of the servers.
- Scalability for various versions of ENUM (like operator and user ENUM etc.)

The autonomic simulation platform that we have developed can help to study the feasibility study of different ENUM enabled services with very few additions made on the existing set up. The approach we have taken to study the ENUM French model can also be used to study other models and use those parameters into the simulator to have an emulation of the real world scenario.

The contribution of this phase is in using the simulation tool we have conducted, for three case studies and from each of this case study we propose some recommendations which could be used in real implementations for improved performance. In keeping in mind that cross layered approaches are needed to leverage all possible optimizations, we also have an example of how our simulation model uses a cross layered approach to further optimize the system.

Below we explain briefly the three case studies we conducted using the simulation model and how the results from these case studies can be used as recommendations in real scenarios.

Case Study 1 We conduct simulation experiments with different Time to Live (TTL) values and number of requests per second. From the experiments conducted we come up with a proposition that response time is inversely proportional to TTL values upto a threshold depending on the configuration of the system. Increasing the TTL value above the threshold does not benefit the response time, but it rather leads to delayed response to updates or modifications in the server.

Case Study 2 Here we conduct simulation experiments to study the performance with respect to the number of hops. Unlike the case of DNS which can benefit the effective use of caching, the number of hops do have much impact on response time in ENUM. We propose a change in the design of the French ENUM delegation model and prove that such a design change could reduce the ENUM resolution time.

Case Study 3 Here we initially run experiments to study at what load for a particular scenario, optimization or increase of resource is needed. Then we use an asynchronous feedback technique from the cache server at the network layer to the resolver at the application layer to further optimize the response time.

The rest of this thesis is organized as follows. In Chapter 2, we discuss related work why we chose ENUM and the basic overview of ENUM is given in this chapter, while the analysis phase is explained in Chapter 3. Chapter 4 and Chapter 5 describes the simulation and optimization phase. Finally in Chapter 6, we present a summary of our work and outline some directions for future research.

Chapter 2

Background and Related work

Contact (not Content) is King.
- Douglas Rushkoff

The Purpose of this chapter is to introduce the reader to related works in NGN and why we chose ENUM for this work to study addressing issues in NGN. We start with a description of how NGN is revolutionizing today's communication, motivate the need for studying and optimizing address mapping across different networks, and discuss several technologies proposed in the literature. In section 2.2, we explain the main signaling protocols for IP telephony and their disadvantages to justify the need for ENUM in NGN signaling. Finally in section 2.3,2.4 and 2.5, we give an overview of ENUM and also discuss various literature studies on DNS in which is the basic database of ENUM.

2.1 Next Generation Network

Historically, Communication was done through PSTN, which was designed to carry voice. As demand for data communications developed, a separate network was built to carry data traffic. These networks were designed to run in parallel. As the network technology developed, the number of networks multiplied gradually. As a result, there are different network platforms today (ATM, IP, X.25, PSTN, Frame Relay etc.).

The problem with multiple network platforms is that it results in different complexities like interoperability, maintenance, operational inefficiencies etc. To counter this plethora of difficulties, NGN was proposed. NGN aims to go back to the simplicity of one single network and deploying one network platform capable of supporting all traffic types, while facilitating new services and streamlining the support structure.

NGN is defined as

- A packet-based network, able to provide services including Telecommunication Services
- Able to make use of multiple broadband technologies
- QoS-enabled transport technologies and
- In which service-related functions are independent from underlying transport-related technologies
- It offers unrestricted access by users to different service providers
- It supports generalized mobility which will allow consistent and ubiquitous provision of services to users.

2.1.1 Addressing Issues in NGNS

To make NGN a possibility, overlapping of perviously separate and distinct technologies is required. This overlapping is termed as *convergence*. Convergence of communications technologies complicates the management of names and numbers, which are used for addressing. Even though there are different types of convergence, we concentrate on PSTN-IP convergence in our work.

The routing of a call or a packet is based on the telephone number (E.164 number) in the PSTN. In Internet, however routing is based on the Internet IP address. For example the domain name "www.int-evry.fr" is accessed by the IP address "157.159.11.8". Similarly email access is also done by IP addresses. For VoIP, performing routing based on the telephone number over the Internet involves storing the telephone number in the IP header, and requires a new function to process this number in all routers in the Internet. These routers must also contain the path table of the telephone number. Besides the IP address for routing, a domain name and a user address are used to identify the target on the Internet. The user address is an identifier with a higher degree of abstraction and takes the form of "user@domain" which is used as the mail address or SIP (Session Initiation Protocol) address [48]. A general Internet address includes those in a URI (Uniform Resource Identifiers) format [19].

For all bilateral communications, contact information is a crucial part. The contact could be obtained either by names or numbers. The internet as we have seen in the previous paragraph is providing well known naming and addressing schemes like domain names and URI's. The PSTN is providing another well known addressing scheme, the public telephone numbers (E.164).

One of the key issue in NGN is how to manage naming and addressing resources across the converged networks (e.g. IP-PSTN). It is easy to integrate services over IP in NGN, whereas PSTN services cannot be integrated. It is not possible to ignore the huge penetration in user numbers in the communication world already made by PSTN. Migration of traditional PSTN to all IP NGN is an attractive option, but the question, is it really feasible? Inter operability between PSTN and IP networks is the pragmatic option.

Any technology which is able to inter operate between different networks should take into account the following details:

- Any technology which tries to inter operate between different networks should be able to support existing addressing infrastructure. This is also mentioned as *Backward Compatibility.* Compatibility with evolving and existing services and systems will be needed in order to meet the needs of end users and service providers.
- Addressing requirements of users where the communication is originating as well as the requirements of the user who is in the receiving end. Thus there should be an interoperability between communication services, devices, networks and service providers.
- Should be able to reach a resource even when the resource is relocated. It can be in cases of change of the provider or a change in the communication channel.
- interworking between services will need to address issues such as numbering, naming and addressing.

Addressing uniquely

Another challenge that arises is having a single PCI that can be used for multiple communication services across PSTN and IP networks. In such cases, one needs a unique identifier which may never change. All the other identifiers can be placed in a public database on the Internet, which can be queried by anyone using the unique identifier.

In the telecommunications and Internet sectors, the fundamental 'service' that has to be traded is a connection, whether it is for person-to-person communication like in the case of VoIP or for entertainment services such as Video on Demand. In order for such services (take the example of a telephone call) to be delivered, the two endpoints of the connection (subscribers), or their locations (telephone devices), must be identified. Names and numbers are how these endpoints are identified; they are so-called 'identifiers'. Names and numbers represent the resources that enable providers of telecommunications and Internet services to identify effective endpoints (an e-mail address for example is an endpoint in its own right but can also relay communications to other endpoints) and/or device locations. To grasp the imperative of numbering and naming, imagine making (or billing) a telephone call without either a number or name.

The fundamental issue related to the significance of current naming and numbering practice is that the supply of both names and numbers is finite, i.e. the resources are, in economic terms, scarce and must be unique, which, in turn, necessitates some form of control system.

In telephony, this scarcity is imposed by convention: under the telephony rules laid down by the ITU Recommendation E.164, telephone numbers should not exceed 15 minus n digits (where n is the number of digits in the international dialling code of a particular country, not including the international prefix). The issue of scarcity of telephone numbers has until now been managed successfully, by a hierarchical system.

With relation to the Internet and also IP addresses, the scarcity is more complicated. For any naming or numbering system to work, it is essential that the names and addresses used cannot be confused with any other, in other words, no one system can have two end points with the same fully qualified number or name.

Quality of Service

The deployment of NGN, using IP connectivity to support fixed, wireless and mobile voice, video, data, and broadcast TV services, provides new opportunities. It also raises new challenges for QoS.Consumers have certain expectations of the quality of their communication service, primarily based on their past experience of the well established PSTN voice quality. However the increasing amount of choice of products available to them through NGN may well alter their perceptions of and satisfaction with the overall QoS provided.

There are a number of QoS-related aspects that need to be addressed as NGN is deployed. These include:

- Service disruption during the migration from PSTN to NGN
- Management of end-to-end voice quality of service
- Access to emergency services and emergency call location
- Number portability
- Feasibility of alternative text relay services
- Differentiation of QoS
- Network integrity
- Network security

We have already explained many of the QoS issues arising because of the identifiers in NGN. In the above listed aspects one of the main issue to point out due to the identifiers is number portability. The central issue in number portability is how communication providers route calls and messages to numbers that have been ported. In order to route calls correctly, providers need to know the location of the destination number, based on a number range analysis. Due to the increased convergence between services that have traditionally been regarded as 'fixed' and 'mobile' services, and a rise in the number of services using VoIP, the process of locating the numbers depending on the services or operators is marginally reduced. Similarly the distinction between the geographic location of a fixed number and between fixed and mobile numbers is also being eroded and the deployment of NGN architectures will further erode the association of area number codes with a fixed location.

The rest of this section briefly surveys the key technologies used for address mapping across network boundaries. For each of them we also point out the reasons why we didn't choose them for our work.

2.1.2 Universal Personal Telecommunications (UPT)

The first approach in direction of personal numbers and subscriber mobility for telephony service was the UPT, as defined by ITU-T Recommendations F.85x [11] [12] [14]. The identifier used within UPT is an E.164 number.

UPT allows a subscriber to use any terminal, if the network supports it, to originate calls from this terminal on the subscribers account and to receive calls directed to the UPT number at this terminal.

Although the UPT service is defined to be technology independent, it was originally designed as an intelligent network based service on circuit-switched networks and for voice communication. Currently a development is in progress to extend the access to this service to IP based networks, using IP technology. This would allow the subscribers to place and receive voice calls from any IP based terminal, provided that the appropriate client software is installed.

Even though UPT is not restricted to voice services, in practice it is centered around voice and voice related communications. Another problem is that it requires a certain amount of infrastructure, prerequisites and inter-carrier agreements before it can be used.

2.1.3 Universal Communication Identifier (UCI)

UCI has been developed as a system which aims to bind many services to a single identity and manage the communications with a personal user manager. The UCI would consist of a friendly alpha part, a numeric part and an additional information which will not be seen by the communicants:

Sandoche Balakrichenan [33160000004] <a6;f1;d234;k78>

The numeric part is usually based on an E.164 number. The UCI is designed in such a way that it is an unique identifier and can be used for all type of communication networks and services. It need not change when the UCI owner uses new communication services or change the provider. It facilitates end users to allow them to identify the source of the incoming connection or to confirm the identity of the remote end-point to whom a connection has or will be established.

To achieve its full potential, an identifier would need to operate within an advanced communication architecture capable of supporting the concept of Personal User Agents (PUA), which manage all aspects of user communication. The PUA is a software entity located in an overlay network and associated with a specific UCI. It would contain a profile, updated by UCI user, identifying a range of available communications media and rules by which each could be used.

The problem with UCI is that it requires additional infrastructure and considerable additional standardization work, and, worst of all, modifications in existing standards

2.1.4 Telephone Routing over IP (TRIP)[82][83]

Where a telephone number does not have a SIP resource associated with it, the IP network routes a call to a telephone gateway, which connects to the PSTN. In an interconnect environment with many peering relationships between service providers, resources in the IP network need to be able to discover which telephone numbers are associated with which gateways.

The IP Telephony (IPTEL) working group has defined TRIP as a policy driven inter-administrative domain protocol for advertising the reachability of telephony destinations between location servers, and for advertising attributes of the routes to those destinations. TRIP is designed to allow service providers to exchange routing information in order to avoid the over-provisioning or duplication of gateways. It uses established Internet protocols such as Border Gateway Protocol (BGP), Open Shortest Path First (OSPF) and Server Cache Synchronization Protocol (SCSP) to exchange routing information between different providers, so that policies can be applied to the resulting data to create a forwarding information base.

While TRIP is used to carry routes to the destination on PSTN, a method for locating terminals on the IP network is still required. This problem is simpler than the gateway location problem, since the amount of information describing a terminal on the IP network is less than the information about a gateway on the PSTN. Even though TRIP can be used for this purpose it is not worth the complexities taken. A simple DNS system could be used to locate terminals in the IP network.

2.2 IP Telephony Signaling

In this section we initially point out the network capabilities to support the IP telephony interworking between PSTN and IP. Then we survey the literature for the two main standards of IP telephony signalling. A very important benefit of IP telephony is the integration of voice and data applications. Examples of such applications are Voice mail, teleconferencing, automatic and intelligent call distribution etc.

In order for the IP telephony to be accepted by PSTN users, it should meet certain requirements:

- The delay and packet loss of the IP network must meet the requirements of telephony application.
- The ease of operation and functionality offered to the end user must be at least at the same level as in PSTN.

In order to meet the requirements of providing a simple system to the user as in PSTN, the IP telephony architecture has to provide a *signaling infrastructure* that offers the same capabilities and features as the Signaling System 7 architecture in PSTN. The signaling infrastructure must provide [27]:

- the functionality required to set up, manage, and tear down calls and connections.
- be scalable to support a very large number of registered endpoints (in the order of billions worldwide), and a very large number of simultaneous calls (in the order of millions worldwide).
- support network management features for policy control, accounting, billing, etc.
- provide a mechanism to communicate and set up the Quality of Service requested by the end points;
- be extensible to help with adding new features easily.
- support interoperability among different vendors implementations, among different versions of the signaling protocol, and with different signaling protocols.

Two standards compete for IP Telephony signaling. The older and currently more widely accepted standard is the ITU-T recommendation H.323 [13], which defines a multimedia communications system over packet-switched networks, including IP networks. The other standard, Session Initiation Protocol (SIP)[59], comes from the IETF MMUSIC working group.

2.2.1 H.323

The most widely deployed use of H.323 is "VoIP" followed by "Video conferencing", both of which are described in the H.323 specifications [13]. H.323 consists of several components: terminals, gateways, gatekeepers and Multiple Control Units (MCUs). These components could be implemented individually or incorporated as a group within a single product.

H.323 has been described as an umbrella standard, under which a number of other protocols, supporting call setup and disconnect, audio encoding/decoding, video encoding/decoding, fit under. These protocols include the ITU-T H.225, H.245 protocols, plus the IETF's Real-Time Transport Protocol (RTP), and others.

For addressing structures H.323 has flexible addressing mechanisms, including URIs, email addresses and E.164 numbers. H.323 defines an interface between the endpoint and gatekeeper for address resolution using Automatic Request(ARQ) or Location Request (LRQ). The H.323 gatekeeper may use any number of protocols to discover the destination address of the callee. The endpoint does not have to be concerned with the mechanics of this process, and the processing requirements for address resolution placed on the gatekeeper by H.323 are for just a single message exchange.

Disadvantages of H.323 protocol

its lack of scalability. H.323 is mostly used in small LANs. When extending to world-wide IP networks, H.323 has many disadvantages:

- H.323 has no support for loop detection, which can cause network overload
- H.323 uses gatekeepers, which are devices used for handling call states and redirecting calls to aliases. As every call is carried out statefully, the gatekeepers must keep a call state during the entire call. This of course makes the gatekeepers a major bottleneck in the system.
- There is also a need for a central point when performing multi-user calls, which means that someone must provide this central point, and that this machine must be dimensioned for the size of the call. Establishing a connection using H.323 takes about three times the data and turnarounds compared to when using SIP.

2.2.2 SIP

SIP was designed to setup a ”session” between two points and to be a modular, flexible component of the Internet architecture. It has a loose concept of a call (that being a ”session” with media streams), has no support for multimedia conferencing (Here we mean it does not have tightly-coupled conferencing, where explicit membership and conference control mechanisms are applied), and the integration of sometimes disparate standards is largely left up to each vendor.

SIP, defined in [48] (with various extensions, such as extension for Instant Messaging, for IP Multimedia Subsystem (IMS) etc.), handles creation, modification and termination of various media stream sessions over an IP network. It is a signaling protocol used to create, manage and terminate sessions in an IP based network. A session could be a simple two-way telephone call or it could be a collaborative multi-media conference session (Here we mean SIP has loosely-coupled conferencing capability wherein sessions are established without explicit membership and conference control mechanisms) . This makes possible to implement services like voice-enriched e-commerce, web page click-to-dial or Instant Messaging with buddy lists in an IP based environment. SIP is limited to only the setup, modification and termination of sessions. It serves four major purposes:

- SIP allows for the establishment of user location (i.e. translating from a user's name to their current network address).
- SIP provides for feature negotiation so that all of the participants in a session can agree on the features to be supported among them.
- SIP is a mechanism for call management - for example adding, dropping, or transferring participants.

- SIP allows for changing features of a session while it is in progress.

Even though SIP has been in use for some time, it comes up with a vast number of interoperability problems. While SIP has been successfully deployed in some environments, those are generally "closed" environments where the means of interoperability has been PSTN gateways.

SIP only understands URI-style addresses. This works fine for SIP-SIP devices, but causes some confusion when trying to translated various dialed digits. A "+" sign is inserted in the SIP URI (e.g., "sip:+160764542@int-evry.fr") in order to indicate that the number is in E.164 format, versus a user ID that might be numeric.

SIP has support for overlapped signaling defined in RFC 3578, though additional digit received requires transmission of three messages on the wire (a new INVITE, a 484 response to indicate that the address is incomplete, and an ACK). While SIP has no address-resolution protocol, a SIP user agent may route its INVITE message through a proxy or redirect server in order to resolve addresses. The SIP proxy may use various protocols to discover the destination address of the callee, including TRIP, ENUM, and/or DNS. The endpoint does not have to be concerned with the mechanics of this process. Unfortunately, the processing requirements placed on the SIP proxy are higher than with H.323 because at least 3 message exchanges must take place between the SIP device, SIP proxy, and the next hop. The interesting observation for us here is a SIP user agent may perform its own address resolution using ENUM and/or DNS and then place a direct call to the resolved address or through a proxy.

Disadvantages of SIP

The main disadvantage of SIP is that it has no means to detect network failures. The processing requirements placed on the SIP proxy are higher than with H.323, which requires more message exchanges to pinpoint the position. Another main problem with SIP is that all the new versions are not backward compatible, which causes operational problems

2.3 Electronic Number Mapping (ENUM) Background

In the previous sections 2.1 and 2.2 we briefly explained about the different technologies that are used in IP-PSTN addressing and also some of their disadvantages. In the current and forthcoming sections (2.4 and 2.5) we explain about ENUM and the other protocols and concepts used by ENUM. We identified ENUM for our work because it is the most simple and easy method to have interoperability between PSTN and IP. It also solve the multiple

PCI problem. Most of all it uses the existing DNS infrastructure which has been successfully used for a while and is proven method of address mapping.

ENUM [32][31] is a DNS based service that maps a standard telephone number to a list of contact URIs. It enables PSTN as well as mobile users to register special ENUM domains for their numbers. Using DNS, other telephone users can reach these numbers over Internet which is almost free. The ENUM system is operational in many countries like Austria, Germany etc.. Many other countries are also following suit [5]

ENUM is a result of work by IETF(Internet Engineering Task Force) Telephone Number Mapping Working Group. ENUM is not a single entity. It is a convention for the use of a specific set of existing protocols and Domain Name System (DNS) database to translate standard telephone numbers to information. For this conversion the following existing protocols are used:

- E.164 numbers and the E.164 arpa domain
- The DNS protocol
- URI'S

In order to do modifications in the existing ENUM architecture and improve performance, it is important to first understand the components that is used by ENUM. We will give a brief overview of the protocols and the database used by ENUM in the following subsections.

2.3.1 E.164 Telephone Numbers

E.164 is an international numbering plan for public telephone systems. The ITU-T(International Telecommunication Union's Telecommunication) Standardization Sector makes recommendations for this numbering plan. The discussion of the E.164 ITU-T numbering plan is intended to provide a background for understanding the relationship between E.164 numbering with ENUM. The number structure and functionality used for international public telecommunication systems can be classified into three categories:

1. ITU-T numbering for Geographic Areas
2. ITU-T numbering for Global Services and
3. ITU-T numbering for networks

ITU-T Numbering Structure for Geographical Areas

Under this category, number structure are assigned in such a way that each number contains a Country Code (CC), a National Destination Code (NDC), and a Subscriber

Number (SN). For example in France a PSTN number according to E.164 numbering plan is as follows 33-1-60760000 where 33 is the CC, 1 is the NDC for the Paris areas and 60760000 is the SN. There can be up to 15 digits in an E.164 number. The country code consists of 1 to 3 digits. The size of the telephone number together with the NDC and SN depends on the size of the country code. For e.g. in case of France, where the size of CC is '2' the maximum size of the telephone number including NDC and SN can be up to '13'.

ITU-T Numbering Structure for Global Services

Services such as International toll free numbers, International premium rate services etc. are identified as Global Services. The subscribers here are identified uniquely only at the International level. It consists of a three digit country code. The country code is always in the 8XX and 9XX range.

ITU-T Numbering Structure for Networks

This networking structure is used to identify specific network operators or service providers and also used to identify global mobile satellite systems. This numbering structure uniquely identifies a subscriber within a network internationally. It consists of a three digit country code, one to four digit network identification code and the remaining digits for the SN. The country code is always in the 8XX range.

2.3.2 E164.arpa domain

arpa is an abbreviation for the *addressing* and *routing parameter area* domain [42] and is designed to be used exclusively for Internet-Infrastructure purposes. E164.arpa is a SLD(Second Level Domain) that was designed specifically for providing ENUM services. The domain e164.arpa. is being populated in order to provide the infrastructure in DNS for storage of e.164 numbers. In order to facilitate distributing operations this domain is further divided into sub domains.

E.164 telephone numbers will be delegated from this domain. For e.g. for France with country code +33 will become the domain 3.3.e164.arpa. The importance of this model is that it uses the same branching structure of the existing PSTN model.

2.3.3 Uniform Resource Identifiers [19]

A URI is a sequence of characters which make it possible to identify resources such as a document, image, database file, email address or other resource or service presenting a common format. The most common URI is the URL(Uniform Resource Locator) which is

used to locate resources using the World Wide Web (WWW). An URI can point to various type of resources such as mail address (mail to: user@int-evry.fr), a web page (www.int-evry.fr) or a SIP address (SIP: user@int-evry.fr).
Some examples of URI are:
http://www.int-evry.fr
This URI uses a http scheme for Hypertext Transfer Protocol Services
mailto:sandoche.balakrichenan@int-evry.fr
This URI uses a mailto scheme for electronic mail addresses
Please refer to [19] for complete URI specifications.

To summarize, ENUM uses DNS database to map e.164 numbers to the URI corresponding to it.

2.3.4 Domain Name System

Since IP address are unfriendly and could not be remembered easily DNS(Domain Name System) is used, whose primarily functionality is *name resolution*. DNS is a distributed look up service (database) that is used to translate between domain names and IP addresses. DNS is a well studied topic. It already exists and is global. It is the most efficient, open and scalable system for name resolution. So the research community has decided to use DNS for ENUM to resolve the information connected with the E.164 numbers. Apart from the advantages of DNS that is already discussed, it also provides a low cost solution for ENUM.

2.4 Domain Name System and ENUM

In the previous section 2.3 we have explained that ENUM relies on DNS to look up the information corresponding to E.164 numbers. In this section we explain key concepts of DNS which will be used by ENUM.

DNS is nearly as old as the Internet itself and is still its corner stone. There can be no communication on the web like email or web page resolution without DNS. In the beginning of the Internet, a simple *host.txt* file located on a single computer was responsible for the translation between IP addresses (such as 157.159.100.44) and domain name (Such as "www-rst.int-evry.fr").As Internet grew Paul Mockapetris designed a more suitable distributed database ([69]and [70]) which is the DNS.

A classical DNS usage pattern to resolve the IP address of the FQDN "www-rst.int-evry.fr" is as follows (See Fig: 2.1):

A Host who wishes to access the web page of "www-rst.int-evry.fr" uses a client application such as web browsers or mail clients in the Host computer. Such applications send a request to the local DNS resolver in the local Operating System. The DNS resolver will invariably have a cache containing recent DNS look ups. If the cache can provide the answer to the request, the resolver will return the value in the cache to the application that made the request. If the cache does not contain the answer, the resolver will send the request to one or more designated DNS servers.

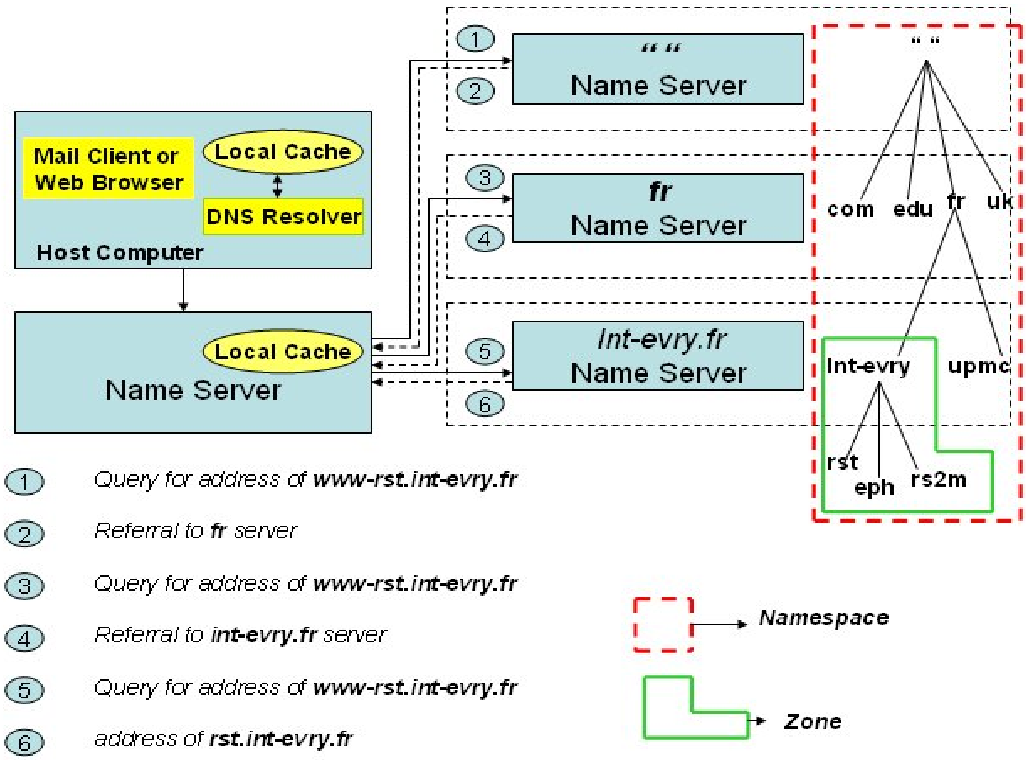

" "
Name Server
fr
Name Server
Int-evry.fr
Name Server
Mail Client or Web Browser
Local Cache
DNS Resolver
Host Computer
Local Cache
Name Server
1
2
3
4
5
6
" "
com
edu
fr
uk
Int-evry
upmc
rst
eph
rs2m
1 Query for address of www-rst.int-evry.fr
2 Referral to fr server
3 Query for address of www-rst.int-evry.fr
4 Referral to int-evry.fr server
5 Query for address of www-rst.int-evry.fr
6 address of rst.int-evry.fr
Namespace
Zone

Figure 2.1. Resolution of "www-rst.int-evry.fr" on the Internet

In the case assuming that the answer is not found in the local cache of the Hosts computer, the resolver (client) sends a UDP recursive query to one of its configured local nameserver. In case of home users mostly it will be their Internet Service Provider(ISP). The local nameserver in turn, looks for the answer in its local cache and, if an appropriate record is found, returns the cached address to the client.

On the contrary if the answer is not found in the cache of the local nameserver, then the burden of finding the response for the resolvers query becomes the responsibility of the local name servers. The local nameserver queries the root nameserver for the address of *www-rst.int-evry.fr*. There are 13 root servers (from a.root_servers.net to m.root_servers.net). The root server process the query and even though it does not know the address of *www-rst.int-evry.fr*, it knows that the information should be under the control of the "Top Level Domain (TLD)" *fr* servers. In this case one of the root server will refer the query to the *fr* nameservers. The local nameserver asks the *fr* nameserver the same question, and is referred to the *int-evry.fr* nameservers. Finally, the local nameserver asks *int-evry.fr* nameserver for the queried domain name address and gets the answer.

We explain the key DNS concepts which are involved in this name resolution process in the following subsections.

2.4.1 DNS Namespace

As we have already mentioned, DNS is a distributed database. Each domain name is essentially just a path, in a large inverted tree called the *domain namespace* (The red dotted rectangle covering the inverted tree path in figure 2.1) The Name space consists of all combination of domain names and TLDs (fr,edu etc..)existing below the root *zone*.

2.4.2 Zone

A Zone is any domain name that has been delegated by an ancestor zone. A Zone is a point of delegation in the DNS tree. It contains all descendant domain names from a certain point downwards that has been not delegated (because those delegated other zones are authoritative). A zone is therefore a discretely managed portion of the total domain name space (Zone of "int-evry.fr" can be seen in the figure 2.1) within a single domain and is represented by the data stored on a particular nameserver.

2.4.3 Delegation

The process of separating a descendant of a zone into a separate zone is called *delegation*. The delegation is accomplished with nameserver Records (a type of a Resource Record). The delegation system in the DNS enables hosts to control given chunks of the database and the whole database is reachable with a client-server mechanism.

2.4.4 Nameserver

A *nameserver* is software that runs on a host. A host is any machine on any network. This software stores information about the domain *namespace*. The DNS software implementation known as Berkeley Internet Domain Name (BIND) is the most commonly used domain nameserver on the Internet.

Nameservers generally have complete information about some part of the domain namespace. There are two types of nameservers, Authoritative and Caching.

The Authoritative nameserver (AS) contains an entire copy of the zone that is derived from the local configuration data, possibly with the help of another authoritative nameserver for the zone. A server can be authoritative for one zone but not authoritative for another. The Master (primary) server is an AS that gets its zone data from the local configuration, not from an outside source. This is where all changes to the zones contents are made. The DNS protocol provides an automatic mechanism for propagating the contents of a zone to slave (secondary) servers, which will be queried upon in case of the failure of primary servers.

The Caching nameservers are usually local nameserver. It initially looks up a name within the current zone and ends up asking one of the zones nameserver for the answer and keeps the result of all the name resolutions in its cache for a Time To Live(TTL). The TTL on a resource record(see 2.4.5) is the length of the time for which any nameserver can cache that record.

2.4.5 Resource Records

Resource Record (RR) is a unit of data in the DNS. The data associated with domain names are contained in RRs. It defines some attribute for a domain name such as IP address, a string of text or a mail route. A nameserver RR declares that a given zone is served by a given nameserver. Every nameserver record is either a *delegation record* or an *authority record.* If the name of the nameserver record is the name of the zone it appears in, it is an *authority record.* If the name of the nameserver record is that of a descendant zone, then it is a *delegation record.*

Another type of RR is the *Naming Authority Pointer* (NAPTR) RR. It specifies a regular expression-based rewrite rule that, when applied to an existing string, will produce a new domain label or URI. NAPTR is used by ENUM to provide a URI and look up what services are available for the related E.164 number. NAPTR RR [66] is an entry in the DNS containing rules for transcribing inquiries.

The Start of Authority (SOA) RR is the first record in every properly configured zone. The SOA record contains information about the zone in a string of fields. It tells the server to be authoritative for the zone. The result is returned to the application making the inquiry.

2.4.6 Resolver

Resolvers are clients that access nameservers. Programs running on a host that needs information about the domain namespace use the resolver. In BIND the resolver is a set of subroutines that is linked to programs such as ssh or ftp. The resolver relies almost entirely on the nameservers it queries to locate the RRs. The resolver handles:

- Querying a nameserver.
- Interpreting responses(which may be RRs or an error).
- Returning the information to the programs that requested it.

2.5 Technical Aspects of ENUM

In section 2.3 we gave an overview about the components used by ENUM. This section goes more technically explaining how the information in the DNS database is retrieved and interpreted by the client software of the user querying the database. Also we give a brief explanation about the ENUM tiered architecture in this section.

2.5.1 Dynamic Delegation Discovery System (DDDS)

DDDS is introduced here because as explained in RFC [32], ENUM is a DDDS application using DNS as a database. DDDS is defined by four RFCs [62],[64],[63] and [65].

RFC[62] defines DDDS as follows:

The DDDS is used to implement a binding of strings to data, in order to support dynamically configured delegation systems. The DDDS functions by mapping some unique string to data stored within a DDDS database by iteratively applying string transformation rules until a terminal condition is reached.

The DDDS algorithm is defined in RFC [64]. The document defines the following DDDS concepts:

- The basic DDDS vocabulary
- The generic algorithm used by all applications
- The requirements on applications using the algorithm
- The requirements on databases that store the DDDS rules

RFC [64] explains the specifications of DDDS algorithm. But it is not complete without specifying how and why the algorithm is used by an application. The ENUM system is one of these applications that use the DDDS algorithm. According to the last point in the DDDS concepts that have been mentioned above, applications require databases to store their rules. These databases are called DDDS databases. It can be the DNS database and the details of using DNS NAPTR RR are mentioned in RFC 3403 [63] Any DDDS application must use some type of database. RFC [63] defines the following:

- The generic specification of how the database works
- The formats of the keys (e.g. domain names in case of ENUM)
- The format of the rules (e.g. the NAPTR records in case of ENUM and DNS)
- The Key look up process (a standard DNS request)

- The insertion rule procedures (rules are inserted by adding new records to the appropriate DNS zone)
- The collision avoidance measures, in case that two applications use the same database

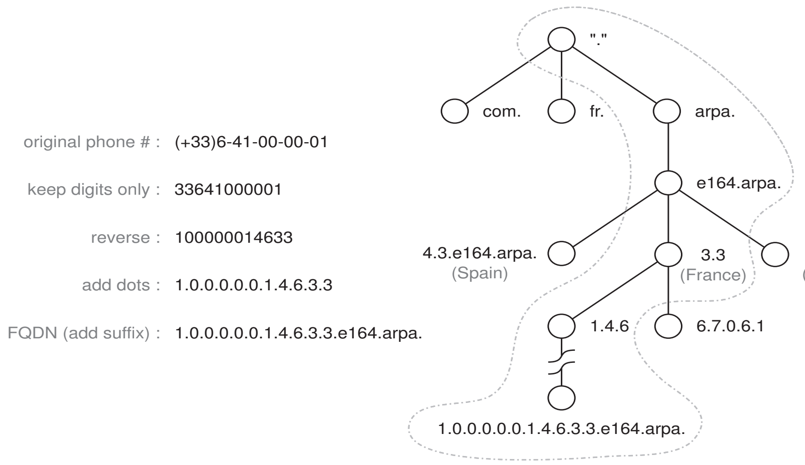
RFC [65]explains the application specification:

- The Application Unique String (AUS): the thing the delegation rules act on (e.g. an e.164 number)
- The first well known rule that says where the process starts, and the algorithm to create the unique key of a valid database.
- The list of valid database (e.g. DNS)
- The final expected output (e.g. an URI in case of ENUM)

2.5.2 ENUM DDDS Application

In subsection 2.5.1, we summarized the four RFCs defining DDDS. This section explains how an ENUM application uses DDDS algorithm to query the DNS database (Refer to the last paragraph of subsection 2.5.1 which explains the rules for an application to use DDDS algorithm):

The first well known rule is to create the unique string of a valid database. In the ENUM case it is the public telephone number i.e. the e.164 number. The e.164 should be converted to an AUS. AUS is a fully qualified e.164 number minus any non digit characters except for the + character which appears at the beginning of the number. For example an e.164



original phone # : (+33)6-41-00-00-01
keep digits only : 33641000001
reverse : 100000014633
add dots : 1.0.0.0.0.0.1.4.6.3.3
FQDN (add suffix) : 1.0.0.0.0.0.1.4.6.3.3.e164.arpa.
"."
com.
fr.
arpa.
e164.arpa.
4.3.e164.arpa.
(Spain)
3.3
(France)
4.4
(UK)
1.4.6
6.7.0.6.1
1.0.0.0.0.0.1.4.6.3.3.e164.arpa.

Figure 2.2. Telephone Number Conversion to FQDN in ENUM

number which comes as ”+ (33)6-41-00-00-01” is converted to an AUS as ”+33641000001”

In order to convert this string to an unique key for the valid database (DNS), the string is converted to a domain name according to the following algorithm (Fig: 2.2)

1. Remove all characters with the exception of digits. For e.g the first well known rule has produced this string "+33641000001". This step of the algorithm should simply remove the leading "+" producing "33641000001".
2. Put dots between each digits which results in "3.3.6.4.1.0.0.0.0.0.1"
3. Reverse the order of the digits 1.0.0.0.0.0.1.4.6.3.3
4. Append the string ".e164.arpa." to the end. 1.0.0.0.0.0.1.4.6.3.3.e164.arpa. which results in a Fully Qualified Domain Name (FQDN)

The domain name is a unique key to request the rules from the DNS database. These rules are stored in the NAPTR records. *The final expected output* in this case, the application ENUM turns phone numbers into information, so the expected output is a valid URI pointing to a service such as SIP or email.

2.5.3 ENUM Tiered Architecture

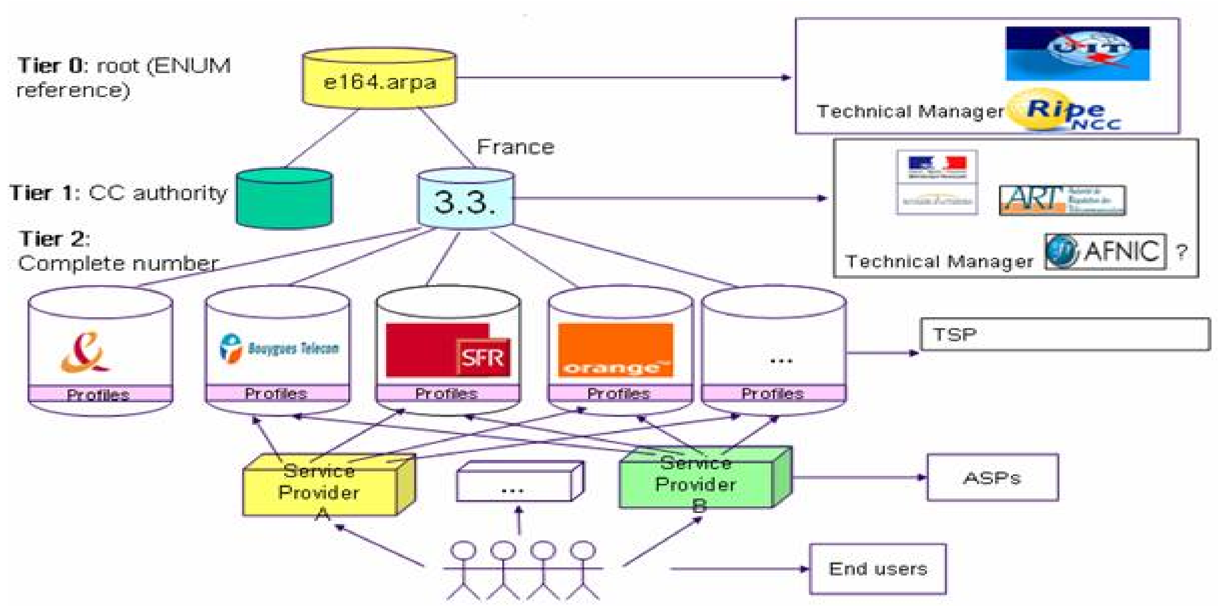


Figure 2.3. French ENUM Functional Architecture {*Picture Courtesy:Numerobis Project*}

To align the assignment path of E.164 numbers with the delegation procedures in DNS, the ENUM tiered architecture has been created(see Fig:2.3). The ENUM architecture is distributed and multiple tiers are responsible for different parts of the DNS tree.

In this tree *Tier-0* corresponds to the base of the internet domain space that is designated for ENUM i.e. e164.arpa. This will point to the registry that is the Authoritative NS for that country code or a portion of a country code. Tier-0 is administered by ITU-TSB(Telecommunication Standardization Bureau) and the operational responsibility is handled by RIPE-NCC(Réseaux IP Européens Network Coordination Centre). Tier-0 has NS records for the Tier-1 NS.

The *Tier-1* is a level below Tier-0 and corresponds to the E.164 CC, i.e. [CC].e164.arpa. (For France it is 3.3.e164.arpa.) . It is managed nationally. Tier-1 registries will manage name servers whose entries point to Service Registrar for a Telephone number. Records at this level contain pointers to the ENUM Tier-2 for a full e.164 number.

The next level *Tier-2* is the entity that stores NAPTR records for each subscriber. Tier-2 is managed by a telecom operator as they rent chunk of numbers to give them to their clients (For example numbers from +33-1-60-76-00-00 to +33-1-60-76-99-99-99 are assigned to one operator and numbers from +33-2-60-76-00-00 to +33-2-60-76-99-99-99 are assigned to another operator). In France Tier2 is divided into two levels, the Tier2 chunk which handles delegation for chunk of numbers and Tier2 Number which contains the real information linked to a phone number (i.e. NAPTR RRs)

2.5.4 How ENUM works

Let's say user "X" internet telephone services are mapped to the E.164 address "+33-1-60760000 " (see Fig:2.4). When user "Y" wants to call "X", he just has to call X telephone number (i.e. the E.164 number). The telephone network routes the call to the Internet Gateway that is the nominated service agent for this E.164 number.

The internet gateway takes the call set up request with "X" number and first reverses the digits, then inserts a "." between each digit and finally appends "e164.arpa" at the end (Conversion of E.164 telephone numbers to FQDN is explained in section 2.5.2). The resultant DNS string is the Fully Qualified Domain Name (FQDN) 0.0.0.0.6.7.0.6.1.3.3.e164.arpa. This name is then passed as a query to the DNS, to retrieve all associated Naming Authority Pointer (NAPTR) DNS Resource Records (RRs). The URI RRs used by ENUM are NAPTR records (RFC 2915). Let's suppose user "X" has the following entries into the DNS.

IN NAPTR 100 10 "U" "E2U+sip""!^.*$!sip:user@sip.int-evry.fr!"

IN NAPTR 100 10 "U" "E2U+sip""!^.*$!mailto:user@int-evry.fr!"

Each NAPTR RR follows a specific format(Class, Type, Order, Preference, Flags, Service, Regexp, Replacement). Explanation of each field is given in table 2.1 The two DNS NAPTR entries has an order value of 100 and a preference of 10. Since both of the entries have the same preference the first entry is taken. The "U" flag indicates that the rule is terminal and that the specified URI is to be used. The service field specifies that the SIP protocol is to be used, in conjunction with the E.164 to URI (E2U) resolution service (RFC 2543). The operation of the regular expression produces the URI of the form sip:user@sip.int-evry.fr.

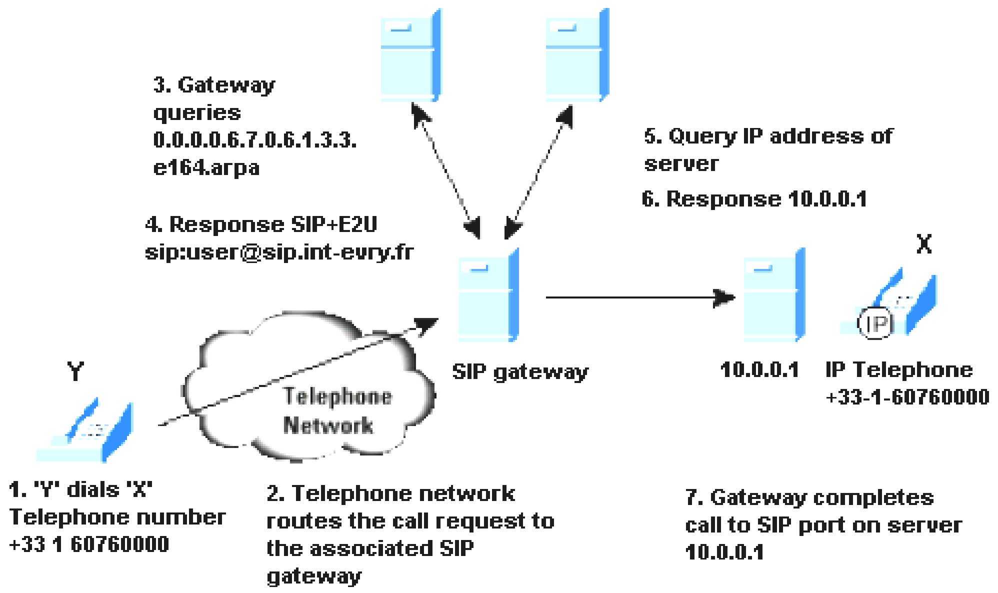

3. Gateway queries 0.0.0.0.6.7.0.6.1.3.3. e164.arpa
4. Response SIP+E2U sip:user@sip.int-evry.fr
5. Query IP address of server
6. Response 10.0.0.1
X
Y
Telephone Network
SIP gateway
10.0.0.1
IP Telephone +33-1-60760000
IP
1. 'Y' dials 'X' Telephone number +33 1 60760000
2. Telephone network routes the call request to the associated SIP gateway
7. Gateway completes call to SIP port on server 10.0.0.1

Figure 2.4. How ENUM works

Class	IN – Internet
Type	Name of the resource record
Order	The order for selecting rules.It is the rule with the lowest number that is applied.
Preference	if two rules have the same number, the preference value is used to refine the order further. The lowest preference value has priority. (The preference value corresponds to the MX record of a mail exchanger.)
Flags	This flag is used to check the transcription and stands for either a final transcription or the mapping of the inquiry with a URI as the output.
Service	The protocol of the service requested + the resolution service sip+E164 to URI.
Regexp	The regular expression for transcribing e inquiry is used.
Replacement	”.” means no replacement.

Table 2.1. NAPTR Format

For this call request, the gateway picks the E2U+sip service and performs the associated regular expression transform, using the original E.164 number and the regular expression. This produces the sip: URI. The gateway then uses the DNS a second time to translate the domain part of the URI, sip.int-evry.fr, into an IP address using a DNS A record. The gateway then opens up a session with UDP port 5060 on this SIP server to complete the call setup, requesting a voice session with the user ”X” on this server.

2.6 Summary

This chapter started with an overview of NGN, and motivated the need for ENUM to study addressing issues in NGN. We also discuss here the salient feature of other technologies related to this issue. We then give an explanation about ENUM, how it works and how the DNS system is related to ENUM. We then give a background explanation of the different concepts in the DNS system that will help the reader to understand better the forthcoming chapters.

We observe the current approaches towards NGN addressing do not comprehensively solve all the problems. The material described in this chapter set the stage for our work. The next chapter explains the methodology used to measure and model the empirical test bed used by us.

Chapter 3

Analysis

You know my methods, Watson

- Sherlock Holmes in 'The Crooked Man,' by Sir Arthur Conan Doyle

In this chapter, we explain the methodology used to measure and model the French ENUM model i.e used for the empirical measurements to perform the research reported in this thesis. We start by describing the need for studying the DNS performance for ENUM. Finally we conclude this chapter by the description of our research methodology used for the real measurements and the performance metrics.

Our goal is to build an autonomic simulation platform and to study the performance issues that affects ENUM request response time. To achieve this goal we use a combination of analysis, simulation and optimization as explained in chapter 1. This chapter describes the analysis phase.

3.1 Introduction

In the *analysis* phase, we begin with analyzing the importance of studying the DNS performance for ENUM. We analyze two cases. First the DNS performance measurements without taking ENUM considerations. Since ENUM has its own impact on the DNS server, in the second case the impact of ENUM on DNS performance is measured on a real French ENUM model. Using tools(existing as well as developed) we collect packet level traces of these measurements and analyze them in detail. This analysis helps us to develop a mathematical model based on the empirical measurements. Parameters are obtained from this mathematical model which are used in a developed simulation tool to have a simulation model mimicking the characteristics of the empirical model.

The gathered data and the functional characteristics of the empirical model serves as the starting point of the second phase *simulation*. Simulation is a powerful approach that allows us to explore the design space of parameters and algorithms more thoroughly than in a direct implementation. Building and verification of the simulation model is explained in chapter 4.

Once we obtain a simulation model which is validated, we move into the third phase *optimization*. This phase is explained in chapter 5.

3.2 Importance of studying DNS Performance for ENUM

The need for this section arises in concern with respect to the possible effects of ENUM on DNS performance. Performance in case of DNS, with respect to name resolution from IP address and vice versa, is a well researched topic. A brief overview of the literature in this case is given in section 3.2.1.

Despite Internet's popularity and wide spread use of addressing and naming schemes, the E.164 international number planning for telephone systems is still the most used addressing and naming scheme and the only one addressing scheme supported by most of the telephonic and other communication devices. To inter operate E.164 and Internet domains, ENUM is used. For this inter operation ENUM relies on DNS to distribute data about subscriber services.

Even though DNS is considered highly scalable as it is withstanding the pressures of the exponential growth of the Internet, there is concern whether it will be able to scale sufficiently to meet the resource requirements put forth by the E.164 recommendation. Introduction of ENUM service to the existing ones will make it more heavier. Each digit in the ENUM FQDN may represent a zone in DNS terms (For e.g. 33.e164.arpa. for France, 1.33.e164.arpa. for Paris, etc..)

For ENUM to be successful it is important that the global response time should be in correspondence with the response time in the telecom world. The response time in the telephone world is limited to milliseconds. But in a classical DNS request via the Internet, the response time is not short as the ones compared with classical telephony. Establishing a PSTN call is expected to take less than 200 milliseconds [3]. So this must be the upper bound of ENUM based look ups regardless of the amount of data the DNS server is offering or query load the DNS server is experiencing.

Personal information like contact address, email etc. are put on the public Internet in the ENUM case. At some point of time it is important to maintain the integrity and security of the ENUM zone files which will be possible by adding security options like DNSSEC. Adding security options can increase latency. So it is vital to study the DNS performance for ENUM which is a nascent research area according to our knowledge and try to improve it.

3.2.1 Challenges of DNS without ENUM considerations

Internet has dramatically grown and changed in the last few years. It is important that DNS must be highly scalable and offer good performance under high load to withstand the stress of the exponential growth of Internet. In this section we evaluate the research already done to improve DNS.

DNS is a well studied subject. One of the first major studies in the DNS performance was conducted by Danzig et al. in 1992 [79]. Danzig's study found that large number of implementation errors in DNS caused it to consume about twenty times more wide area network bandwidth than necessary. Danzig showed that by using the Berkeley BIND version 4.8.3 one could reduce the network bandwidth consumption by a factor of twenty and more. But the present DNS implementations and DNS usage patterns have changed. The Current BIND version is 9.4.1 which is the most used DNS server software has better nameserver and resolver algorithms than the version mentioned by Danzig.

Another important study [49] on DNS performance was conducted at the time when World Wide Web was the bulk of traffic for DNS. Their work is oriented towards studying whether effective caching mechanism could improve the DNS performance. They conclude that the scalability of DNS and its performance are not because of its hierarchical design of its name space or efficient caching mechanisms using reduced TTL, but because of the nameserver cache records which partition the name space and avoid overloading of any single server on the Internet.

Danzig[79] and Jaeyon[49] both look from a design point of view to improve DNS performance. Its true that an optimal DNS architecture will reduce DNS request response times. But the most vital factor that will affect DNS performance is the capacity of the servers to service the requests and perform the DNS look up. This is highly influenced by the load on the servers.

The high rate of DNS queries on the wide area internet is not caused just by real queries on the internet, but also by unwanted queries. About 35% of the DNS queries on a normal weekday are unwanted queries according to the work done by Brownlee [71]. These unwanted queries can be caused due to the following reasons:

- DNS traffic is on UDP. The reason UDP is used instead of TCP for DNS is because UDP packets travel much faster than TCP packets. Data transfer through TCP is slow not because of its inherent weakness, but because it attempts to be reliable. TCP waits for acknowledgement for every packet that is sent. This process is time consuming. The drawback of UDP is that they don't always reach the destination. The DNS client sends the queries and waits for a timeout. If it does not receive a response for the sent query within the timeout it resends the query. [71] points out that reliability is achieved by the DNS client repeating unanswered queries up to 12 times each separated by an exponential back off timeout interval.

- In certain cases the request itself contains the IP address. For e.g. a DNS clients query is "what is the IP address of 1.2.3.4?". Such a query is bogus since the query name is already an IP address.

- Some IP address space has been reserved for so-called "private addresses"[96], that is normally used in combination with NAT (Network Address Translation). An important requirement expressed in the RFC 1918 is that these "private addresses" must never leak into the global Internet, since they are not unique and there would be no meaningful way of handling them. Unfortunately, practice shows that both DNS queries and updates for such addresses occur in large quantities. As of April 2004 such bogus queries have made up to 1-3% of load at the F root server [93]

- Queries with invalid TLDs are the most common type of unwanted queries in DNS. They are the most problematic because the DNS does not provide any way for a cache to realise that the query is invalid. For e.g. if a user makes a query named "aa.something", the caching server contacts the root server and receives a name error. Subsequent queries result a negative cache hit. If another case comes with a query named "bb.something", the caching server again contacts a root server. There is no way the DNS understands that "something" is the problem rather than "aa" or "bb".

- Unwanted queries caused by denial of attacks is also a major force in flooding the DNS servers. In recent years a number of attacks have been made on the availability of DNS. For e.g. in October 2002 [73] a coordinated distributed DoS attack flooded all root servers with high volume network traffic. Although root servers were able to keep up with the traffic some root servers were unreachable for specific period of time or incurred an increase in the response time because of the network congestion caused by the attack.

Even though we have not made an exhaustive study of the way unwanted queries affecting the DNS performance, the above mentioned reasons are the most important causes.

It's not only the unwanted queries that affect the DNS performance. Legitimate queries are also the reasons for increasing load, not to mention the increase in number of web contents and number of users on the Internet.

Today more and more Web Pages are built with several different frames in a single page, where each frame sources data from a different server. When a browser accesses these complex web pages, it needs to resolve numerous server names using DNS for each page. This results in multiple DNS queries per page thus increasing the number of queries that has been done on the distant DNS server for a single page.

E-mail also needs DNS services. Once a mail server receives a mail, it checks the recipient list to identify where the mail has to be delivered. The recipient list contains the address which is of the form user@host.domain.. The user field will be a unique identifier for that particular domain. The host.domain field contains the hosts FQDN. DNS servers use the FQDN to locate the mail servers that service the respective domain. In the course of a mail being sent several checks such as Realtime Black List (RBL) checks, virus checks, spam checks are done, all of those which creates quite a high load on DNS queries. 75% of an ISP's DNS requests are used for mail exchanges and if each one of these triggers additional DNS lookups for all these checks, we can imagine the effect of this on caching nameservers which will markedly increase memory and CPU usage.

3.2.2 Core DNS extension affecting DNS performance

This section explains how sometimes extensions to DNS can also impact its performance. EDNS0(Extension Mechanisms for DNS) as defined in RFC 2671 is an extension of the DNS protocol to get rid of the size limit. It facilitates the transfer of data larger than normal DNS packet size which is 512 bytes. In this case one must take into account the networks transmission paths MTU (Maximum Transmission Unit). The reason is that the network path through which the over sized packet will traverse may contain routers which may not support UDP packets larger than 512 bytes. This may result in the packets being dropped and the packets sent again as in case of UDP. Over flooding the UDP MTU may result in performance degradations.

Another important extension of DNS is the DNS Security Extensions (DNSSEC) which adds security to the DNS. DNSSEC was designed to protect the Internet from certain attacks, such as DNS cache poisoning, provides:

- Authentication of origin of the DNS data,
- Data integrity,
- Authenticated denial of existence.

All answers in DNSSEC are digitally signed and can be checked. DNSSEC doesn't address the DDoS threats, on the contrary it is time consuming. Another fear is that thanks to the DNS NextSECure Resource Records (NSEC RR) for signing non existent data, it will be easy to reconstruct the DNS tree which will be 'more' public. One wonders that the DNS maybe suffering from so many new extensions (Dynamic Update,IXFR,IPV6 notification, Load Balancing etc..) which can reduce the scalable feature of DNS as reported in this work [24].

3.3 DNS Performance Measurements Without ENUM consideration

When we want to measure DNS performance, the first question always arise what aspect of DNS are we going to measure? The next question is what is the measurement technique that is going to be used for these measurements. The final question is what is the goal of our measurements. The goal of our measurements is to study and improve the global DNS response time in a ENUM scenario. This answers the first and the last question. In order to better understand and come with an answer to the second question about the measurement techniques to be used, we look into the DNS measurement literature (since not much research has been done on improving DNS performance with ENUM considerations) in the following subsections.

3.3.1 DNS Configurations

As pointed out in in Danzig[79] and Jaeyon[49], an optimal DNS architecture will reduce DNS request response times. A DNS administrator can improve the performance by appropriate DNS configurations adhering to the requirements of the DNS server he is administering. The important DNS configuration are the HW(Hardware) the OS(Operating System) and the DNS server SW(Software). Current DNS configurations(HW,OS and SW) can handle a huge load of queries(up to 60,000 queries per second). The HW and the OS are not going to be a big issue. But identifying the appropriate DNS server software is important. The DNS administrator might have to decide on what type of DNS server software to use and the compatibility of the software with the HW or the OS. To compare two configurations administrators must change only one thing at a time (e.g.:compare SWs on the same HW-OS set or compare HWs with the same SW-OS set) [15]. To choose appropriate DNS software implementation administrators have also to know which features they need.

There are number of DNS Server software packages. Out of a survey conducted [2] nearly 70% of the registered domains use BIND for DNS service. It should be noted that the survey did not include the root nameservers for any TLD. A study [54] on the DNS server software performance comparison was done. We look into the details of this study to understand the importance of the DNS software. The Hardware configurations for this study was

- Compaq Armada 4131T Laptop
- OS: FreeBSD 4.6.2-RELEASE
- CPU: Pentium 133
- RAM: 48MB real, 384MB virtual
- NICs: Asanté FriendlyNET AL1011 ”Prism2” 802.11b WiFi PC Card and Linksys EtherFast 10/100 PC Card (PCM100)
- HD: 10GB IBM Travelstar 20GN with 4200 RPM and 12ms avg. Seek

A brief overview of the DNS softwares used are given here:

- *BIND* (Berkeley Internet Name Domain) is a DNS protocol implementation and provides an open source distribution of major components of the DNS. The BIND DNS Server is used on the vast majority of name serving machines on the Internet, providing a robust and stable architecture on top of which an organization’s naming architecture can be built. The resolver library included in the BIND distribution provides the standard APIs (Application Programming Interfaces) for translation between domain names and Internet addresses and is intended to be linked with applications requiring name service. Two different versions (BIND 8.3.3-REL and BIND 9.2.2rc1) of BIND was used in this study[54]. The BIND 8 version has security holes which can be exploited by malicious users. The BIND 9 version uses extensions like DNSSEC with signed zones and TSIG (signed DNS requests) to fill the security holes. We do not go into the detailed explanation of Pros and Cons of different versions of BIND since it is not the objective here.

- The *djbdns* [4] program is a simple and security-aware DNS implementation created by Daniel J. Bernstein. "djbdns" is the name of the entire DNS server package with a collection of daemons and support utilities. Tinydns is the DNS content/ authoritative nameserver and dnscache is the caching-only recursive resolver is this package.

- *nsd 1.02b1* (for "nameserver daemon") is an open-source server program for the Domain Name System. It was developed by NLnet Labs of Amsterdam in cooperation with the RIPE NCC, from scratch as an authoritative nameserver (i.e., not implementing the recursive caching function by design). The intention of this development is to add variance to the "gene pool" of DNS implementations used by higher level name servers and thus increasing the resilience of DNS against software flaws or exploits.

The test was conducted on caching and authoritative servers of "root and "tv" zones. The tests were conducted on both cache and authoritative servers since both of them have different requirements (explained in subsection 2.4.4). The root zone was used because it is the well known small zone and "tv" was the largest zone ($\sim$ 20Mb) the author could get.

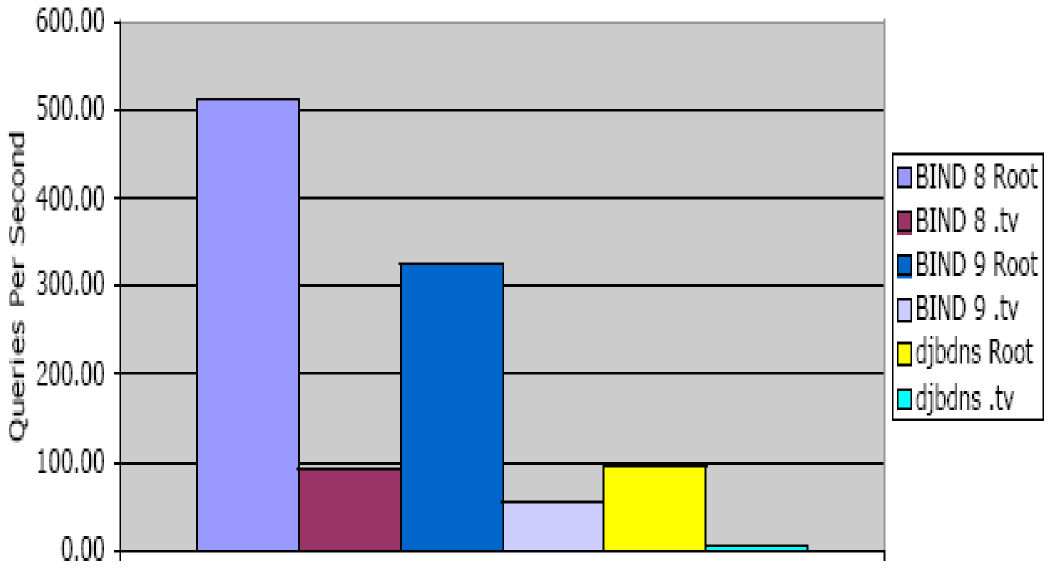


Figure 3.1. DNS Caching Server Performance (*Copyrights (2002) by Brad Knowles*)

Performance results for cache server are shown in figure 3.1 and for authoritative server in 3.2. For studying the DNS software which is suitable for cache server *nsd 1.02b1* was not used since it is built only for the authoritative server. Of the other three softwares used BIND 8 has better performance than the other two. In the authoritative server case 3.2 BIND 9 has the better performance.

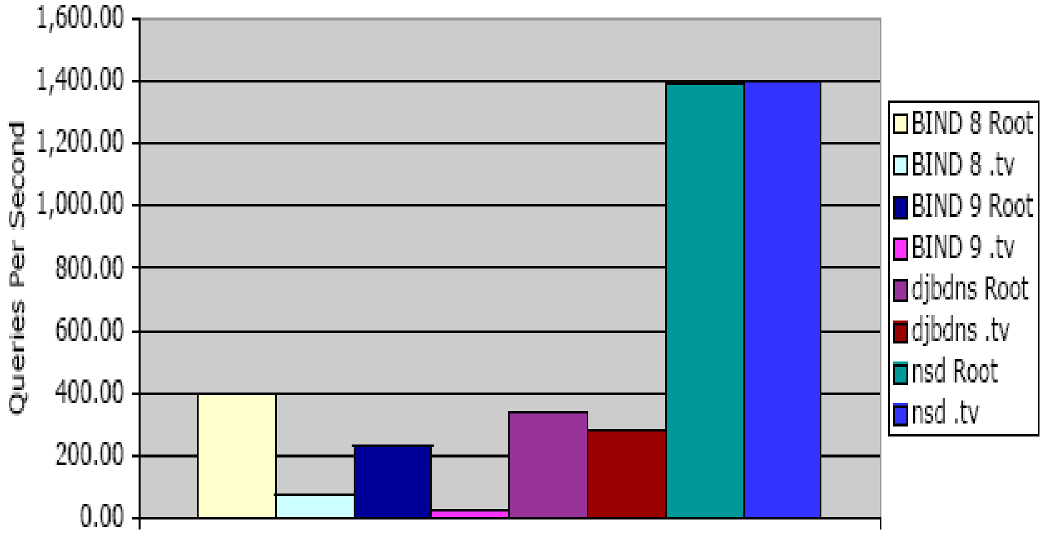


Figure 3.2. DNS Authoritative Server Performance (*Copyrights (2002) by Brad Knowles*)

There are several reasons one might use BIND 9 (in our experiments we have used BIND 9 version):

- BIND 9 is the most used DNS Software. One third of all Internet-connected DNS servers run BIND 9.
- As free software, BIND has the lowest possible initial cost.
- Has security extensions like DNSSEC with BIND 9 which can mitigate the usual DNS attacks

3.3.2 Measurement Metrics

From an end user point of view the most important metric that impacts him is the delay. In DNS terms it is the global *DNS Resolution Delay* (DRD). DRD can be defined as "the sum of all the local response times, the latencies generated by the link and network equipments, and the retransmission when a packet is lost (Timeout)."

Delay and Loss

In [4] Nigel Walker gives the parameters impacting the DRD by decreasing order of importance:

- Consecutive retransmissions with losses on the network (timeout) on the server level
- Number of hops caused by following factors:
  - Structure of the DNS tree (height)
  - Popularity of the name (Influence in the cache) and the associated Time To Live (TTL).
- The delay caused by following factors:
  - Round Trip Time (RTT) in the network
  - Service time of the server
- Speed of the light (Geographical Distance)

It is worth examining to identify the cause of the DRD increase, for improving the response time. Of the parameters affecting the DRD, the structure of the DNS tree, number of hops and popularity of the domain name are design considerations and are not the primary consideration factors for us here to measure when trying to improve the DRD. The delay is one of the important factors since delay can also induce loss (In DNS after a timeout the packet is considered lost)

Delay depends in part on the quality of the DNS caching server implementation, such as whether it can effectively use cached information instead of performing repeated queries, consistently select the fastest of multiple authoritative servers, and avoid waiting for unresponsive servers for a long time when alternative servers are available.

Unfortunately, delay also depends heavily on a number of factors unrelated to the caching server itself, such as the following:

- The amount of repetition among the queries. Queries that are answered from the cache will show a very small latency compared to ones where the server has to contact one or more authoritative servers to find the answer.
- The quality of the network connection.
- The percentage of queries that fail with a timeout due to unresponsive authoritative servers. These can take hundreds of times longer to complete than the average successful query: in a typical case, a successful query might take 0.1 seconds, but query that fails with a timeout might take 30 seconds[15]. Therefore, even a small percentage of failed queries can easily dominate the overall average delay.
- The client's overall query timeout (how long the client will keep trying before giving up on a query). In practice, the delay of a failed query will be determined by the shorter of the caching server's internal query timeout and the client's query timeout.

- The client's retransmission timeout. This is of particular interest when measuring the performance of BIND version 8 or older because of a peculiarity in its lookup algorithm. In some circumstances, BIND 8 will find that it is missing some nameserver addresses needed to answer the query and will initiate a lookup for the missing addresses, but will not automatically resume processing the query when they arrive. Instead, it relies on the client to retransmit the query.

Delay will not significantly depend on the speed of the machine the caching server is running on, since it is dominated by external network delays and timeouts rather than by processing time. It is of not much use in measuring the delay factor in a closed laboratory environment because of the external factors influencing the delay metric. As done in [54] the best way to measure the latency is to send a large number of queries similar to those of the real environment taken to study, with similar timeouts used in the real scenario, using the same network connection and calculate an average.

In theory, measuring delay is easy: A query is sent to a caching server, wait for a response, and see how long it took. This does not need any specialized measurement tools. A simple "dig" on the command line conveniently prints the "Total query time" as part of its output. If the query did not get a response with the stipulated timeout, then it is considered as loss.

Throughput

A likely explanation of the DRD would be that some nameservers are overloaded. In case of overload loss of packet occurs which can occur in the network path leading to these servers or in the queues inside the servers. The question is to find whether a server can withstand the load generated by the DNS packets. This issue can be calculated by the metric throughput.

The throughput of a caching (caching because authoritative servers usually do not get overloaded) DNS server is the number of DNS queries per second it is capable of handling. Throughput and delay can look related to each other, but its not. If a server is capable of answering 1000 queries per second it does not mean that each query will take 1/1000 seconds on average. This is because in the case the answer to the query is in the cache itself, the response time is much less, but its not the same when the server has to look up other authoritative servers to get the response for the query.

The factors affecting the throughput is the CPU time needed to process each query, network bandwidth and also on the capacity of the network interface on the server machine(on how many packets it can process per second). But it is possible to mitigate all these factors by increasing their capacity.

One way to measure the throughput is to have the cache servers actually query the authoritative servers on the Internet. This method can give realistic results. It should be made sure that the Internet connection has high bandwidth or the connection for the experiment is dedicated. Otherwise the Internet connection will become a bottleneck when high volume of packets are sent from the same client to the server.

Putting stress on a cache server, by having thousands of users querying at the same time is not a feasible solution. Stressing the Multiple clients can be simulated from one Host using the QueryPerf tool (this tool is explained in section 3.4.1)

To identify whether the maximum capacity of the server is achieved, command line options like ”top” in the unix command line can be used. If capacity is not close to 100% then more load is needed. Another option is to divide the number of queries resolved in one second by the percentage of the CPU used. For example with 10,000 queries resolved in a second and at 30% CPU the server is capable of resolving 33,333 queries per second.

3.4 DNS Performance in ENUM case

Deployment of ENUM as an overlay on the DNS system will increase the load on already stressed DNS servers. The ENUM applications specially VoIP ones need performance related to PSTN applications. ENUM puts a requirement on DNS that even with high load levels the expectation on performance is also high. In this section we will see the factors that can affect DNS performance during an ENUM resolution process.

Lets start with an example of a small enterprise to demonstrate the impact of ENUM in the DNS. A simple call within the enterprise needs at least one DNS query on the authoritative server infrastructure, translating the resolved name to appropriate IP address of the VoIP gateway for delivering the call internally. If redirections are required on the basis of NAPTR records additional lookups are required.

In case of an outbound call, each call must make at least one DNS query. In case of a fully delegated E.164 number, a DNS query is made for each level of delegation plus a minimum of one final query for the IP address. Additional queries are required if the query returns a NAPTR record with another domain within it. Clearly, somewhere between around 512 bytes per query (not counting the response, since response packets can be of larger size), and with multiple queries per call can account for a substantial amount of traffic, placing a burden not only on the recursive server, but on outbound bandwidth as well.

ENUM zones typically store large sets of RRs. An entry for one E.164 telephone number could contain 10-20 NAPTR records or more. An answer returning such an RRs will almost certainly exceed the capacity of a DNS response. It is a consideration point since it may often exceed the UDP size limit (512 bytes) specified in RFC 1035 which states the proportion of normal DNS queries exceeding this limit is very small.

The ENUM tree is most of the times deeper than the classical DNS tree, but the number of hops (which is one of the factors influencing DRD) can be kept low if TTL values are well chosen. The distribution of E.164 numbers may be different from domain name distribution which follows a Zipf Law [49]. In classical DNS, a small number of domain names are very popular and even small degree of cache sharing can take advantage of this. This method may not be effective in ENUM case.

Since ENUM has its own unique requirements of DNS, it is important to study how parameters such as timeouts, number of hops, popularity can be varied to reduce the DRD.

In this section, we explained the impact of ENUM on DNS. In the following subsections we explain the measurement methodology we made on a real French ENUM model(see Fig:2.3) and use the measurement results to understand ENUM and come up with promising solutions to improve its performance.

3.4.1 Measuring the Capacity of the Servers used

In subsection 3.3.2 we explained how *throughput* could be used as a measure to calculate the maximum load capacity of the DNS servers. Here we use the same procedure to identify the CPU utilization of the servers in the ENUM case.

This experiment was conducted to test the CPU usage of the servers used in the real French ENUM model (Fig: 2.3). To understand how much load these servers can withstand under stress, measurements were conducted on the following servers.

- ns1 and ns2 server in AFNIC i.e. at the Tier-1 server.
- is1 and is2 server in France Telecom i.e. at the Tier-2 Bloc and Tier-2 Number servers

In order to stress the servers with queries following the norms of ENUM, a file was created using the programming language 'perl' containing data in the ENUM addressing format. The file has random telephone addresses converted to ENUM query format, concatenated with 3.3.e164.arpa, which is the domain for metropolitan France. Each of the generated ENUM query has the following format:

```
8.5.5.4.6.7.0.6.1.3.3.e164.arpa.   NAPTR
1.8.7.4.6.7.0.6.1.3.3.e164.arpa.   NAPTR
```

The domain names thus generated confirm to the suffixes delegate by ART(Autorite de Regulation des Telecommunications). Such an effort is done to make the experimental approach in the real ENUM scenario. The file thus generated contained around 18,515,000 requests.

The file containing the above addresses forms as an entry for the configuration file of a tool called *Queryperf* which in turn uses these parameters to send as DNS requests to a local DNS nameserver. For years DNS administrators have been using the tool Queryperf given with the BIND DNS SW distribution [1], to measure the maximum capacity of their DNS server. This tool can also be used to simulate multiple clients.

Basically, the Queryperf program will maintain a fixed number of queries (default value is 20) running on a target DNS Server. Whenever a query takes more than the timeout value (default is 5 seconds) to complete, the query is considered lost and a new one is issued. At the end the program gathers statistics and, among them is the number of queries per second (qps) metric.

Queryperf was designed to test authoritative servers (servers where the records for the particular zone are stored) basically and not caching servers (servers where usually DNS queries are sent to identify the corresponding authoritative DNS servers). Authoritative servers respond with a much smaller latency than caching servers, and Queryperf's default settings reflect this. By default, Queryperf will only keep 20 queries simultaneously, in effect simulating 20 concurrent clients. This is enough to keep a typical authoritative server busy, but it is nowhere near enough to test the limits of a caching server. A command line option ”-q” can be used to specify a much larger number of outstanding queries, typically several hundred. Unfortunately, if the value of the ”-q” option is too large than the cache server can service, then it results in some queries getting lost.

Maximum throughput at the server can be achieved by running the command line option ”top” and monitoring the amount of CPU used by the caching server. It should be close to 100% - if it is not, we can either increase the ”-q” value, or run Queryperf on multiple client machines simultaneously and add up the results from all the Queryperf runs. In our case we added the ”-q” option.

The stress tests were conducted on two servers that were situated in AFNIC at the Tier-1 level. The configuration of these machines and the CPU utilization is depicted in table 3.1.

Server Name	ns1.numerobis.prd.fr	ns1.numerobis.prd.fr
Machine Type	DELL PowerEdge 2650	DELL PowerEdge 1750
CPU/RAM	2*2.8 GHz/2Gb	2*2.8 GHz/2Gb
Hard Disk Capacity	9Gb	10Gb
Operating system	Linux Debian 3.0	Linux Fedora Core 1
BIND version	BIND 9.2.3	BIND 9.2.3
Maximum Load	$\sim$ 19,000 Qps	$\sim$ 19,000 Qps
CPU utilization	72%	78%

Table 3.1. CPU Usage at Tier-1 level

The tests were also conducted on two servers that were situated in France Telecom at Tier-2 bloc and Tier-2 number level. The configuration of these machines and the CPU utilization is depicted in table 3.2. In these tests the packets that are lost are not taken into account.

3.4.2 Measurements on the French ENUM Model

From subsection 3.4.1 we have an idea on the capacity of the servers in the experimental platform that we we will be using for ENUM measurements.

Our goal is to create a simulation model and then experiment and test with various algorithms or techniques to find out ways to minimize the ENUM response time. The simulation model developed should reflect the features of the real ENUM set up. In the case of simulation models, when full measurement data is available it may be possible to use trace-driven simulation to observe the model under exactly the same conditions as the

Server Name	I-sf1.ft.numerobis.prd.fr	i-ds1.ft.numerobis.prd.fr
Machine Type	Sun Fire V210	DS10
CPU/RAM	2*1 GHz/2Gb	600 MHz/2Gb
Hard Disk Capacity	2*36Gb	36Gb
Operating system	Solaris 3.0	Tru64 Unix 5.1
BIND version	BIND 9.3.0s200221217	BIND 9.3.0s200221217
Maximum Load	$\sim$ 13,000 Qps	$\sim$ 8,000 Qps
CPU utilization	82%	89%

Table 3.2. CPU Usage at Tier-2 bloc and Tier-2 number level

real system. For this case we need to get parameters(response time and loss rate at each local nameserver) from an Empirical ENUM set up through real measurements which when input into the simulation model should have an output(global response time from simulation) corresponding to the real measurements (global response time from measurements). This is one of the methods to validate a simulation model.

In order to have the parameters three types of real measurements were made:

1. Measure and model the local DNS server performance (loss rate and response time of the DNS cache server at a given query rate and under a given DNS environment (Hardware, DNS software type and DNS database configuration))
2. Measure the global response time of the queries made on a French ENUM model.
3. Measure and model the two important metrics which impacts the performance of the IP links i.e. delay and loss.

Figure (3.3)represents the real set up and how the measurements were made on it.

We used the same method that we used in the subsection (3.4.1) to create different E.164 address of ENUM format which is shown below:

8.5.5.4.6.7.0.6.1.3.3.e164.arpa. NAPTR
1.8.7.4.6.7.0.6.1.3.3.e164.arpa. NAPTR

Alteration of Queryperf Tool

In the subsection (3.4.1) we explained about the Queryperf tool. There we explained that the "-q" option can be used with Queryperf tool to send more queries than the default option, which is 20. The original Queryperf tool stresses the target server with the defined range of queries and waits to see if there is any loss(i.e it does not receive the response for the query within the timeout period). If there is any loss it reduces the number of queries per second to make the target server process all the queries received. Our objective is to stress the server from a low to maximum range without considering loss. The maximum range

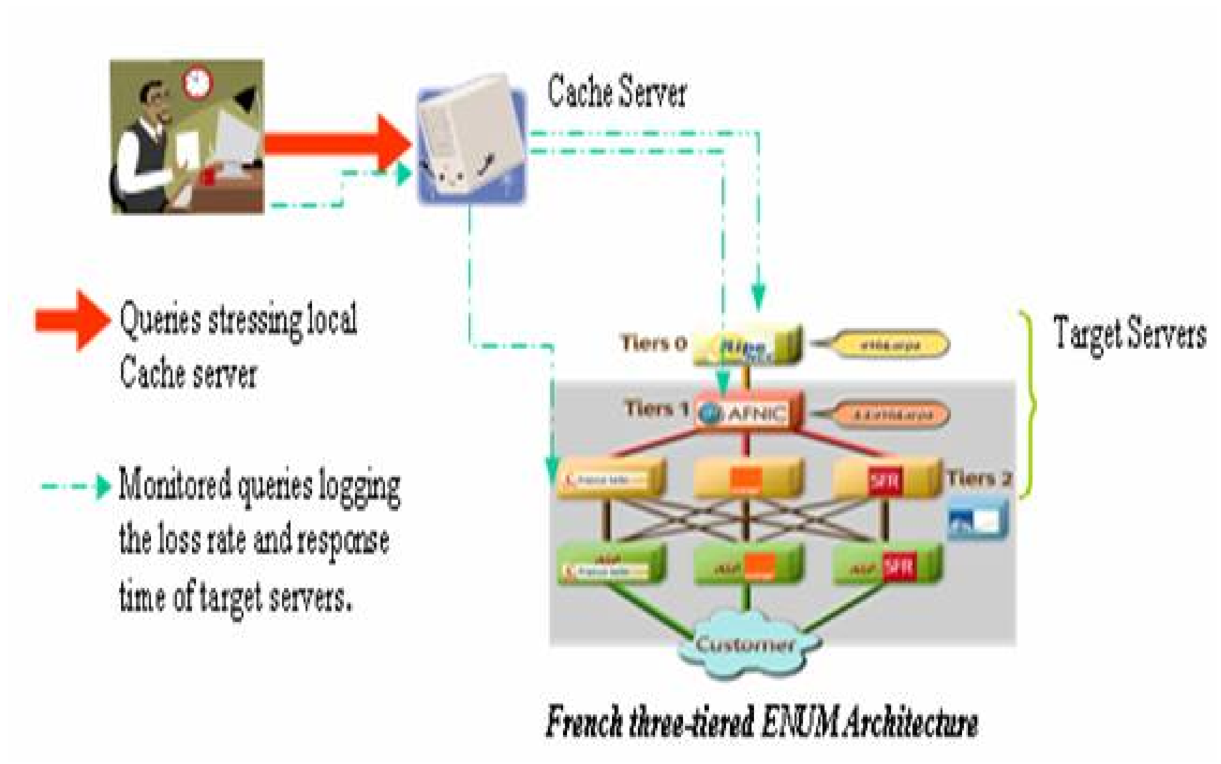

Cache Server
Queries stressing local Cache server
Monitored queries logging the loss rate and response time of target servers.
Tiers 0
Tiers 1
AFNIC
Tiers 2
SFR
Customer
Target Servers
French three-tiered ENUM Architecture

Figure 3.3. Empirical Measurement Set up

considered is more than three times the maximum range of queries used in the subsection 3.4.1 to measure the capacity of the servers.

Since Queryperf has a limited test range and we want to study the behavior of a local DNS server on a wider range, the source code of Queryperf was altered. The recompiled program enables us to choose the QPS load (any range) to stress the local DNS server with. This changed version of Queryperf no more checks for timeouts, its only purpose is to stress the local DNS server. This tool was used to load the server at different low (1000 - 20000 QPS), medium and high (50,000 - 80,000) QPS rates.

3.4.3 Measurements made on each Authoritative Server

The *response time* here indicates the latency between a single query and response from the particular server back to the local nameserver. *Loss* is observed once the load at the server passes the threshold the server can handle. This charge depends on different parameters (as explained in section 3.3.1) such as the hardware, the DNS software etc.

We used the utilities(the program for generating ENUM query and the altered Queryperf tool) to generate valid ENUM addresses and to stress the local DNS server. While the stress program was running and sending queries to the local DNS server at a given rate, we also logged the performance (loss rate and response time) of the target DNS servers by sending monitored queries.

These monitored queries are issued with a script using the "dig" command. The dig (domain information groper) is a flexible tool for interrogating DNS servers. Although dig is normally used with command-line arguments, it also has a batch mode of operation for reading lookup requests from a file. Here the requests are taken from our configuration file which we have explained before. The response for the query are captured by libpcap [6] software for analyzing.

Tcpdump was used for analyzing. Tcpdump is a network debugging tool that is used to intercept and display transmitted and received packets on a network. Filters can be used to restrict analysis to packets of interest. The command "tcpdump -w filename.log" helps to view the log file. The content of the log file helps to analyse the trace.

The measures have been done on the three DNS servers of the French ENUM model in order to resemble the French ENUM database:

1. Tier-1 server
2. Tier-2 Bloc and
3. Tier-2 Numero server

We reiterate the methodology we used for our measurements which consists of two stages:

1. To load the DNS server with maximum number of requests. The requests being sent from an altered Queryperf tool which simulates concurrent clients.
2. Another tool which is developed is used to send at regular intervals with requests to the target server and the response times for these requests are calculated.

For each qps value we measured the average response time, its Standard Deviation response time as well as the loss rate for the three target servers (Fig: 3.3).

The Results of these measurements are shown in the tables (3.3, 3.4 and 3.5)

In the beginning of the experiment when the local nameserver cache is empty all the queries are sent to the Tier-1 server to get information about the server that has the NAPTR RR for the query. The Tier-1 server has information about the Tier-2 nameserver provider which hosts the NAPTR associated with the query. The telecom regulatory authority ART maintains the Tier-1 server in France.

Table 3.1 shows that at around 19,000 queries per second, the CPU utilization rate of Tier-1 servers are around 75%. From the results of the measurements at Tier-1 server 3.3, we understand that loss rate increases rapidly around 20,000 queries per second. It is quite evident, since each lost packet results in re sending the query again and when the server is reaching around it maximum capacity more packets are dropped and as a result loss rate increases at a much higher rate. The response time also shows a sudden increase after around 20,000 queries per second.

Number Of Packets sent	Max. response time (ms)	Avg. Response time (ms)	Standard Deviation response time	Loss Rate(%)
0	0	0	0	0
1000	0	0.152	0.450	0
2002	2	0.155	0.421	0
3004	2	0.168	0.417	0
4002	2	0.18	0.442	0
5003	2	0.245	0.553	0
6004	2	0.255	0.574	0
7004	4	0.309	0.409	0
9004	4	0.316	0.464	1
10005	4	0.345	0.515	1
12003	4	0.732	0.857	1
14001	4	0.746	1.300	2
16001	4	0,782	1.272	2
18001	72	1,063	7.413	6
20000	232	1,409	16.702	14
30001	582	4,05	23.186	19
37648	623	10,54	29,94	26
50004	702	16,53	32,98	34
60004	714	26,97	33,597	42
70005	759	27,18	34,89	49

Table 3.3. Results of the measurements at Tier-1 server

Tier-2 is the entity who has contracted with the end user for telephone subscription. It either contains the information for the requested query or knows the server which contains the information for the query. The functional architecture of ENUM (see Fig: 2.3) is designed in such a way that Tier-2 is further classified in to Tier-2 Bloc and Tier-2 Number.

Tier-2 Bloc zone is managed by different TSPs(Telephone Service Providers) in France. For the fixed telephone numbers it is France Telecom. For Mobile numbers it is Bouygues telecom, Orange and SFR. Depending on the category of the request(fixed or mobile and if mobile, the corresponding operator) is redirected to the corresponding Tier2-Bloc server. Once the local nameserver cache gets populated most queries are directly send to the Tier-2 bloc until and otherwise the same query is not sent again with the TTL period.

From table 3.2 we know that 82% CPU utilization for Tier-2 Bloc server reaches at around 13,000 queries per second. On looking at the results (Table 3.4) obtained from the measurements, the metrics(loss rate or response time) do not show a big change after this level. The loss rate or response time seem to have a steady growth. At after 15,000 queries a small surge is seen.

Number of Packets sent	Max. response time (ms)	Avg. Response time (ms)	Standard Deviation response time	Loss Rate(%)
0	0	0	0	0
1027	3	0,315	0,72207	0
2026	2	0,615	1,33517	0
2855	7	0,895	2,78809	1
4064	7	1,395	3,91825	1
5085	36	1,78	7,95045	3
5423	12	2,665	11,43635	3
7155	51	4,775	12,8532	5
8167	125	6,305	13,4138	8
9211	173	8,285	14,5212	12
9909	213	8,495	15,3008	15
11210	268	10,745	16,8345	21
11697	323	11,39	18,4538	31
11760	383	10,78	20,073	32
13225	354	11,51	20,702	39
15294	463	13,31	25,307	42
20429	493	15,17	29,529	47
26912	597	18,2	33,382	49
28067	702	24,95	36,679	50
38347	806	25,655	49,115	54
41208	819	26,305	51,776	56
47203	894	26,4724	59,583	59

Table 3.4. Results of the measurements at Tier-2bloc server

Tier-2 number contains all the NAPTR RR related to the registered E.164 telephone numbers. For every query sent this is the final destination. 82% CPU utilization is seen (from table 3.2) around 8,000 queries per second. A sudden surge in the average response time can be seen (from table 3.5) around 9,000 and 20,000 queries initially.

The figures 3.4 and 3.5 gives an idea of the two metrics, loss rate and average response time at each of the target server for different range of Queries per second.

Model

On observing the graph of the results (figures 3.4 and 3.5) one notes that in the beginning loss rate and average response time follows an exponential curve, while after a certain threshold it follows a linear pattern. This threshold value depends on the technical characteristics of the server (Hardware and DNS Software used on the machine).

In case of loss rate and average response time, we could not fit the experimental data with

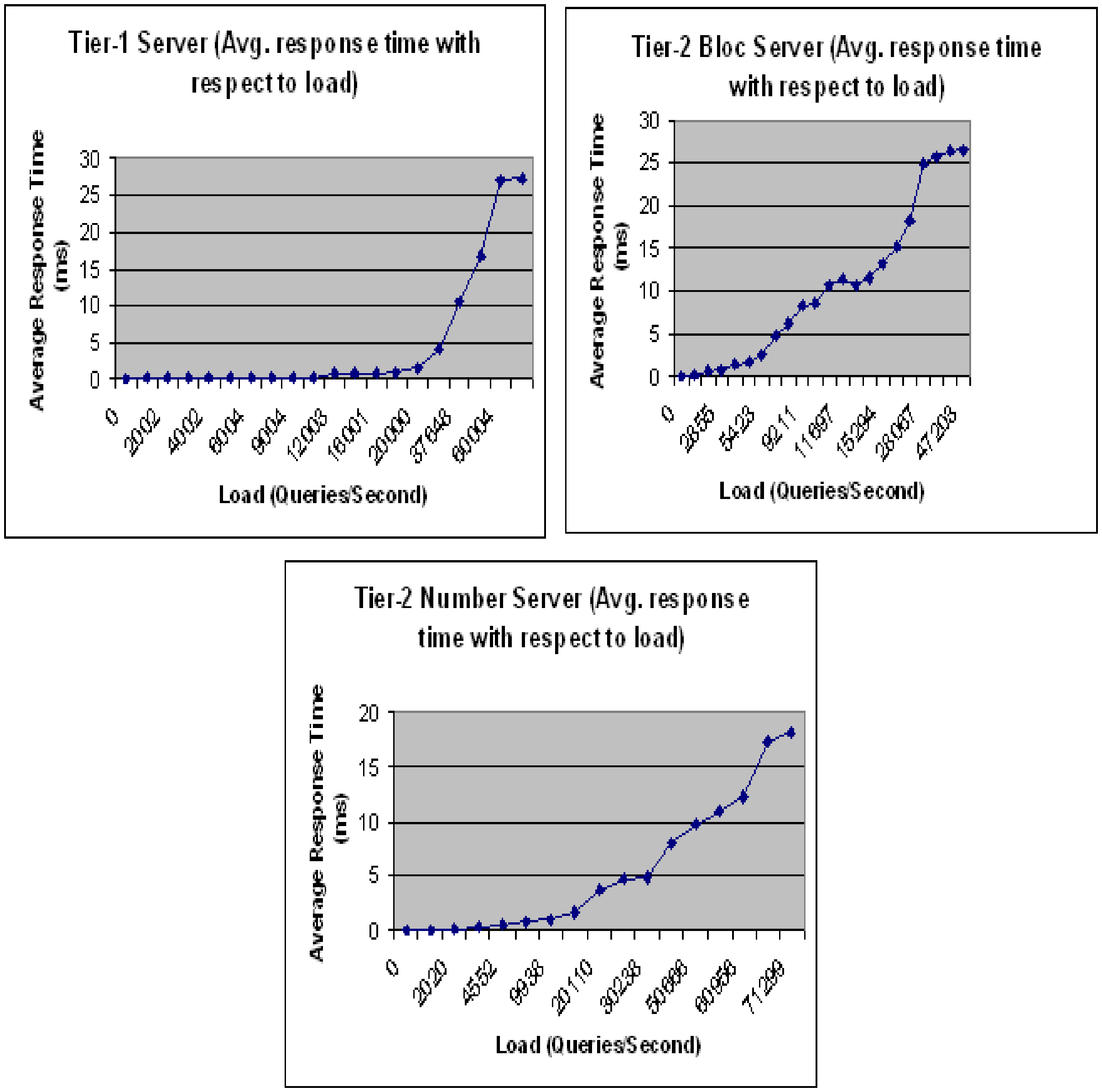


Figure 3.4. Average response time of Tier-1, Tier-2 bloc and Tier-2 number server

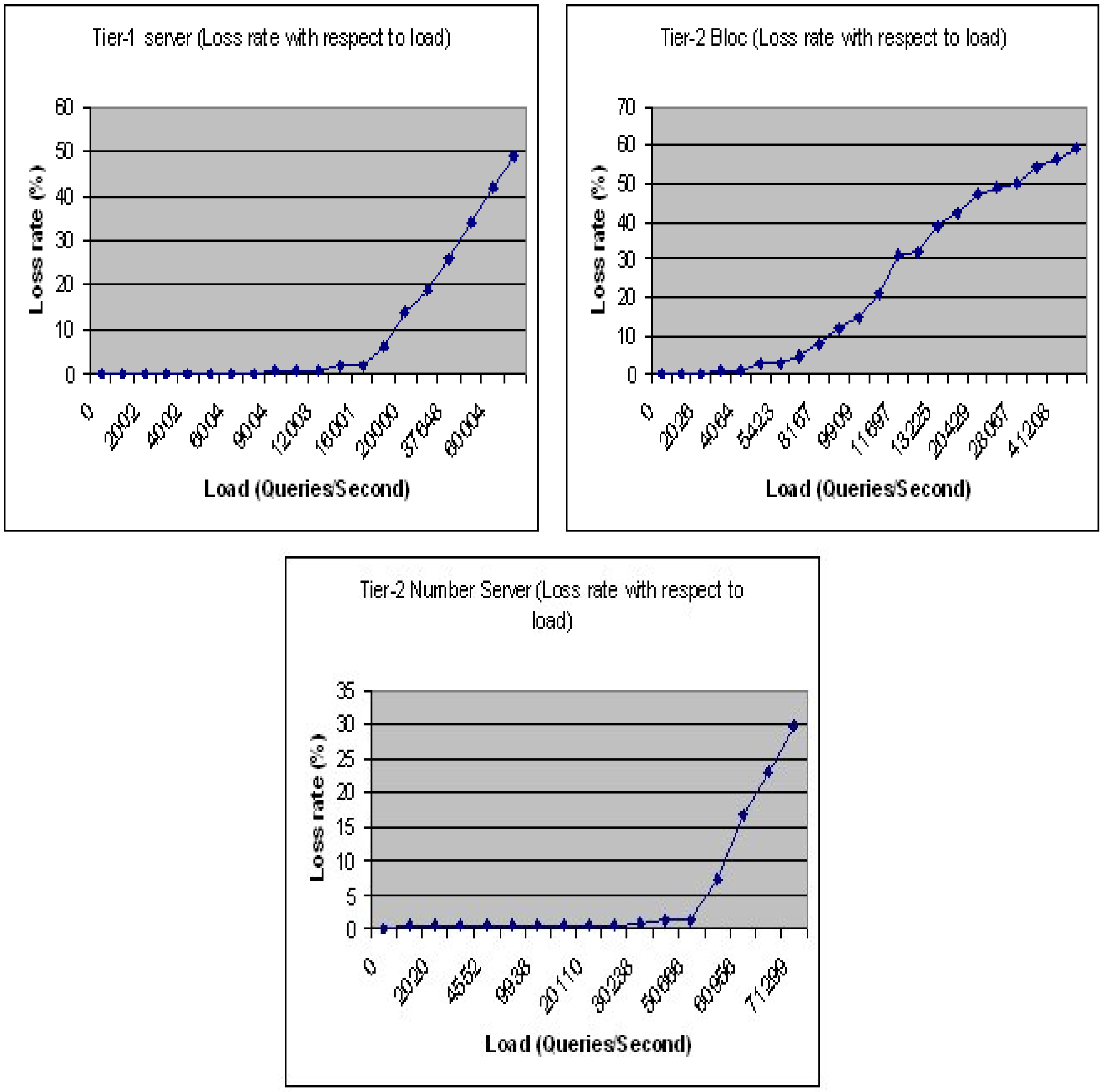


Figure 3.5. loss rate of Tier-1, Tier-2 bloc and Tier-2 number server

Number of Packets sent	Max. response time (ms)	Avg. Response time (ms)	Standard Deviation response time	Loss Rate(%)
0	0	0	0	0
1009	2	0,025	1,18587	0,415
2020	6	0,135	2,677547	0,421
3325	29	0,35	3,775516	0,417
4552	57	0,5125	5,46765	0,442
6119	62	0,82	7,392875	0,553
9938	83	1,05528	9,636861	0,574
10027	108	1,6	9,879856	0,509
20110	176	3,762626	18,55316	0,464
30090	235	4,625	26,022	0,615
30238	281	4,95	27,36198	0,857
40414	433	8,13	36,65	1,3
50666	984	9,815	59,68371	1,272
56234	1023	10,925	77,623	7,413
60956	1934	12,32	87,4676	16,702
64956	1345	17,32	88,323	23,186
71299	1173	18,142	88,38496	29,94

Table 3.5. Results of the measurements at Tier-2number server

one function. So we applied the appropriate fitting techniques in order to find a threshold between exponential and linear part of the distribution. When there is a hybrid model like in our case, fitting techniques were applied on the experimental data modeling [20][72]. The experiment that we conducted to measure the capacity of the servers used in the model (Section 3.4.1), enables us to come up with the threshold value. The point at which the division was made depends on the threshold value. The observation from the average response time graph indicates that the increase in the delay does not have much variation from 1 to 5 ms. But after this the variation is high and the growth of the curve (response time) is at much faster rate.

In case of loss rate there is a sudden variation after a certain point. This is quite visible in the Tier-1 and Tier-2 number servers loss rate graph. This scenario is quite normal. Since the altered Queryperf tool does not take into account the loss, the server gets overloaded with original as well as queries that are resent due to not responding within the TTL period.

In both average response time and loss rate scenarios the threshold value is based on the load at the server. Approximately the threshold value is the maximum capacity of the server. In other words after this threshold one observes a linear growth.

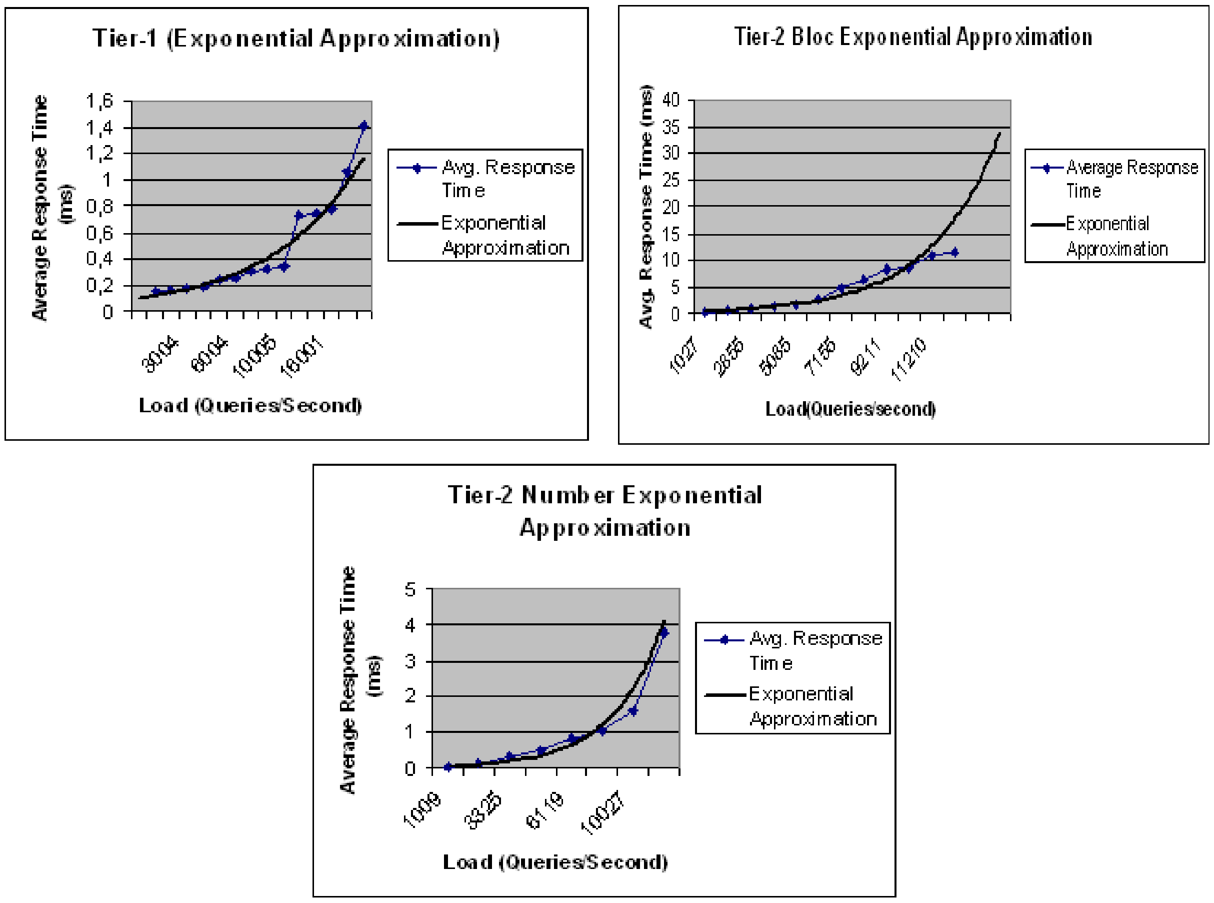


Figure 3.6. Response Time Exponential approximation of Tier-1, Tier-2 bloc and Tier-2 number server

Server	**Type of Approximation**	**Equation Values**	**R-square value**
Tier-1	Exponential	y = $0,0829e^{0,1765x}$	0.9478
Tier-1	Linear	y = 6,269x - 1,753	0,9523
Tier-2 Bloc	Exponential	y = $0,3356^{0,3295x}$	0,9593
Tier-2 Bloc	Linear	y = 2,3604x + 7,3482	0,9285
Tier-2 Number	Exponential	y = $0,0313e^{0,61x}$	0.9172
Tier-2 Number	Linear	y = 2,0256x + 1,6633	0,9617

Table 3.6. Exponential and Linear Approximation equation and Autocorrelation values for Average response time for the three target servers (Tier-1, Tier-2 bloc and Tier-2 number)

Model Validation

By observing the graph in the previous subsection we derive a hypothesis that the two metrics(loss rate and average response time) for the three target servers(Tier-1, Tier-2 Bloc

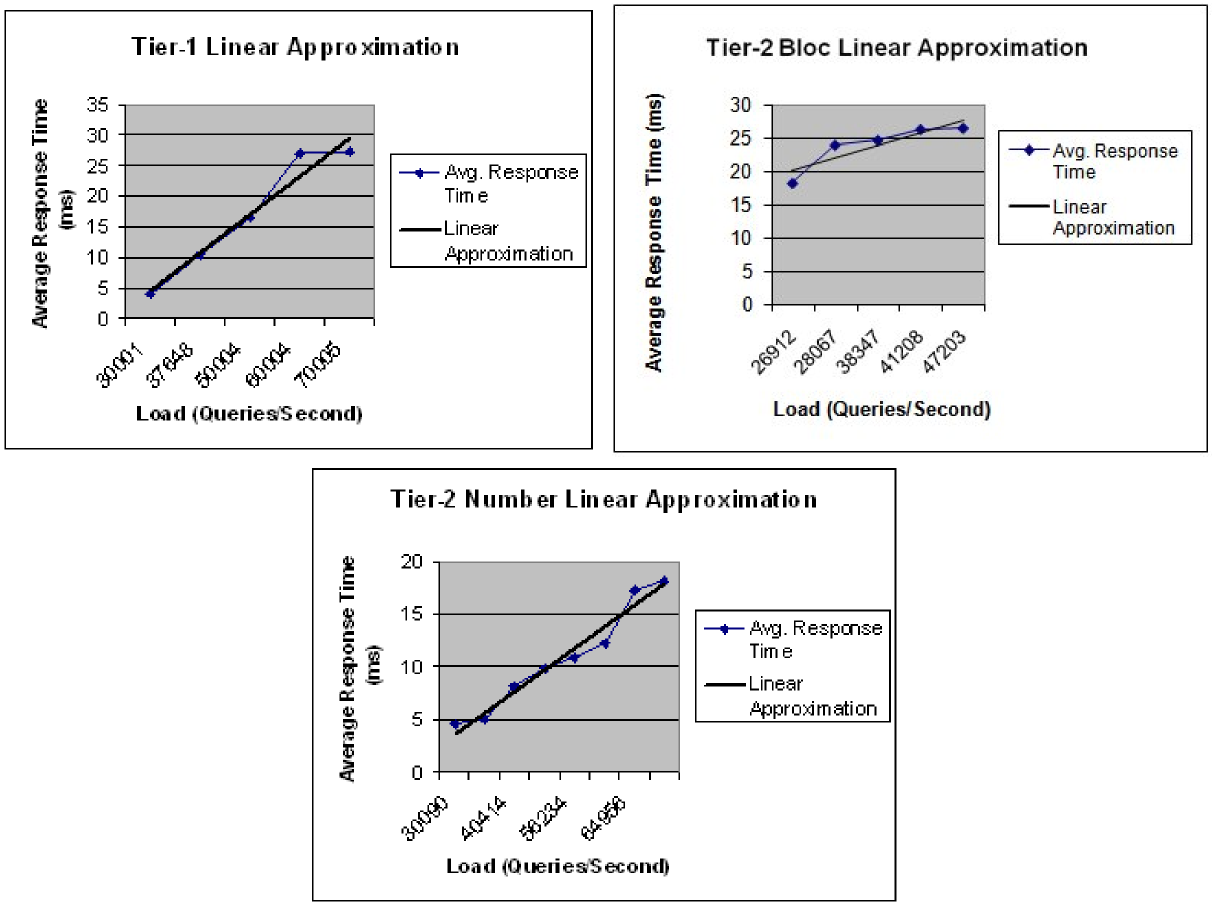


Figure 3.7. Response Time Linear approximation of Tier-1, Tier-2 bloc and Tier-2 number server

and Tier-2 number) initially follow an exponential growth and after a threshold value continue with a linear one. In order to prove our hypothesis we used regression techniques.

The linear function is of the form f(x) = ax + b and the exponential function is of the form f(x) = $ae^{\alpha x}$). The obtained values were divided into exponential and linear curve by the threshold value. For each exponential and linear curve we used linear and exponential regression methods to identify how best the the predicted curve fits the observed curve.

The "goodness of fit" is estimated by the *R-square* value which is the square of the *correlation coefficient*. It is an indicator of how well the model fits the data. An R-square value close to 1.0 indicates that it is a good fit and the value closer to "0" indicates a bad fit.

From the regression methods we identified parameters, which when applied into the exponential and linear equations, best fit exponential and linear part of the real world measurements. Thus parameters(a and b) for both exponential and linear equations were calculated for all the three target servers to be used in the simulation tool. Values obtained for exponential and linear approximation of the average response time and loss rate is shown in the

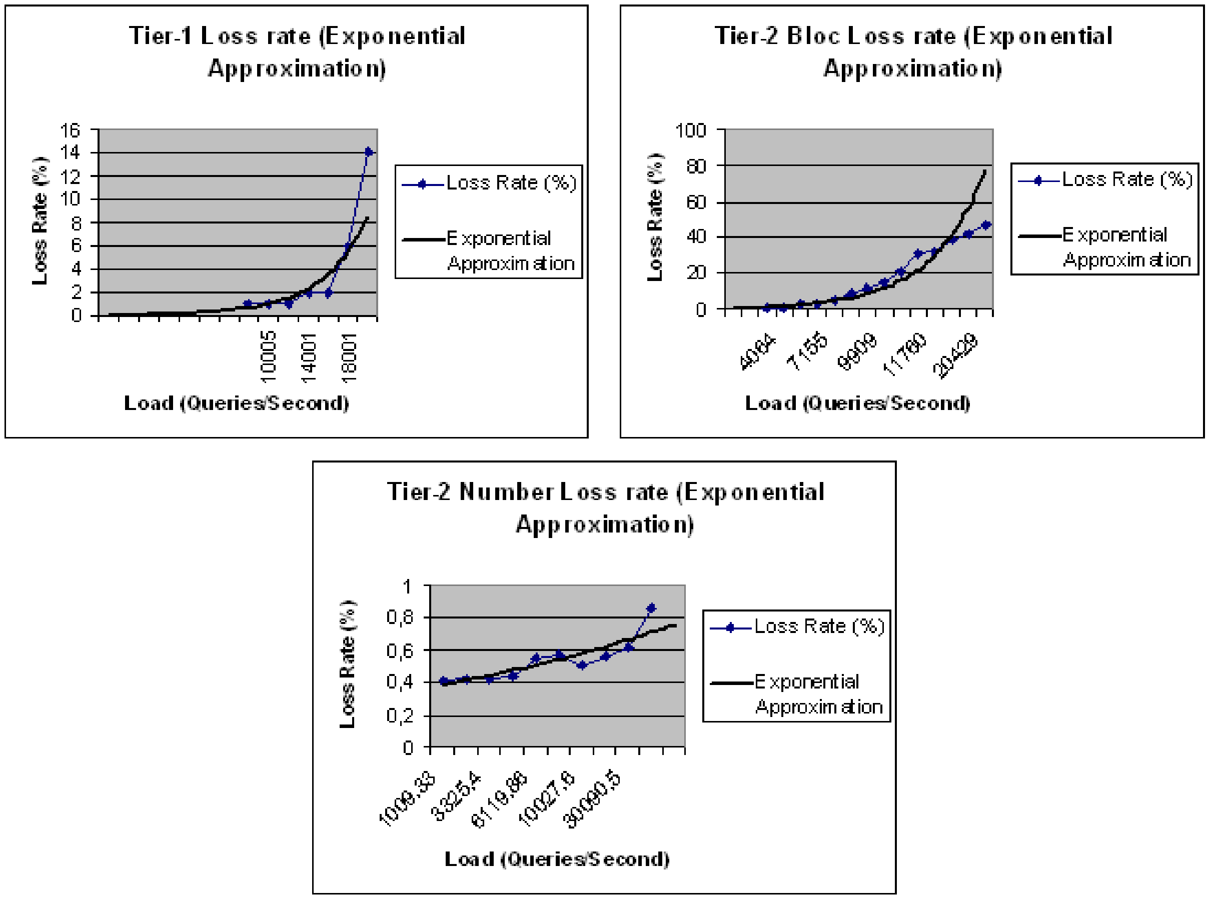


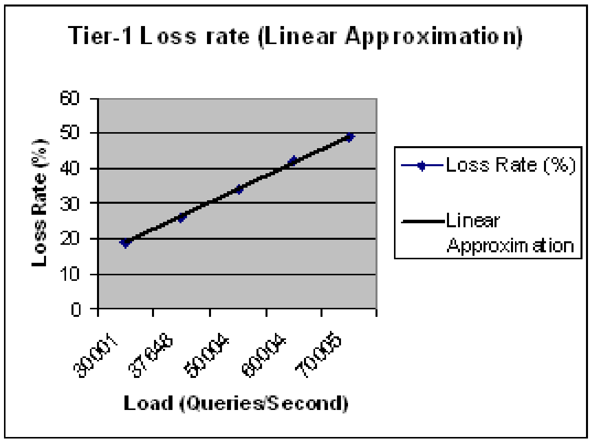
Figure 3.8. Loss Rate Exponential approximation of Tier-1, Tier-2 bloc and Tier-2 number server

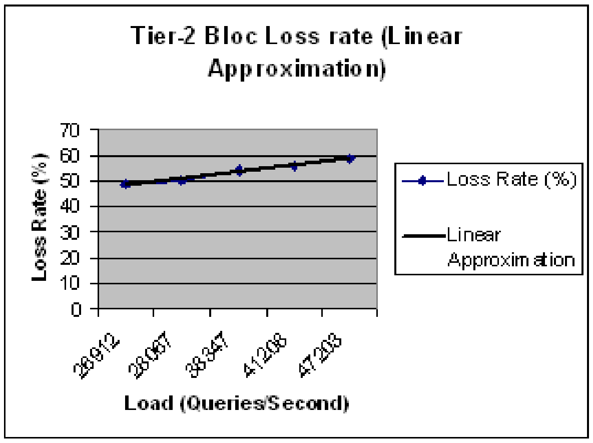
Server	**Type of Approximation**	**Equation Values**	**R-square value**
Tier-1	Exponential	y $= 0,0829e^{0,1765x}$	0,9478
Tier-1	Linear	y = 6,269x - 1,753	0,9523
Tier-2 Bloc	Exponential	y $= 0,3356e^{0,3295x}$	0,9593
Tier-2 Bloc	Linear	y = 2,3604x + 7,3482	0,9285
Tier-2 Number	Exponential	y $= 0,313e^{0,061x}$	0,9172
Tier-2 Number	Linear	y = 7,3109x - 6,2301	0,996

Table 3.7. Exponential and Linear Approximation equation and Autocorrelation values for Loss Rate for the three target servers (Tier-1, Tier-2 bloc and Tier-2 number)

table 3.7. As one can see from the tables the correlation coefficient for all the approximation is quite near to 1. The approximation of model values with experimental data for for both exponential and linear curve and for both average response time and loss rate are shown in the figures 3.6, 3.7, 3.8, 3.9

This model characterizes the whole behavior of each authoritative DNS server under





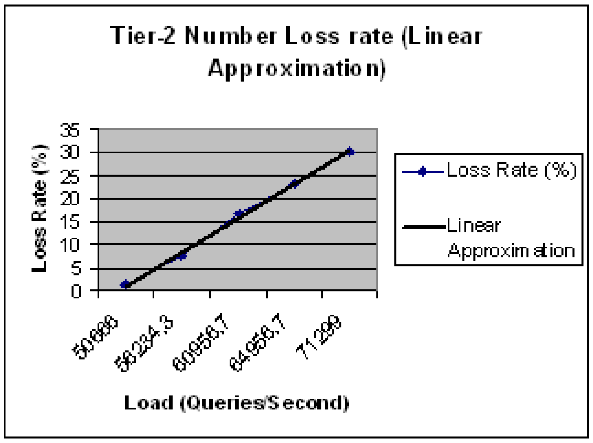


Figure 3.9. Loss Rate Linear approximation of Tier-1, Tier-2 bloc and Tier-2 number server

a wide range of load. The values of the parameters of this model depend on the type of hardware used, the embedded software, the size and type of the DNS database, the type of queries the DNS server receive, the activation or not of DNSSEC.

3.4.4 Measuring and Modeling the DNS IP links

In this section we explain how we measured and modeled the different IP link connections present in the ENUM system. An ordinary ENUM DNS lookup has the following states:

1. Resolver sends request to local name cache server.
2. Local cache nameserver sends request to Tier-1 server.
3. Tier-1 refers local name cache server to another remote nameserver(e.g. Tier-2).
4. Local nameserver sends request to remote nameserver.

5. Remote nameserver replies to local nameserver.

6. Local nameserver replies to host if it has the response for its query.

All the above resolution process passes through IP link. It is natural that the QoS in the IP link affects the DNS response time. Since all these resolutions pass through DNS which is a network service, it is itself dependent on all of the other uncertainties in a network like loss and delay.

In order to study the factors affecting the response time, it is vital to study the QoS factors affecting the IP links connecting the different entities of the French ENUM architecture we are studying. An IP link for us is the network connecting two hosts in the Internet.

Two important metrics of IP links are "loss" and "delay". The ENUM performance is to be measured, using these two metrics. The "loss" is the fraction of the queries that are not receiving a response from the server within a predefined time i.e. the TTL . The "response time" is defined as the interval between sending a query and receiving the corresponding answer. These two are analogous to the packet loss and round trip delay metric in an ordinary internet data transfer.

Measuring each path independently highlights the performance difference between the two paths which may traverse different Internet service providers, and even radically different types of networks. A large-scale study [17] [74]of Internet routing has found that paths through the Internet are often asymmetric meaning that the routers visited by the packet in the forward and reverse direction often differ. Also, even if the packets go through the same route, and of the same size, the packets experience different levels of queuing inside the network.

The usual way of finding the delay between one host to another host is using RTT. RTT make a poor approximation of identifying the delay between two hosts. For example if the reverse path gets congested, the reverse delay gets much larger than the forward delay. So identifying the delay between the two hosts using the method RTT/2 is flawed in this aspect.

Measuring loss using round trip measurements is also not appropriate. It can have its implications in both TCP and UDP based applications. Usual TCP transactions consist of a sender which transmits data and a receiver which receives acknowledgement. If loss occurs in the reverse direction (i.e. from receiver to the sender), the sender assumes that there is a bottleneck and reduces its transmission rate, even if the link in the forward direction (i.e from sender to the receiver) has the capacity to handle the data transmissions. It is the same in the UDP based applications like the DNS where the link in one direction has acceptable QoS while the opposite direction does not.

The IPPM (Internet Protocol Performance Metrics) group defines the methodologies for measuring this two metrics. RFC_2679 [34] explains the methodology to study *one_way delay* and RFC_2680 [35] explains the methodology to study *one_way packet loss* across Internet.

Measurements Techniques

There has considerable research done on measuring the Internet. A literature survey on this topic is not our focus. One can collect detailed information about the transmission of packets on the network including their timing structure and contents, it can be done by specialized network measurement hardware or software. We look into two of the important techniques that has been already used to measure the Internet to understand which of these techniques could be useful to us or whether we have to develop a new one.

Active Measurements In this measurement approach, packets are generated by a measurement device to probe the Internet and measure its characteristics. Active measurements, even though intrusive (when one sends packets into the network) are the best technique to know the QoS of a link because one can monitor the effect of the network upon a small flow of packets along its route.

Passive Measurements In comparison to the active measurements, the passive measurements do not inject test packets into the network. They require capturing of packets and their corresponding timestamps transmitted by applications running on network attached devices (e.g. switches and routers) over various network paths. Some of the popular passive measurement techniques include collecting Simple Network Management Protocol (SNMP) data, Syslog data and NetFlow data from switches and routers in the network. SNMP data provides switch-level and router-level information such as availability, utilization, packet errors and discards. Syslog data provides details of activities and failures of network switches and routers. NetFlow data provides bandwidth/link utilization information between network backbone routers. This information could be used to determine the flow-level type, duration and amount of application traffic traversing the network.

The computation of the one-way delay or loss requires generation of unique packets(normally with a unique packet sequence number) and a timestamp(for delay calculation) at the observation points(source and destination). All these details are needed for each packet that is to be measured. Non-intrusive measurements like Passive measurements for one-way delay, require the transport and correlation of measurement data at the multiple observation points.

For abstraction purpose, at this point we are not concerned about the internal characteristics of the system. Our requirement is to have an idea about the loss and delay between the local name cache server and the target servers. For this purpose active measurements are enough. Hence, we decided to use active measurement to measure the loss and delay metrics for our experiment.

Measurements

During an usual name resolution process, the DNS uses UDP for making queries and receiving the response for the query. When the DNS application in a host wants to make a query, it constructs a DNS query message and passes the message to a UDP socket. Without performing any handshaking, UDP adds a header field to the message and passes the resulting segment to the network layer. The network layer encapsulates the UDP segment into a datagram and sends the datagram to a nameserver. The DNS application at the querying host then waits for a reply to its query. If it doesn't receive a reply (possibly because UDP lost the query or the reply), it either tries sending the query to another nameserver, or it informs the invoking application that it can't get a reply.

Since DNS uses UDP, we used the same transport protocol for our measurements. A simple utility was developed and used for performing the active measurements. It comprises of a "Client" and a "Server" program. The client sends UDP datagrams at a fixed interval (Constant Bit Rate) or using an exponential (Poisson) distribution. Each packet contains a unique sequence number. When the packet is sent it is time stamped and this is recorded in a log file with the sequence number of the packet. The log file looks as in the table 3.8

Sequence number	TimeStamp
61	1094728989.131322
62	1094728989.158886
63	1094728989.187941

Table 3.8. Packet Format in the Log file

When the server at the receiving end receives the datagrams sent by the client, it performs a time stamp of the received time with the sequence number of the packet in another log file.

After the completion of the active measurements the log file of the server and the client are collected. The one-way delay and loss can be identified as follows:

Let us assign

Tc - Timestamp of the packet sent from the client

Ts - Timestamp of the packet received at the server

Ts - Tc of a unique packet sequence number will give the time taken by the packet to travel from the client end to the sender end. i.e. the one-way delay. We used the libpcap and libnet [10] libraries for the measures to be more accurate.

The graph (3.10) shows the one-way delay calculated (Ts - Tc) between the computer at France Telecom (Tier-2 Block) and AFNIC (Tier-1 server).

Observing from the graph (3.10) one notes that the delay is more than 16ms and that the delay cannot be negative. A simple ping from the sender to the receiver comes around 5ms. Now we will investigate into what could be the problem and how to resolve it.

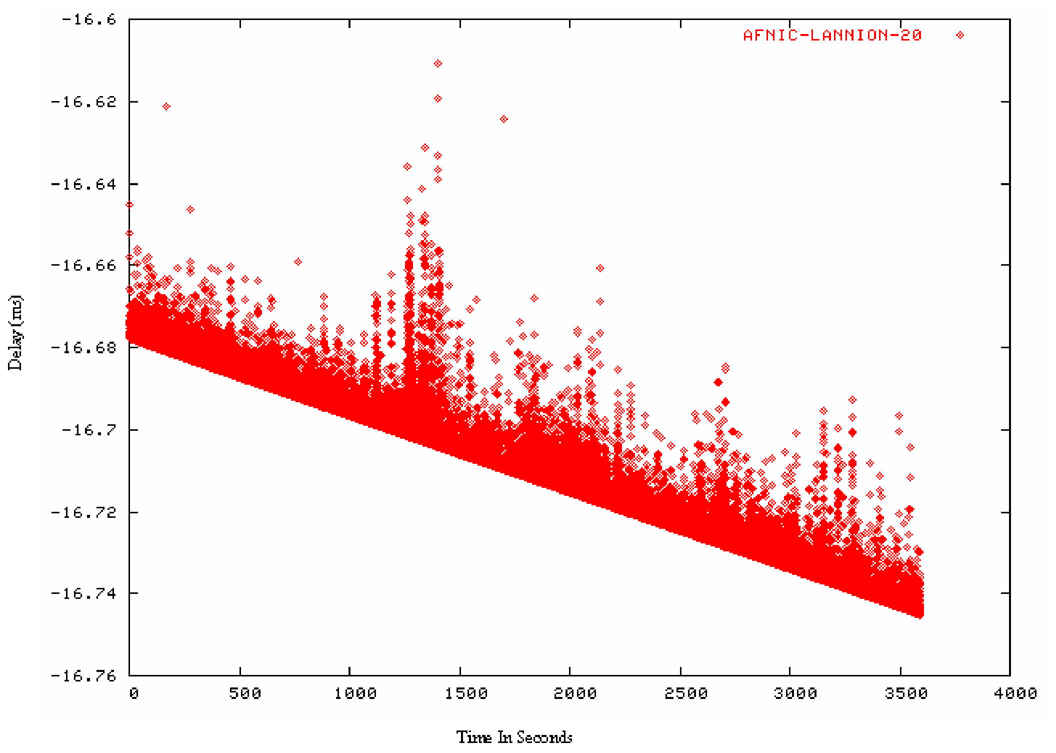


Figure 3.10. Raw One-way delay graph between Tier-1 and Tier-2 Number server

Clearing the Traces

Real delay values will be positive. Therefore, it does not make sense to report a negative value as a real delay. It is true that there is error in the results obtained. One by one we try to identify the causes and eliminate the error.

- The server at the receiving end is connected to the Internet. So it may not receive only the packet from the client. Packet received can be from another source. One way of eliminating the cross traffic is to eliminate the packets with sequence numbers which are not corresponding to the ones that have been received from the client. Another case may be that there are two packets with same sequence number. In this case consider the packet which is quite closer to the previously received packet. For example if the sequence number of the packets is as follows:
  1 2 3 4 7 5 6 7
  It is normal to eliminate the packet with sequence number "7" which is immediately after "4".

- If the packet fails to arrive within a reasonable period of time, the one-way delay is taken to be infinite. If (Ts - Tc) > Reasonable time i.e. the timeout value then that packet is considered loss.

- There are cases when the timestamp is completely unrelated to the average timestamp.

This cause may be due to the error in the clock synchronization at the measurement points.

Removing the Error Caused by NTP clock Synchronization

When we add timestamps to the packet both at the sender and the receiver, the timestamp is given by the clock of the sender or the receiver. The clock at the site of measurements normally synchronizes the local clock to the NTP (Network Time Protocol) [68] server. This NTP server is in turn synchronized to an atomic clock or a Global Positioning System (GPS) that receives time from satellites.

Actually there are other devices like GPS which are more precise than NTP. In case of GPS, the timestamps are very precise up to 100 nano seconds. But the disadvantage in GPS is that it is expensive and there needs to be an antenna at every source and destination. Though NTP is not very precise and sensitive to network conditions, measurements with NTP do not need any additional hardware. In our measurements for the ease of use and ubiquity we used NTP.

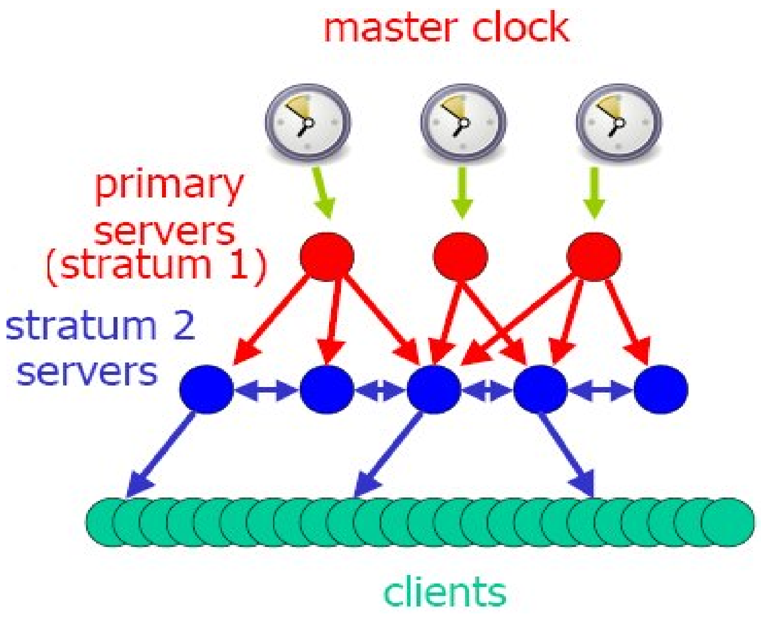


Figure 3.11. NTP Clock Synchronization Scenario

NTP uses a hierarchical system of clock *strata*. The stratum level defines the distance from the system clock and the associated frequency. As shown in the figure 3.11 Stratum 0 is the master clock(atomic clock) which is usually connected to computers and not networks. Stratum 1 is attached to the master clock(stratum 0). Stratum 1 is also connected to the network and act as servers for timing request from stratum 2 servers. Normally a Stratum 2 computer will reference a number of Stratum 1 servers and use the NTP algorithm to gather the best data sample, dropping any Stratum 1 servers that seem obviously wrong. Stratum 2 computers will peer with other Stratum 2 computers to provide more stable and robust time for all devices in the peer group. Stratum 2 computers normally act as servers for Stratum 3 NTP requests. The NTP server at INT is normally a Stratum 3 NTP server.

One-way delay measurement which is shown in the figure(3.10), one at France telecom and another at AFNIC could be synchronizing with different Stratum 3 servers. Basically, the NTP client at each measurement site sends a packet containing its timestamp to its configured NTP server at frequent intervals. When the server receives a packet, it will in turn store its own timestamp and transmit the timestamp back to the client. The client logs the receipt time in order to estimate the traveling time of the packet. This process of packet exchange will continue until the client gets the accurate time.

There are two clocks(at each measurement site) involved in synchronizing with its configured ntp server. Both of them are not true clocks, i.e. they adjust the time according to the master clock. Since both the clocks are involved in measuring delay, synchronization of the two clocks become an issue in the accuracy of the delay measurement. But the actual measurements obtained are likely to have an error ratio(due to non synchronization of the two clocks) added with the actual delay measurements. This is usually called *synchronization error*. Thus if the synchronization error is known it is possible to get the correct delay measurements.

Let us assign:

Tsrc - Time at the sender

Tdst - Time at the receiver

Tsynch - Synchronization error

One-way delay = (Tdst - Tsrc)$\pm$Tsynch

It is very important to remove the "Tsynch" factor from the delay measurements to get an actual assessment of the one-way delay. Now let's see what are the factors affecting the "Tsynch" factor

Getting to know the clock terminologies is quite useful to remove the Tsynch from the original trace. They are:

Frequency: The rate at which the clock progresses.

Skew: The two clocks may run at different frequencies. The difference in frequency is called the clock skew. It is therefore possible for the receiver to receive a packet from the future resulting in a negative delay as in the case of our one-way delay measurement.

Offset: The difference between the time reported by the clock and the true time.

Vern Paxson in his thesis [76] uses forward and reverse path measurements of delay between a pair of hosts(sender and receiver) to deal with clock synchronization problems. A technique for solving the clock skew issue is done using linear regression [75]. From the cluster of delay points obtained a line is fit by minimizing the sum of the squares of the distance between the line and the data points. Another method called piecewise minimum algorithm is used to estimate the clock skew, where the total delay trace is partitioned into segments. From each segments minimum delay is picked. Now all these minimum delays are collected for a line segment. The resulting concatenation of line segment is taken as estimate of the skew.

The linear regression method works when the distribution of the data are normal, which is rare. Furthermore sudden surge of congestion can add to significant amount of deviation for skew slope estimation. The minimum square segments obtained via the piecewise algorithm are unlikely to a straight line or has the same slope, thus not providing the correct skew estimation.

The algorithms used in [89] and [57]have been used to remove the Tsynch factor. We used the regression algorithm by Moon et al[89] to fit a line lying under all delay points and as closely to them as possible. The line satisfies that all the delay points are above or on the line, and the sum of vertical distance between each point and the line reaches the minimum and finally with the following assumptions the clock skew is estimated:

- There is no clock reset(The clock reset is done in a couple of days through system calls).
- On an average there is a minimum delay between two packets.

This clock skew estimation is based on the work by Zhang et al[57]. They use a convex hull approach. *A convex hull is the minimum set of points X in a real vector space V.* On the delay measurements obtained, after applying the regression algorithm, the convex hull of delay points is calculated and an estimated skew value is found according to different optimal requirements. The requirements include

1. Minimizing the sum of vertical distance between the point and the line
2. Minimizing the area between the segments made up by the points and the line and
3. Maximizing the number of points on the line

Now we try to remove the clock reset. A Clock reset can be possible during the measurement period. Our solution to this problem is to segment the measured data into multiple divisions and apply convex hull algorithm on each of this partitions.

The minimum delay can be estimated by the smallest observed ping. Since the ping is the sum of the delay between the two asymmetric paths the minimal one way delay can be calculated by $\frac{RTT_{min}}{2}$. Since it is not symmetric it is important to take into account of the number of hops on both the paths and get the barycenter of the results.

The Clock Offset is estimated by the following manner. Lets assume the sum of the clock offset is β i.e the clock offset on both the asymmetric paths.

Delay from Source to destination $d_{s->r} = \overline{d_{s->r}} + \beta$

Delay from destination to Source $d_{r->s} = \overline{d_{r->s}} - \beta$

So the Offset $\beta = \frac{(d_{s->r} - \overline{d_{s->r}}) - (d_{r->s} - \overline{d_{r->s}})}{2}$

The results obtained thus obtained on the original measurements(Fig: 3.10) by the methods explained to clear the traces is as shown in the figure (3.12)

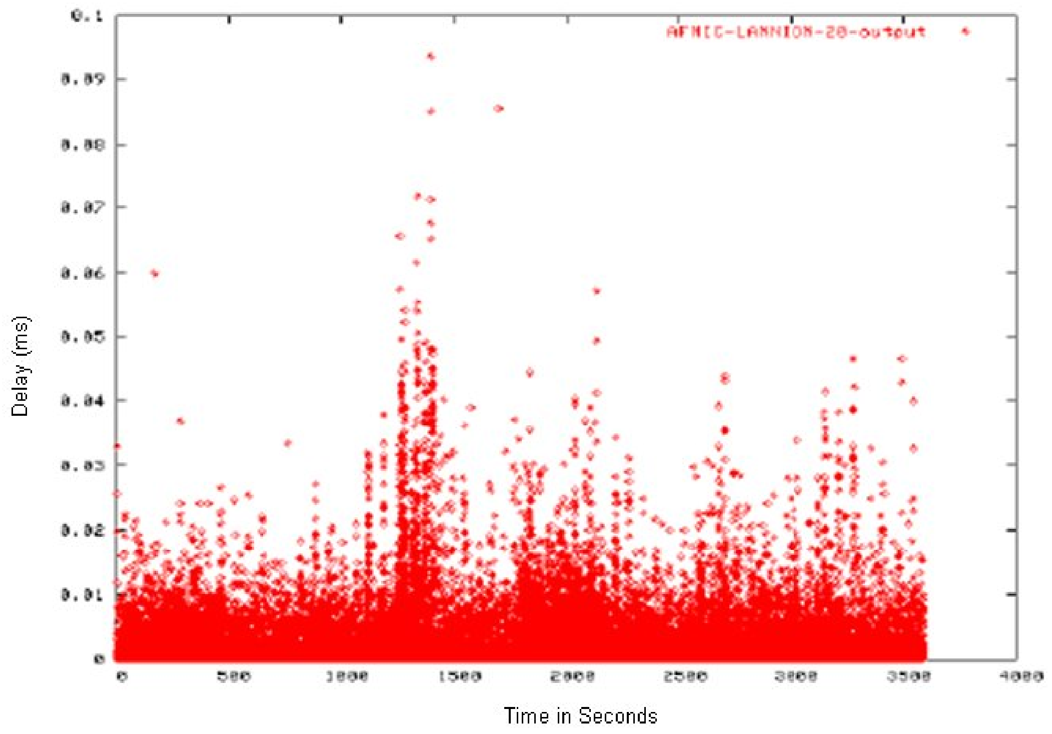


Figure 3.12. Clear Trace of One-way delay graph between Tier-1 and Tier-2 Block server

3.4.5 Modeling the IP link

In the sub section 3.4.3 we presented with a mathematical model (exponential and linear equations) we developed to model loss rate and response time between the local cache and each of the target authoritative servers. Parameters were obtained from this mathematical model to be used in the Simulation model in chapter 4. In the subsection 3.4.10 we measure the global response time for the ENUM queries. The global response time depends on the QoS of the Internet links between the different nodes connected, by which the data packets are transmitted or received. In Subsection 3.4.4 the two important metrics; delay and loss, that are particulary important have been measured. Now we want to model the traces of this measurement and obtain parameters, which will also be used in the simulation model in chapter 4.

3.4.6 Usage of HMM for Modeling the Internet Link

To have the simulation model reflect the features of the system under study, two metrics, delay and loss, had to be modeled. To create and evaluate an Internet packet delay and loss model, we first need to collect and analyze Internet packet delay traces [**?**]. Collection of traces is done in subsection 3.4.4 and now we will proceed to the modeling part.

There has been considerable work done on modeling loss and delay in the Internet links. This work [25] shows RTT delay in the Internet can be modeled by Weibull Distribution while another work [33] proposes Erlang distribution. A packet loss in the Internet indicates congestion. Similar to the delay models research has also been done to model the loss factor in the Internet. [51] uses the combined feature of the two state Markov model also known as the Gilbert model and the inter-loss distance to characterize the loss scenario. For ENUM type of data transmission, RTT modeling cannot be feasible and we have discussed the drawback already in subsection 3.4.4.

In our case in the IP link, packets are loaded at the transmitting node and sent. This packet may be received at the receiver or may be lost. If the packet comes after a maximum delay it is considered loss. Now we have to model this scenario with one-way delay and loss. We don't want the model to be complex like fractals or wavelets since it will be difficult to build the model in the simulation. A simple Poisson model is not realistic enough [77] to model Internet traffic.

Markov chains are a useful tool in modeling communication systems [39]. These two works [29],[37] on modeling burst error channels for bit transmission show how a simple two state HMM are effective in characterizing real communication channels. End-to-End packet channels also show burst-loss behavior similar to bit transmission channels. This work [47] used an Hidden Markov Models (HMM) based delay modeling of TCP traffic to infer loss nature in hybrid wired/wireless environments. The paper by Wei et al. [92] gave the basic idea for our work. They develop an algorithm to infer the continuous time HMM from a series of end-to-end delay and loss observations of probe packets. They use the model to simulate network environments for network performance evaluation. Finally they validate the simulation method in NS-2 as well as over the Internet. Salamatian and Vaton [84] found that an HMM trained with experimental data seems to capture channel loss behavior and found that HMM with two to four hidden states fit well with experimental data.

Our model of an IP link has four HMMs (Fig:3.13, one for the delay and one for loss on both direction of the IP link. To validate the model, we ran simulation of the HMM driven process and applied the Expected - Maximization(EM) algorithm methodology. The parameters of the HMM found through this methodology were very close to the measurement data, thus validating the model. Eventhough this has already been explained by [22] done in our own lab, we provide a brief explanation of the process that is followed.

The model developed need to reflect the different properties of the IP link:

- As discussed in section 3.4.4 IP link is asymmetric.

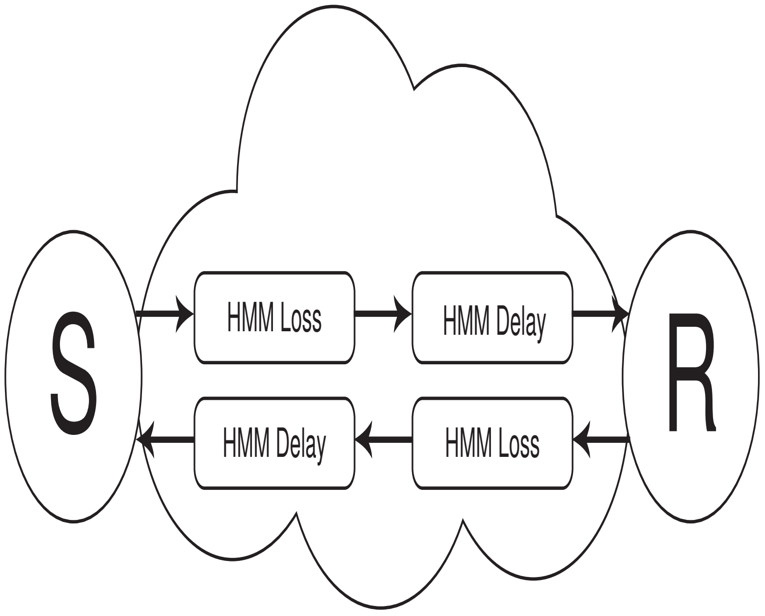


Figure 3.13. IP Link Model

- The delay is distributed on a hop by hop, but the loss occurs only when there is a bottleneck.

In this case the delay and the loss are not correlated. Each HMM chosen represents a state of the network. The Viterbi algorithm helps us to understand the state traversed and thus the behavior of the model in time.

3.4.7 Loss

Packet loss is a discrete event. HMM uses two states to model this event. The first is that the packet is lost and the second one is the packet is received. If the packet is lost it is considered as "1", if the packet is received it is considered as "0".

The model (Fig:3.14) built represents a single server queue with a bottleneck. The buffer capacity of the queue is "N" including the client in service. The queue is a drop-tail queue. The packets that are not treated are considered lost. The traffic is a combination of two independent processes. The first is the real traffic which has been generated by the active measures and the second is the cross traffic. The first traffic is Poisson or Constant Bit Rate (CBR). The cross traffic can be the superposition of different distributions. The cross traffic is modeled by Markov Modulated Poisson Process (MMPP) with "K" states. When the MMPP is in state "i" the load arriving in the queue is in accordance of a poisson process with rate λi. On other terms the interarrival time is exponentially distributed with mean $1/\lambda i$. The transition states is managed by a Continuous Time Markov Chain (CTMC) as in the figure 3.15

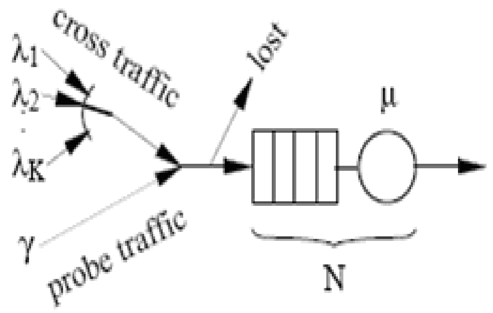


Figure 3.14. The Loss Model

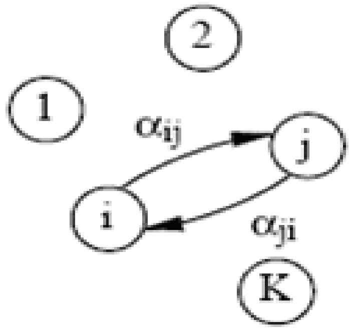


Figure 3.15. CMTC which governs the MMPP traffic

The model is defined by the following parameters:

μ - Service rate of the server

N - the buffer capacity

γ - The load introduce by the probing traffic

λi - The load caused by the cross traffic when the CTMC is in the state "i"

The intensity of the traffic for each state "i" of the MMPP traffic is defined by $\rho_i = \lambda_i + \gamma/\mu$

Once we have defined the model we have to determine all the parameters unknown to better understand the loss process (i.e. observed from the real network measurements). Once all the unknown parameters are estimated, it is easy to study the different parameters of the model.

With N=4 and K=2, the service time will be exponential. The horizontal transition matrix represents the change in size of the queue and the vertical transition represents the state transition in MMPP. The packet arrival rate in the model is going to be higher than the MMPP transition state. This signifies that for every transition state, the queue has

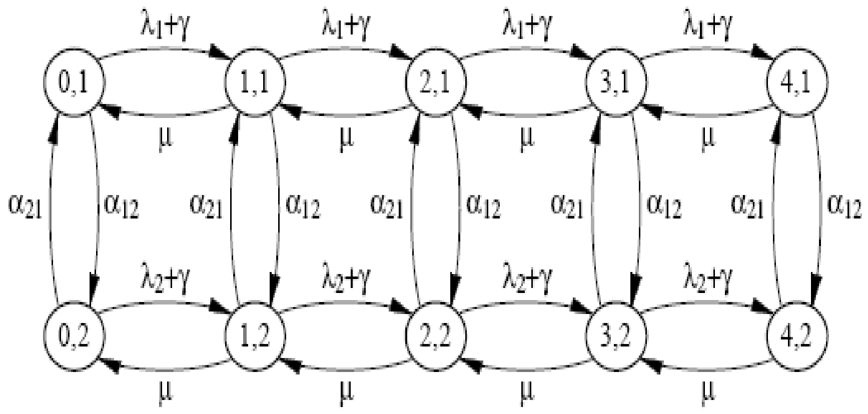


Figure 3.16. Markov Chains of Model M

enough time to attain stationary distribution. With this hypothesis the global behavior of the queue can be described as a mixture of stationary behaviors, each corresponding to a state given by MMPP traffic. This approach is validated in subsection 3.4.8

In order to analyse the loss behavior at the egress of the queue, the behavior of each state of the MMPP queue is studied separately. Each of these states follow a model with a M/M/1/N queue as explained in [87]. This model is explained in 3.17

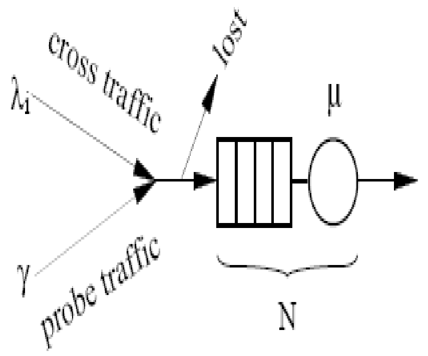


Figure 3.17. The Loss Model Mi

The loss observed at the egress of the model is given by equation

$$\rho_i = \frac{1}{1-\rho_i}$$

The argument that is made is that the interval between the probe packet during the active measurements is greater than the time required for the queues to attain the stationary state, thus the losses are independent.

The behavior of MMPP/M/1/N queue can be described with a mixture of stationary behaviors, each corresponding to a state given by MMPP ingress traffic. The behavior of loss at MMPP/M/1/N egress can be described by a mixture of loss process without memory, each having a loss probability directly linked to the load at the arriving at the same moment. As the state of the arrival follows a Markov (because of MMPP behavior), the loss at the egress of the MMPP/M/1/N follows a HMM, with MMPP state acts just like the HMM state [23].

Estimation of Parameters

Estimation of μ μ is the service rate of the bottle neck server. This value is estimated using the packet-pair technique followed in this paper [21]. If two packets of same size are sent within a small interval and if these packets arrive at the bottleneck server back to back with service rate μ, the receiver will receive the packets with a transmission delay $\tau = \frac{L}{\mu}$. Thus μ can be calculated with the help of the dispersion measured by τ. In the experiments used it is assured the parameter is known.

Estimation of N N represents the capacity of the bottleneck server. Since the interest is in modeling loss, the intensity of the traffic ρ should be greater than 1. To have a realistic scenario N has to kept at a large value. 'N' is given an arbitrary value '50' which signifies that all the loss rate is around 1.9% and the intensity of the traffic is greater than 1.

Estimation of γ γ is the load and the value is evidently known

Estimation of the traffic ρ It has been already seen that the loss at MMPP/M/1/N follows an HMM with a relation between loss probability at each state and the traffic at the ingress queue. In order to estimate the traffic at each state ρ_i and rate λ_i, the EM algorithm is used, which as explained earlier gives the number of states necessary for describing the loss of the real measurements done. ρ_i at each state is given by the equation

$\rho_i = \frac{1}{1-Pr_i}$ and

$\lambda_i = \frac{\mu}{1-Pr_i}$

Infinitesimal Q Value Estimator The formula for calculating the infinitesimal generator 'Q' of the probability transition matric τ is given by

$$P(t) = e^{qt} = \sum_{j=0}^{\infty} \frac{(Qt)^j}{j!}$$

The states 'K' controls the dynamicity of the CTMC. We can thus obtain the

$$Q = \frac{y}{k} log\Gamma$$

The computation may be obtained by digitalizing the matrix Γ. Thus Q value is

$$Q = \frac{y}{k} U(logD)U^{-1}$$

where logD is the diagonal matrix
For every value K, a specific infinitesimal matrix is resolved. As 'k' value increases the MMPP traffic dynamically changes rapidly. In order to interpret and model the active measurements the value of 'k' is taken as 1.

3.4.8 Validation of the Loss Model

Here we apply the model inferred procedure to observed traces generated by real measurements. After a model is developed it is necessary to validate whether it is closer to the original model.

The real trace contains 8 hours of Active measurements. In this measurement 100 bytes of data are sent every 50ms. The loss observed from this experiment is around 18.58%. The loss rate is measured on a 5 second window. The EM algorithm was used to estimate the HMM values [84].

$$p = \begin{pmatrix} 0.95 & 0.206 & 0.07 \end{pmatrix}.$$

$$\Gamma = \begin{pmatrix} 0.937 & 0.0623 & 0.0006 \\ 0.0026 & 0.9973 & 0.0002 \\ 0.0000 & 0.0004 & 0.9996 \end{pmatrix}.$$

$$\pi = \begin{pmatrix} 0.0267 & 0.6581 & 0.3152 \end{pmatrix}.$$

The above parameters are transferred to orient towards the model constructed

$$\rho = \begin{pmatrix} 20 & 1.2594 & 1.07 \end{pmatrix}.$$

$$\Gamma = \begin{pmatrix} 0.937 & 0.0623 & 0.0006 \\ 0.0026 & 0.9973 & 0.0002 \\ 0.0000 & 0.0004 & 0.9996 \end{pmatrix}.$$

Simulation was done in NS-2. On applying the same procedure used for real trace, estimation was also done on the simulated trace, the values thus obtained are as follows:

$$p = \begin{pmatrix} 0.94 & 0.204 & 0.07 \end{pmatrix}.$$

$$\Gamma = \begin{pmatrix} 0.9431 & 0.0568 & 0.0000 \\ 0.0022 & 0.9976 & 0.0002 \\ 0.0000 & 0.0008 & 0.9992 \end{pmatrix}.$$

By comparing the real and simulated values; which seems to be identical the loss model was validated.

3.4.9 Delay Model

According to [25] one-way delay is composed of four components:

- *Processing delay:* The time needed to process the packet at each node and prepare it for transmission.
- *Transmission delay:* The time needed to transmit the entire packet from first bit to last bit over a communication link.
- *Propagation delay:* is determined by the travel time of an electromagnetic wave through the physical channel of the communication path and is independent of actual traffic on the link.
- *Queuing delay:* The waiting time of the packets in the buffer of routers before transmission.

From a measurement point of view, the end-to-end delay over a fixed path can be best defined by two components *processing* delay and a *queuing* delay. The delay distribution is hop by hop and it increases with each node that the packet traverses. The delay distribution over a fixed path follows a gaussian distribution [41].

The delay is represented by continuous hidden Markov process. Two states are used to model this event. A truncated Gaussian distribution with average μ and standard deviation σ is associated with each state of the continuous Hidden Markov chain. Since delays cannot be negative the distribution is truncated to obtain positive values.

The principal difficulty in modeling the delay is to get the number of gaussian and thus the number of states. Number of solutions have been envisioned to find the parameters of the delay Markov chain:

- Discretion of the delay value.
- Analyze the gaussian mixture.
- Use AIC(Akaike information criterion) or BIC(Bayesian information criterion).

The discretion of the delay model is simple, but the question is how to choose the intervals for separating the different gaussian distributions. The analysis of series of measurements permit to separate the different gaussian distributions. The analysis of the gaussian mixture provokes information loss. We loose the transition order of the hidden Markov chain and we will not be able to apply the viterbi algorithm. For the reason we have decided to use AIC and BIC for evaluation of the number of states. The nodes in the measured network can be modeled by a simple M/M/1 queue.

The M/M/1 queue has one server (Fig:3.18. The arrival rate is Poisson with parameter λ. Like in the case of loss, it is assumed that the service time is independent and follows an exponential distribution with parameter μ. The random process $N_{t,t\epsilon\Re}$, the number of

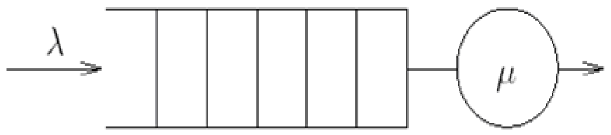


Figure 3.18. M/M/1 queue

clients in a queue at an instant 't' is a continuous time Markov process with value N. The load in the queue is defined by $\rho = \frac{\lambda}{\mu}$. The queue is stable if $\rho < 1$. In a stationary state the probability for a new client to find K clients in the queue will be P(K) = (1 - ρ) ρ^K.

On applying the Littles law L = Rλ where L is the average number of customers in a stable system R and λ is the average arrival rate, the average response time is

L = E(kP(k)) = $\frac{\rho}{1-\rho}$

One obtains on using the relation $\rho = \frac{\lambda}{\mu}$ and the Littles law

R = $\frac{L}{\lambda} = \frac{1}{\mu-\lambda}$

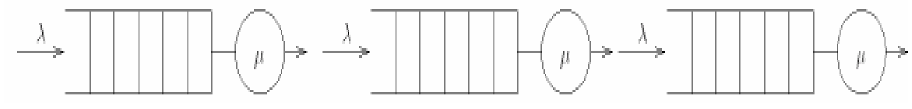


Figure 3.19. 'm' M/M/1 queues

On studying "m" queues in M/M/1 where each queue is independent and for simplicity we use the same parameters λ and μ. One can express the average transit time in the "m" queues as

$$E(transittime) = \sum_{i=1}^{m} R_i = \frac{m}{\mu - \lambda}$$

The variance and the sum of the variance is :

$$v(transittime) = \sum_{i=1}^{m} (R_{i^2}) = m(\frac{m}{\mu - \lambda})^2$$

Taking p(n,m) as the probability to have "n" clients in "m" queues (k$_1$ being the number of clients in the first queue and k$_2$ the second like that ..) one obtains

$$P(n,m) = \sum_{\sum^{k_i - 1}} (1-\rho)\rho_{k^i} P(n,m) = G(n,m)(1-\rho)^m \rho^n$$

where G(n,m) is the number of permutations possible for placing "n" clients in "m" queues.

G(0,m) = 1, is the only way to position 0 client and G(n,1) = 1, is the only way to fill the queue with "n" clients

One obtains by recurrence

$$G(n,m) = \sum_{\sum_{n}^{k_i=n}} = \sum_{\widehat{n}\epsilon S(n,m)} 1$$

$$G(n,m) = \sum_{k=0}^{n} (\sum_{\widehat{n}\epsilon S(n_k,m-1)} 1)$$

$$G(n,m) = \sum_{k=0}^{n} (G(n-k,m-1))$$

If we take a random variable "X" equal to the sum of independent exponential variables "r", each with an average "M", the "X" follows and Erlang distribution with the following properties:

$$E[X] = rM$$

$$Var[X] = rM^2; where M = \frac{1}{\mu}$$

The function for Erlang distribution function is:

$$F_{M,r}(x) = \begin{cases} 1 - \sum_{k=0}^{r-1} (\frac{\frac{x}{M}}{K!} e^{\frac{x}{M}}) & \text{if x} \geq 0 \\ 0 & \text{if x} < 0 \end{cases}$$

The probability density function is:

$$f_{M,r}(x) = \begin{cases} \frac{1}{M} \frac{(\frac{x}{M})^{r-1}}{(r-1)!} & \text{if x} \geq 0 \\ 0 & \text{if x} < 0 \end{cases}$$

The probability density response time is equal to:

$$\rho = \sum_{n=0}^{\infty} (G(n,m)\rho^n (1-\rho)^m f_{M,n}(x))$$

As we increase the size of the queue, this distribution tends to be a gaussian (Fig:3.20)

We validated the delay model by simulation in NS-2. The simulation for the delay had the following characteristics: The flow rate is 10mb and the propagation delay is 2ms. The size of the packet is 250 bytes and the delay is around 3.6ms. The results obtained by simulation also confirms that the delay distribution approaches a gaussian. The hidden Markov chain composed of three states and the gaussians were centered around 0.6, 0.2 and 0.375 seconds (Fig:3.21)

It is necessary to compare resulting models using goodness of fit criteria. Well known examples of such model selections are AIC or BIC. We used EM algorithm on the simulation

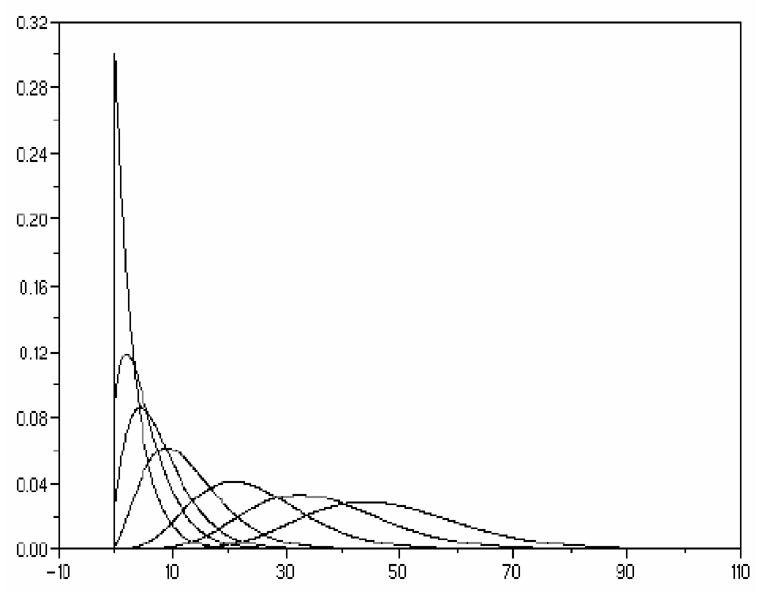


Figure 3.20. Erlang Distribution for 1,2,3,5,10,15 and 20 queues

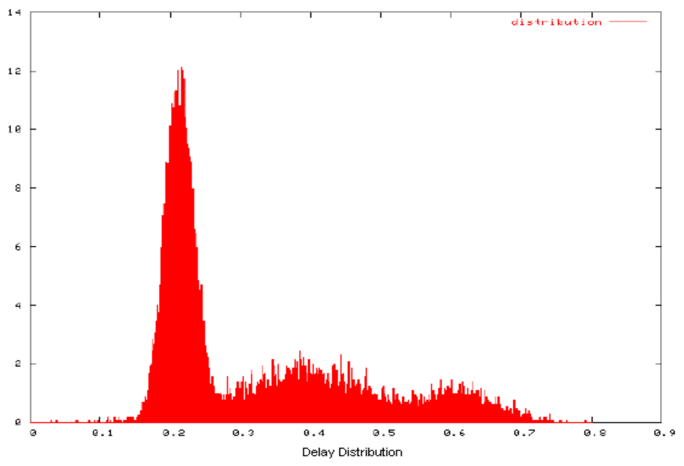


Figure 3.21. Delay distribution using Simulation

Number of States	**Maximum Likelihood**	**AIC**	**BIC**
1	0.530902	1.061928	1.062400
2	1.459564	2.919372	2.920316
3	1.497231	2.994830	2.996246
4	1.499035	2.998560	3.000448

Table 3.9. Initial Likelihood Evolution using EM

results until we get the Maximum likelihood. The EM algorithm used on the HMM for each increasing state is as follows:

We decided to apply the Vitterbi algorithm on hidden Markov chain generated by the EM algorithm. On observing the values in the table 3.9, we see that an evolution of likelihood is seen in the third state and the results for the three states are identical.

Number of States	**Maximum Likelihood**	**AIC**	**BIC**
1	-0.311432	-0.622754	-0.622320
2	-0.311246	-0.622269	-0.621403
3	-0.311238	-0.622143	-0.620843
4	-0.311221	-0.621997	-0.620264
5	-0.311218	-0.621880	-0.619714

Table 3.10. Initial Likelihood after applying vitterbi

3.4.10 Measuring global DNS response time

In the sub section(3.4.3) we got the response time and loss between the local cache server, and each of the authoritative servers represented in the French ENUM model. In this section we measure the total response time. This total response time represented as *global response time* in our work will be used to validate the simulation model which is explained in chapter 4.

Global response time is the sum of the response time of all the local DNS servers. In our measurements it can be of three different types:

1. When there is no information about the query in the cache, then the local cache nameserver may have to query all the three target servers (Tier-1, Tier-2 chunk and Tier-2 number). Then the global response time is the sum of the local response time of the three target servers.

2. When the SLD is already in the cache, then the local cache nameserver has to query only Tier-2 Bloc and Tier-2 Number servers for the TSP and the NAPTR RR. So the global response time is the sum of the local response time of these two target servers.

3. When the TSP is already updated in the local cache nameserver, then the global response time is the sum of the local response time of only the Tier-2 Number server.

For all the above three types the response time also includes the time between the resolver and the local cache nameserver.

The global response time are more influenced by the network uncertainties than the local response time. The number of hops are really a major influence. For each hop the response time also increases proportionally. The parameters influenced here are the waiting time, service time and also the propagation delay.

Propagation delay can be influenced by geographic distance. The farther is the distance between each hop, the worse the response time. Packet loss and congestion also compounds the problem.

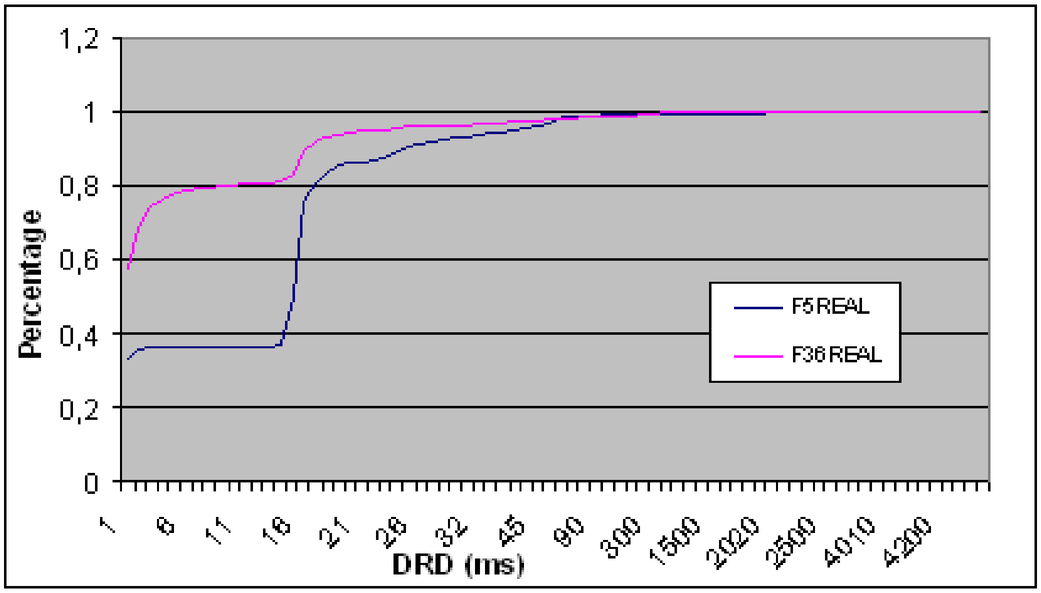


Figure 3.22. Global Response Time Cumulative Distribution Function

Measurements

While the stress tests of the previous subsection(3.4.2) was proceeding we calculated the global response time for the user's request. We calculated the cumulative distribution function for a range of 5000 and 36000 different queries(figure:3.22). The experiments were conducted for a period of one hour. These results has been used to compare with the simulated global response time to validate the model.

Analyze

In experiments conducted we found the response times are rather short; in the worst cases in the INT cache servers 97.3% of the requests are treated in less than 40ms. When the population (of different E.164 numbers, therefore size of the group of numbers) is rather large its true the global response time will be high. This is the case when the range of numbers is 36000. Indeed when the population is small (5000 different queries) the cache server is more frequently seen requiring information which is in its cache (the rate of request in a second being the same one, but the time taken for address resolution is decreased).

There is another cause for the very low global response time. The INT cache server is quite fast because of it high connectivity rate and configuration. This can be clearly noted because the number of requests for which there has been a *hit* is treated in 1ms. The number of transmissions occurred due to loss does not make a strong impact here in the calculation of the global response time since the load on this server is less and most of the losses can be caused only in the network link. But since the network connectivity is good between the cache and the authoritative servers, the impact has been minimum.

3.5 Summary

This chapter explains the analysis phase. Initially we analyse the impact different software and hardware configurations under varying load on the classical DNS. Then to analyse the ENUM case, real measurements were made on the French ENUM model which has a three tiered architecture. Mathematical models were obtained from these real measurements. The mathematical models were validated. The parameters obtained from these mathematical models were needed to verify and validate the simulation model that is developed and built in chapter 4.

Chapter 4

Simulation Model Implementation and Verification

Everything's got a moral, if only you can find it
- *Alice in Wonderland, Chapter 9, by Lewis Caroll*

In chapter 3, we measured the French ENUM empirical model. From these measurements we developed mathematical models and obtained parameters from them so that it could be used in the simulation model, which we are going to discuss in this chapter. In this chapter we start with the motivation for building an autonomic simulation model. In section 4.1 we discuss about how an autonomous simulation tool can be helpful to study and optimize the performance of ENUM applications. In section 4.2 an overview of autonomic computing is provided.

To build a Simulation model [85] we need to statistically generate the entities or nodes and contents and to develop a data structure to hold the information about them. In Section 4.3 the considerations taken into account for building the ENUM simulation model, so as to include different scenarios and designs other than the current one are explained. The implementation of the ENUM simulation platform in the existing NS-2 tool is described in section 4.4 and in section 4.5 the justification of how the simulation model that is built is autonomous is provided.

A Simulation model is considered as a complement of experimentation with the actual system. The system can be an existing or a proposed one. A simulation model which is to be developed should be a close approximation of the actual system; otherwise any conclusions derived from the model are likely to be erroneous. According to [55] validation should and can be done for all models, regardless of whether the corresponding system exists in some form or whether it will be built in future. Validation is thus the process of determining whether a simulation model is an accurate representation of the system, for the particular objectives of the study. The validation part is explained in section 4.6

4.1 Reasons for Building an Autonomous Simulation Model

When this research work on ENUM started, the goal was to build a simulation model, which can be used to conduct different experiments (varying the configuration or the delegation model) and from these experiments conclude an optimized ENUM model (with regarding to global response time) for a particular scenario. In this section we provide reasons why we need to build a simulation model, and in addition to that, why this model should be autonomic.

4.1.1 Different Types of ENUM

The initial research was based on a Public ENUM model (also known as user ENUM) where any user can update his information on the database and any other user can access this information with the help of the former's E.164 number. Even though the public ENUM standard has been there for some time it has not obtained the momentum as expected. Reasons for this limited use of Public ENUM can be ascertained due to the following reasons:

- In the case of any user updating the ENUM database, operators do not have any control over the destination of E.164 numbers. Due to this reason they are reluctant to use the information from the Public ENUM database. There is no benefit of putting a E.164 number in the Public ENUM database, if none of the operators is using the database.
- Privacy is another major problem. There is the risk that Public ENUM will become the ideal SPAM database, because querying this Public ENUM gives user's contact information. This information can be misused by malicious organisations/persons.
- Policy issues are also a consideration. Since ENUM is dependant on national (a government based monopoly on the Tier-1 level) and International (on who controls the Tier-0) bureaucracies, Public ENUM implementation of ENUM is an issue.

Due to the above mentioned issues on Public ENUM, ENUM implementations have moved into other flavors. They are called as *Infrastructure* or *Carrier* ENUM. In an *Infrastructure* ENUM, only an operator is able to alter the information in the ENUM records. The Infrastructure ENUM can be with a *private DNS* where only operators are able to query

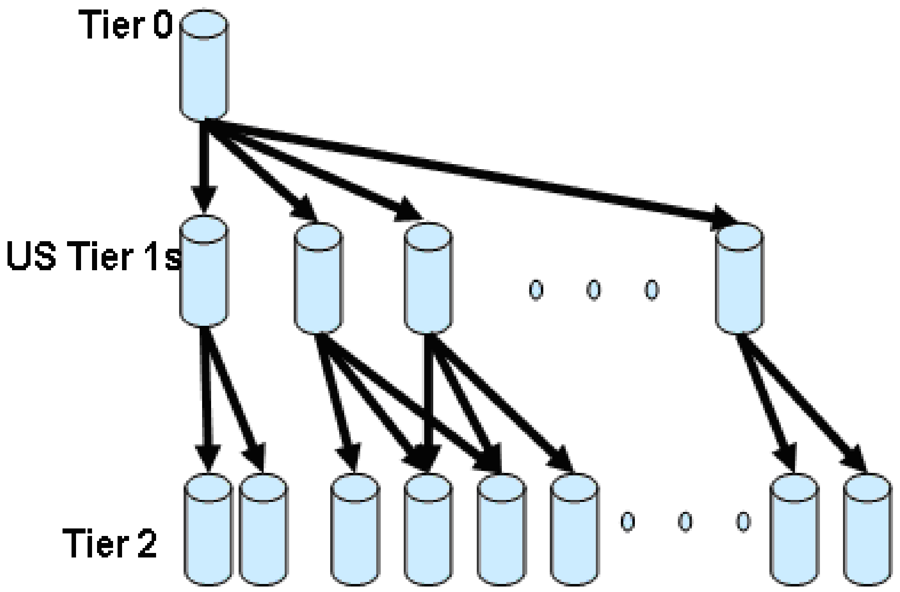


Figure 4.1. US ENUM Delegation Model

the ENUM database which is placed in a private DNS. There is also an open form of infrastructure ENUM which is placed within the *Public DNS*. In this case anyone with access to the Internet is able to query the DNS database, but only operators can manipulate the database. More information on Infrastructure ENUM can be obtained from [61]

4.1.2 Delegation Models

ENUM can also be of different delegation models. For e.g. the United States uses a delegation model (fig:4.1) with one or multiple ENUM Tier1 registries and one or multiple ENUM Tier2 providers, whereas countries like France, Sweden and Australia use a single Tier1 registry with Tier2 having both the roles of name server provider and registrar. The model that we had used for our measurements as seen in chapter 3 is of the French ENUM model with Tier-1 registry, Tier-2 registrar and Tier-2 provider. *Registry* is the manager for the national ENUM zone (For France it is 3.3.e164.arpa. and is managed by AFNIC). *Registrar* provides ENUM registration services to ENUM Applicants and *providers* are the one who give ENUM services. The ENUM delegation architecture can also vary between Public and Infrastructure ENUM.

4.1.3 ENUM Applications

Apart from the architectural point of view we will now look at what services and applications are possible with ENUM. With ENUM it is possible to register numerous services, and

VoIP being the most important. It can be used for other services including fax, email, instant messaging etc. It can also be used with IP Multimedia Subsystem IMS where Infrastructure ENUM can be used to interconnect between IMS networks. Other basic services such as services providing information resources, commercial advertisements, location information etc. can take advantage of ENUM [18].

4.1.4 Why Autonomous

ENUM cannot be viewed as a technology that is static. Like DNS it is also evolving, maybe more than DNS. Initially it started as a convention to address phone number resolution in the IP domain. Then due to policy and security issues public and infrastructure divide of ENUM came in place. Around 2004 mostly in Europe when initial deployments of ENUM started, ideas started popping about how ENUM can be used for providing different services as explained in subsection 4.1.3, which can take advantage of ENUM. To quote from [94] "*ENUM community just knows a fraction of features we might see for ENUM in future*".

Building a simulation model to study the French ENUM scenario as reported in chapter 3, it is quite enough that we build a static model and improve the response time by using different techniques, changing the simulation code wherever needed. But on viewing the conditions that ENUM can be used in different possibilities, a static simulation model will be short sighted.

Here are some arguments we put forth in building an Autonomic Simulation model:

- From an holistic point of view, NGN, encompassing a multitude of different technologies and diverse services needs a system to study and find solutions to increase QoS, while alleviating the management cost and operational complexity. This system should manage itself in a reliable way keeping in mind its objective, i.e performance. For this purpose we need autonomous systems and in considering the factor of management cost and operational complexity we need *autonomic simulation models.*
- From an *architecture* point of view based on ENUM, it is difficult to test the different real delegation models, because it involves changing the contents of the databases in the servers that form the Zone. With a simulation model, the configuration files of the Zone, and also all the data relative to the structure of the DNS tree, are concentrated in one machine. Hence it becomes quite easy to change the configurations to test different delegation models. In a real environment the bandwidth can vary among different systems connected to the DNS. Using a simulation model it is possible to test with different bandwidths according to our convenience. Also in the same manner a simulation model enables to multiply the number of servers without any additional cost, to see whether an increase in a resource can improve the performance. The simulation model should be able to get all the input details from the configuration files. There should not be any need of changing the software code to test different models.

- The simulation model should be able to *incorporate new designs* as plug-ins to the existing model without doing any major modification to the existing model.

- Depending on the *performance criteria* set, the simulation model should be able to test varyingly and come to an optimized set of metrics rather than doing a new simulation every time.

- As we explained in chapter 3, failure of DNS is always possible due to SPAM, DDoS threats etc., so to study such scenarios introducing dynamic failure and recovery, the simulation model should be autonomic.

According to our knowledge, there has not been any other work done for ENUM like the present study. We would like our simulation model, to become a platform to experiment with different ENUM scenarios, and in addition, to be a test platform for any technology which uses DNS for mapping between different network infrastructures.

4.2 Autonomic Computing

To get an idea how an autonomic simulation tool should behave, we look into the inventors of this concept. In 2001 IBM launched the Autonomic Computing vision. It is defined as "*Computing Systems that can manage themselves given high-level objectives from administrators*". Autonomic Computing includes the following four [53] features as shown in the

Self-
Configuring
Self-
Healing
Self-
Optimizing
Self-
Protecting

Figure 4.2. IBM Autonomic Computing vision

figure 4.2:

1. Self-configuring

2. Self-healing

3. Self-optimizing and
4. Self-protecting.

Self-configuring involves automated configuration of components and systems following high-level policies. *Self-healing* can be accomplished by automatically detecting, diagnosing and repairing localized software and hardware problems. *Self-optimizing* involves self-tuning of service parameters. *Self-protecting* means that the system automatically defends against malicious attacks or cascading failure. It uses early warnings to anticipate and prevent system-wide failure.

The architecture of autonomic computing contains two major components; *autonomic managers* and *managed elements*. An autonomic manager is an agent managing its internal behavior and relationships with other agents according to prescribed policies. The managed elements can be a hardware resource such as CPU, or application software such as a database server or an entire system.

IBM has conducted research towards autonomic computing, across all levels of computer management from hardware to software. On the hardware level, systems are dynamically reconfigurable and upgradeable, enabling the movement of hardware resources (such as processors, memory and I/O slots) without requiring reboots [46]. On an Operating System level, a method called hot-swapping [45] is used where an active Operating System allows monitoring code, diagnostic code and function implementations to be dynamically inserted and removed in the system. On the application level database validate the optimizers by itself and web servers are dynamically reconfigured by agents to adapt service performance [95].

From the above examples, we see that some of the ideas from autonomic computing have already been implemented in practice. In our thesis work, we seek to put some of the autonomic computing ideas into the ENUM based simulation model.

4.3 Design Considerations

The design of the model impacts all aspects of simulation study, in particular, the data requirements, the validity of the model and the confidence that is placed on the model results[55]. Law et al[55] proposes a seven-step approach for conducting a successful simulation study. Raj Jain in his book [50] discusses about common mistakes in simulation which he summarizes in eight points. From this two well referenced authors in the field of simulation and modeling we take certain constructive lessons which we use to follow in building the ENUM simulation model.

Simulation model need not completely mimic the real system. Abstractions can be made to make the model simpler. It is generally assumed that a more detailed model is a better model, since it makes fewer assumptions. This need not be true in all the cases. In certain cases a detailed model may require more detailed knowledge of input parameters, which if not available, may make the model inaccurate.

For building an ENUM simulation model we adopted the detailed approach. The conditions that influenced us to take the detailed approach are as follows:

- Our objective to build a simulation model was clear. The overall objective is that the simulation model built should be a surrogate of the real ENUM implementation. The specific objective is that the model should study the global response time of ENUM applications. The scope of the model is that it should be used to study the ENUM global response time performance either by changing the TTL's, or optimization of the algorithms used by the entities in the model. It should also be used to study various ENUM delegation models and to come up with an optimal delegation model for a particular ENUM implementation. The scope of this model is also to study different ENUM enabled services which can be done on the existing model with minimum changes.
- The metrics which we were interested in studying with the simulation model were known. The key metric was global response time.
- From chapter 3 we know the detailed knowledge of input parameters:
    1. Random number generators (from real measurement)
    2. Internet Link metrics
    3. Delay at the cache and authoritative servers
    4. Time for the simulation (5000 and 36000)
- The DNS which makes the heart of the ENUM system is a well studied architecture. We had detailed information about the system layout and the operating procedures of the DNS system.
- We had the output data from the real system to use for model validation.

So far we explained about the approach that we took to build the simulation model. In the forthcoming subsections we explain why we chose NS-2 for our work and also about the configuration and functional aspects of the simulation model.

4.3.1 Need for an existing Simulation Tool

The choice of the tool/language used to build the conceptual model has a significant impact either on the timely development of the model or over the run time and efficiency of the simulation [55].

Building a simulation model using a general purpose language like C, C++ and java is chosen primarily because it allows building a simulation perfectly tailored to the needs of the study. The simulation model thus built will have the highest possible performance level (e.g. in terms of consumed run time). Usually, the choice of this approach is motivated by the extensive use of a program that would be otherwise prohibitively slow. Since general purpose languages are not oriented towards simulation tasks, one has to program the whole description of the system in its finest details, and also the description of all simulation related tasks.

On the other hand simulation tools could also be used to build the model. By simulation tool, it is meant not only the basic tool which runs the simulation model, but also all related utility programs such as graphical aids and development tools which help to build and visualize the simulation model being developed [38]. Library of data structures, routines and algorithms are already available in the simulation tool, which could be used as needed. For example if the simulation model used UDP for data communication, the user need not write code for the UDP subroutine, rather he can just call the UDP library present in the tool. This saves time and also building a model using a simulation tool is simple. The drawback of a simulation tool in respect to a general purpose language is that it cannot be as efficient as the model developed by the later, since the program is tailored to the objectives of the model and is not as heavy as model built using simulation tool.

There are different simulation tools available. General purpose simulation tools such as OPNET and NS-2 are used, where the simulation model can be built over the existing simulation tool with few mouse clicks or by writing down pieces of code and adding to the existing tool through an interface. In this case most of the time, the simulation tool itself provides utilities to visualize the results and even to produce a report.

Programming a detailed ENUM simulation model using a general purpose language is a difficult task. Pieces of code for all the modules in the network layer should be written. The pragmatic approach was to develop the ENUM simulation model using general purpose simulation tools.

We planned to use NS-2 because it is free, it is easy to add new components to the existing NS-2 simulator and it has also complementary tools for visualizing the simulation such as Network Animator (nam), X-graph for plotting graph and topology generator for creating topologies. Another important feature of NS-2 is that the trade off in efficiency caused by using a general purpose simulation tool is reduced since the tool is written in C++.

4.3.2 Modules Used in the Simulation Model

In order to be able to simulate ENUM system, we need a certain amount of information about the different modules that has to be developed to create an ENUM system. In this sub section we start with explaining the basic topology used and then we give an explanation for the different modules used (the topology resembles the empirical measurements that we have already made on the real ENUM model; (in chapter3)).

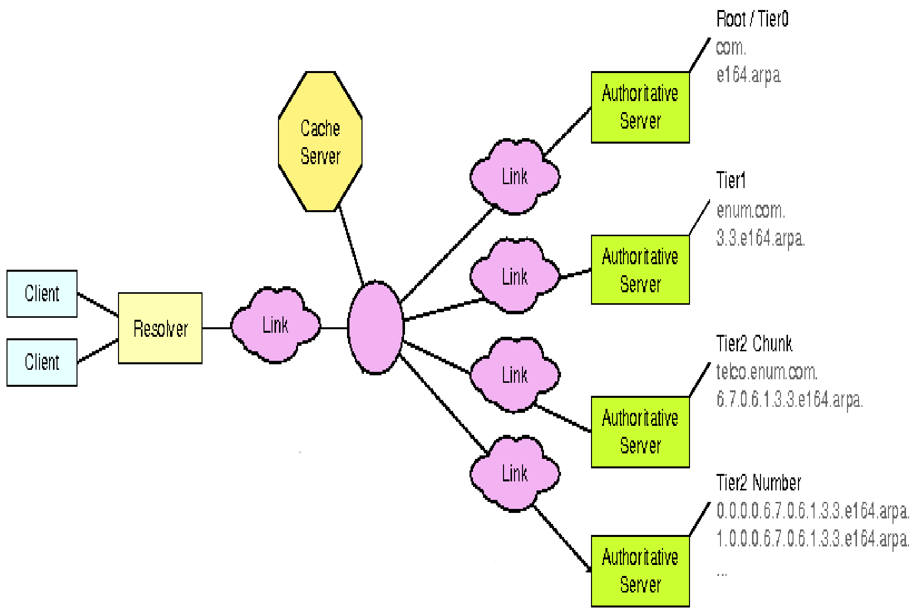


Figure 4.3. Simulation Topology

ENUM Client

In the topology (figure 4.3) we have two ENUM clients. The ENUM client gets data from a configuration file and sends a E.164 type query (0.0.0.0.6.7.0.6.1.3.3.e164.arpa.) to the resolver. Normally it will also retrieve NAPTR records containing URIs related to the E.164 number. Since the objective is to calculate global response time, the response packet is stopped at the resolver.

It is used to generate and send E.164 type queries to the resolver and retrieve NAPTR records containing URIs related to the E.164 numbers.

Resolver

The resolver module receives request from the ENUM client and creates a DNS query type packet with the data obtained from the client. The resolver makes a recursive query to the cache server. The burden of finding the answer to the query from the resolver is placed on the cache server.

Adding more than one resolver is quite easy with the help of the simulation model. The resolver and the client facilitate to load the cache server on a wide range.

Cache Name Server

Most of the DNS resolution is processed by the cache server module. It receives recursive queries from the resolver. If the information asked by the query is present in the cache's database then there's a *cache hit* and it can answer directly to the resolver; otherwise in case of a *cache miss* it has to send iterative queries to authoritative server(s) before it can answer to the resolver. Since it knows only the name and place of the root server(s) in the beginning, it has to send a few iterative queries until it finds the response for the resolve's query (high number of hops). But the process is shortened (fewer hops) when the cache get populated over time as data is stored in the cache database for a TTL period.

Authoritative Name Server

The authoritative server(s) receives the query from the cache server and searches for the query in its hash table. If it has the response, then it replies, otherwise it responds with information about the server which can possibly have the requested information.

4.3.3 Configuration Files

In this subsection we explain about the data structure that is used to generate, transfer, process or look up information. In order to have an autonomous simulation model, we designed all the data structure to be obtained from configuration files. As explained in [53], the configuration files will serve as the *Autonomic Manager* for the simulation model. This will help any future users to use the simulation by just modifying the configuration files for any scenario without understanding the complexity of the simulation code. The DNS software that we have used for the real measurements in chapter 3 (explained in 3.3.1) is BIND. The BIND DNS parameters are used to develop the ENUM simulation model. All the configuration files which are used in this simulation model are real BIND configurations, without any abstractions.

In the figure 4.4 a functional representation of the different configuration files used by each of the modules in the simulation model are shown, in order to have an understanding of how the network nodes process this files and get information. The contents of certain configuration files are represented in the figure4.4 indicated by dotted lines (due to space constrains all the data could not be displayed in the figure). For example the resolver module uses only the resolv.conf file and its contents(such as $n(6) and n(7)$) are the data to identify which cache server to connect to. In the case of modules, like the authoritative servers, use more then one configuration files(Tier-2 chunk uses three files named.conf, db.0.0.1.4.6.3.3.e164.arpa. and db.sfr.numerobis.prd.fr.). So in this case it may be needed to get information from one of the configuration files (named. conf) and using the value from this file access the remaining of any of the file present.

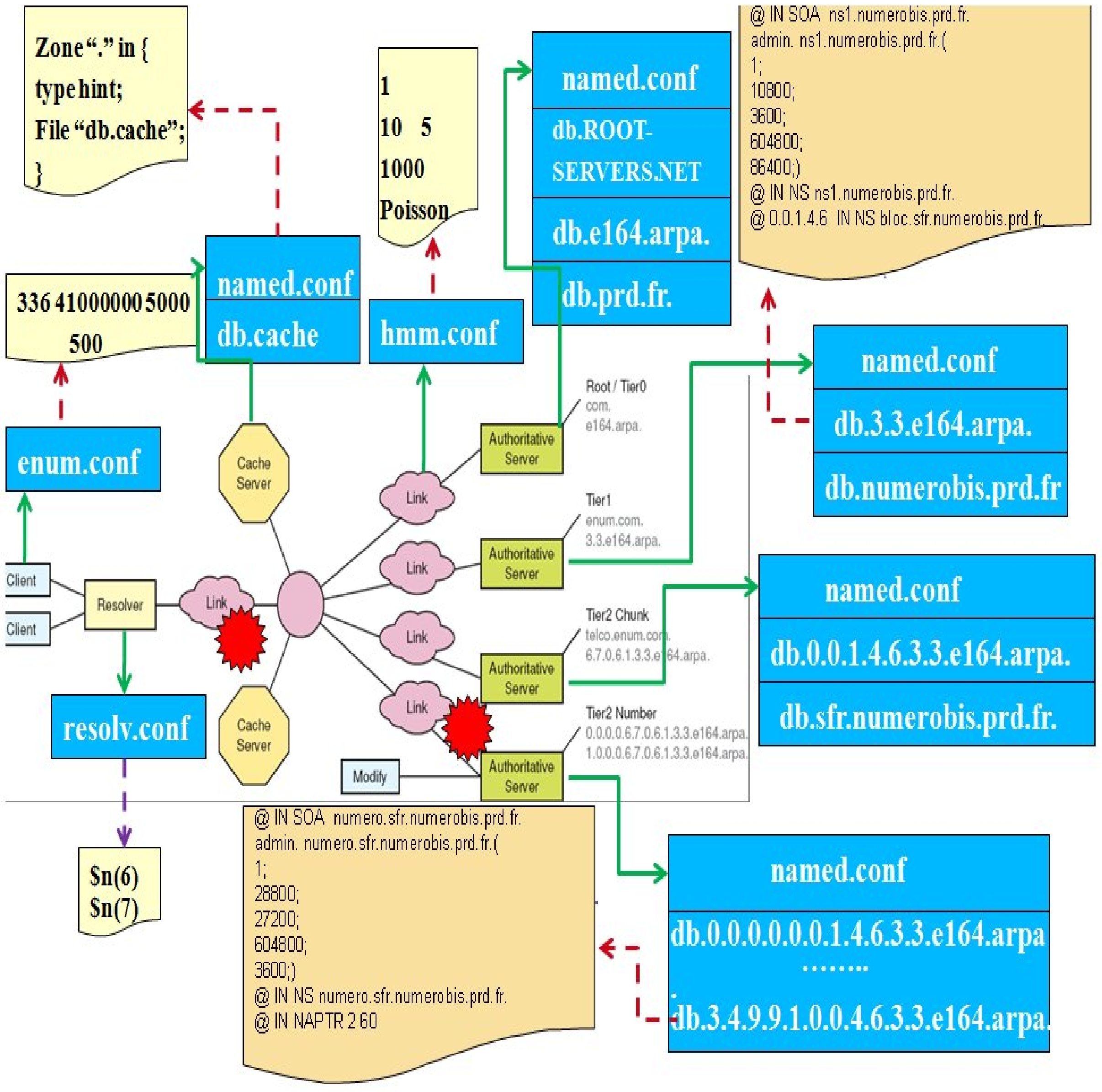


Figure 4.4. Configuration File Explanation

The configuration tree structure for the simulation model (figure 4.5) is classified under five categories for each of the different modules to be implemented:

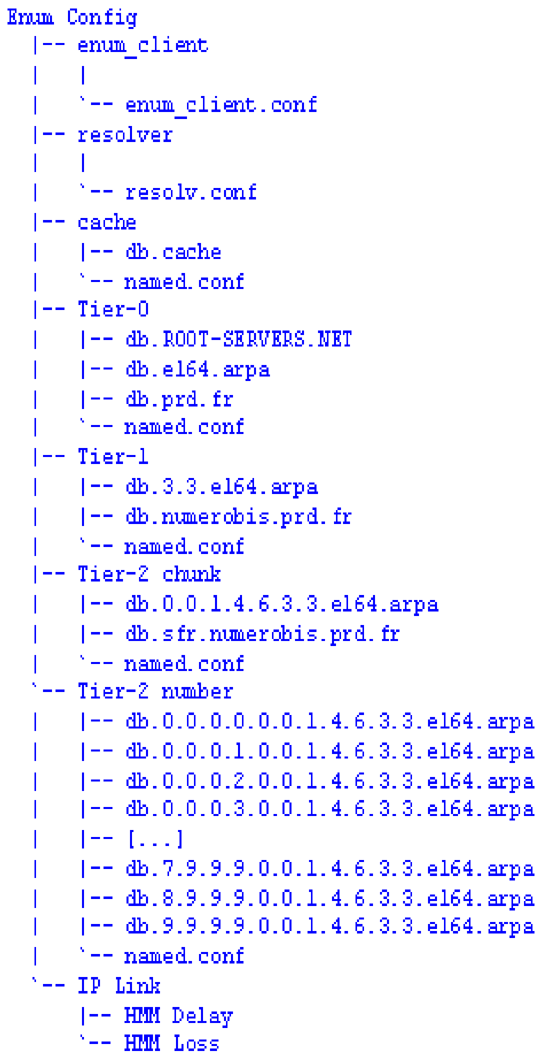
1. ENUM client

2. Resolver

3. Cache Name Server

4. Authoritative Name Server and

5. IP Link



```
Enum Config
  |-- enum_client
  |   |
  |   `-- enum_client.conf
  |-- resolver
  |   |
  |   `-- resolv.conf
  |-- cache
  |   |-- db.cache
  |   `-- named.conf
  |-- Tier-0
  |   |-- db.ROOT-SERVERS.NET
  |   |-- db.e164.arpa
  |   |-- db.prd.fr
  |   `-- named.conf
  |-- Tier-1
  |   |-- db.3.3.e164.arpa
  |   |-- db.numerobis.prd.fr
  |   `-- named.conf
  |-- Tier-2 chunk
  |   |-- db.0.0.1.4.6.3.3.e164.arpa
  |   |-- db.sfr.numerobis.prd.fr
  |   `-- named.conf
  `-- Tier-2 number
  |   |-- db.0.0.0.0.0.0.1.4.6.3.3.e164.arpa
  |   |-- db.0.0.0.1.0.0.1.4.6.3.3.e164.arpa
  |   |-- db.0.0.0.2.0.0.1.4.6.3.3.e164.arpa
  |   |-- db.0.0.0.3.0.0.1.4.6.3.3.e164.arpa
  |   |-- [...]
  |   |-- db.7.9.9.9.0.0.1.4.6.3.3.e164.arpa
  |   |-- db.8.9.9.9.0.0.1.4.6.3.3.e164.arpa
  |   |-- db.9.9.9.9.0.0.1.4.6.3.3.e164.arpa
  |   `-- named.conf
  `-- IP Link
      |-- HMM Delay
      `-- HMM Loss
```

Figure 4.5. Configuration File Tree Structure of the Simulation Model

ENUM client In our initial simulation implementation we have two enum clients. Each one will have two different configuration files enum1.conf and enum2.conf. Each of them will generate different queries of e.164 type. A detailed explanation of how it works is given in subsection 4.4.2.

Resolver As we already explained, *resolver* represents the module which formulates a DNS query packet type and send it to the local DNS cache server. The configuration file "resolv.conf" file contains information about the cache servers connected to the resolver. In the resolv.conf shown in the figure 4.4, there are two cache servers ($n(6) and $n(7) which represents nodes in NS-2) connected to the resolver. The algorithm used to identify which cache server to connect is explained in subsection 4.4.3

Cache Name Server In real BIND set up it is a process called "named" in the cache name server which answers queries from resolvers. This application reads its data from a configuration file called "named.conf" which in turn gets its information from the zone files. Several zone files can exist but one zone file in particular keeps a database of records that supply the named process with most of its answers.

In our simulation set up, when the cache name server gets a query from the resolver, it searches the named.conf file. The named.conf file at the cache server looks as follows (also see fig: 4.4):

```
zone "." in {
type hint;
file "db.cache";
};
```

According to the named.conf configuration file above, the cache server is of type *hint*. This type of servers usually will store a local cache of host name and address mappings. If a requested address or host name is not in its cache, the hint server will contact the master name server (in our case it is Tier-0), get the resolution information and add it to its cache. The zone file "db.cache" has information on how to contact only the root server. So with this configuration at the beginning all queries are redirected to the root server.

The zone statement identifies the location of the hint files which contain the name and address of the root servers on the internet. The zone file db.cache contains the following lines:

```
.                      360000  NS  A.ROOT-SERVERS.NET.
A.ROOT-SERVERS.NET.    360000  A   $n(2)
```

The first line indicates the Name Server (NS) resource records. This record indicates that there is one NS for the root domain "." i.e. A.ROOT-SERVERS.NET. The second line represents the name-to-address mappings. The "A" stands for address and the resource record maps the name (A.ROOT-SERVERS.NET.) to node address ($n(2)). Instead of IP address in real scenarios node address is provided which is used as IP address in simulations.

The real value of cache name server comes only after it build up its cache. Each time it queries an authoritative name server and receives an answer, it caches the records in the answer. Over time, the cache will grow to include the information most often requested by the resolvers querying the cache name server.

Authoritative Name Server The Authoritative Name Server in the simulation environment represents Tier0, Tier1, Tier2 chunk and Tier2 number server of the French

ENUM model. The configuration file in this category has either only partly information (such as the configuration file "db.0.0.1.4.6.3.3.e164.arpa", which contains information on which server to contact if the first seven E.164 number of the query maps with .0.0.1.4.6.3.3.e164.arpa.) or complete information (such as NAPTR RRs in case of the configuration file db.0.0.0.0.0.0.1.4.6.3.3.e164.arpa).

We will take the example of Tier-1 server for understanding the configuration structure of the authoritative servers. Similar to the cache server it is the named application which reads data from "named.conf" file which in turn gets its information from the zone files. The named.conf file at the authoritative server looks like following:

```
zone "3.3.e164.arpa" in
type master;
file "db.3.3.e164.arpa";
;

    zone "numerobis.prd.fr" in
type master;
file "db.numerobis.prd.fr";
;
```

Both the servers are of type "master". They are the authoritative servers for their particular zone. They read the data from the file on its host. For example the file "db.numerobis.prd.fr" contains information for all the FQDN which ends with "numerobis.prd.fr".

The content of the file db.numerobis.prd.fr is as follows:

```
$TTL  10800
@ IN  SOA ns1.numerobis.prd.fr. admin.ns1.numerobis.prd.fr. (
      1 ; serial
      10800 ; Refresh after 3 hours
      3600 ; Retry after 1 hour
      1w ; Expire after 1 week
      1h ; Negative caching TTL of 1 hour
)
```

Their explanation is like this. The $TTL 10800 line states that records looked up and cached in a caching server from this file have a TTL of three hours. The cached entry expires after three hours and is removed from the cache when that much time has passed. The "@" sign in the line refers to the "origin" for this zone file which is "numerobis.prd.fr.". "IN" stands for Internet. SOA refers to "Start of Authority". SOA indicates that this name server is the best source of information for the data within this domain. The first name after SOA(ns1.numerobis.prd.fr.) is the primary name server of the numerobis.prd.fr zone. The second name (admin.ns1.numerobis.prd.fr.) is the mail address of the person in charge of the zone if the first "." is replaced with a @ (admin@ns1.numerobis.prd.fr.).

The NS records for the file "db.numerobis.prd.fr" is as follows:

```
@    IN  NS  ns1
sfr  IN  NS  bloc.sfr
```

These records indicate that there are two name servers for the zone numerobis.prd.fr. The name servers are the hosts ns1.numerobis.prd.fr and bloc.sfr.numerobis.prd.fr.

Now we look into the name-to-address mappings of the file "db.numerobis.prd.fr".

```
ns1       IN  A  $n(3)
bloc.sfr  IN  A  $n(4)
```

ns1.numerobis.prd.fr is mapped to node \$n(3) and bloc.sfr.numerobis.prd.fr is mapped to node \$n(4). So the query which comes under each of these zones are redirected to the respective nodes. This process is followed until the query gets the full NAPTR RRs.

4.3.4 Hash Tables

Packet classification is one of the essential tasks of network processing. In our simulation environment all the incoming packets have to be processed. For example the cache server has to separate the header and data from the packet that it has received from the resolver. The header is used to identify from which client it has received the query from and also the packet ID. In turn the data i.e. the domain name is used to look up in a hash table to check whether the response for the query is in its cache. Such look ups are done at the resolver, cache and authoritative servers in our simulation. The job of look ups had to be done in line speed to decrease the processing delay. Hash tables support one of the most efficient types of searching. Fundamentally, a hash table consists of an array in which data is accessed via a special index called *key*. Key based hashing algorithms have been proved to be a good approach. In the well known work [44], resource distribution between peers is based on certain hashing mechanism. They use a key to map it to a node and data location is done by associating a key with each data item.

The primary idea behind a hash table is to establish a mapping between a set of all possible keys and positions in the array using a *hash function*. A hash function accepts a key and returns its *hash coding* or *hash value*. Keys vary in type but hash codings are always integers.

The hash tables that we have developed in our simulator is based on the Standard Template Library STL in C++. It uses a *chained hash table* technique. The chained hash table fundamentally consists of an array of linked lists. Each list forms a *bucket* in which we place all elements hashing to a specific position in the array as seen in the figure 4.6.

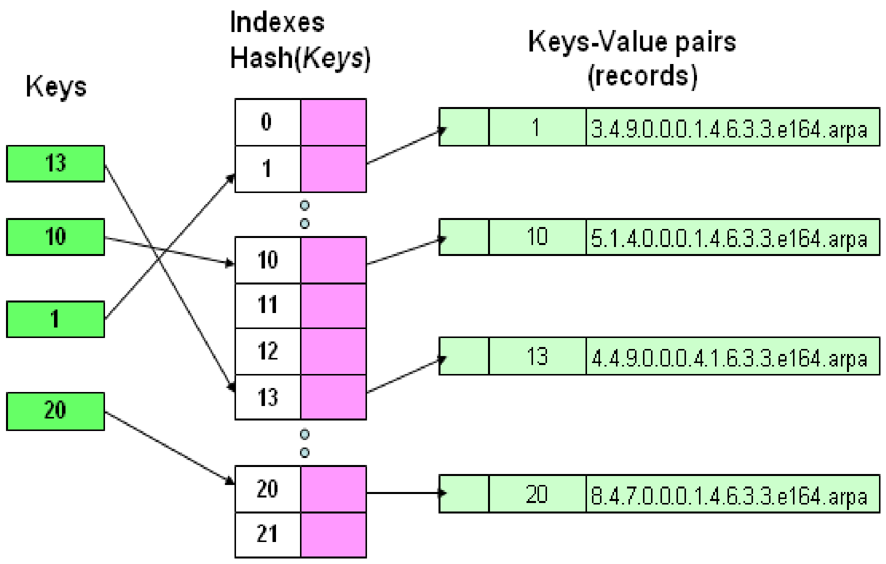

Keys
Indexes Hash(*Keys*)
Keys-Value pairs (records)
13
10
1
20
0
1
10
11
12
13
20
21
1
3.4.9.0.0.0.1.4.6.3.3.e164.arpa
10
5.1.4.0.0.0.1.4.6.3.3.e164.arpa
13
4.4.9.0.0.0.4.1.6.3.3.e164.arpa
20
8.4.7.0.0.0.1.4.6.3.3.e164.arpa

Figure 4.6. Hash Table

The bucket number is computed by taking the hash value modulo the number of buckets. To insert an element, we first pass its key to a hash function. This tells us in which bucket the element belongs to. In figure (4.6) hash value of the packet ID 1 (assuming the hash value is same as that of the ID) is stored in bucket 1(Hash value of 1 modulus 21 buckets = bucket 1). To look up or to remove an element, we hash its key again to find its bucket, then traverse the appropriate list until we find the element that we are looking for. In this case since each bucket is a linked list, the hash table is not fixed to a number of elements.

4.4 Implementation in NS-2

In NS-2 the procedure is as follows:

- The user creates the topology of the network by means of an OTcl based interpreter.
- The Internet links are specified in terms of bandwidth and of scheduling discipline.
- The routing strategy is also specified, through traditional or user-specific algorithms.
- Traffic generators are also provided or user-specific traffic generation could be used.

We will give a brief introduction on how NS-2 basic architecture works in this section. NS-2 is written in two programming languages; C++ and OTcl. In order to reduce the packet processing time all the network objects and the event scheduler are written and compiled in

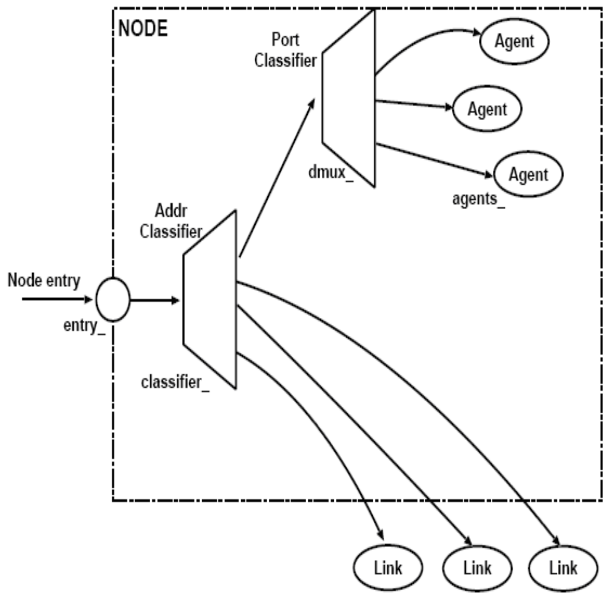


Figure 4.7. Structure of a Unicast Node

C++. For the purpose of ease the simulation script is written in OTcl. To summarize, most of the data functions are in C++ and the control functions in OTcl.

The compiled C++ objects are made available to the OTcl interpreter through an OTcl linkage that creates a matching OTcl object for each of the C++ objects. Any of the control functions and the configuration variables specified by C++ act as member functions and member variables of the corresponding OTcl object. In this way, the controls of the C++ objects are given to OTcl.

In NS-2 the physical and the data link layers are abstracted. Each entity like the source, sink and the router are created as objects, which in turn are represented as nodes. The typical structure of a unicast node is shown in the figure 4.7.

The node structure consists of two Tcl Objects, an *address* classifier and a *port* classifier. The function of these classifiers is to distribute incoming packets to the correct agent or outgoing link. The packet is sent to the port classifier if the destination address is in the node itself. It then forwards the packet to the appropriate agent depending on the port number. If the destination address of the packet is forwarded to the address classifier it in turn sends the packet to the appropriate link.

The function of a node when it receives a packet is to examine the packets fields, usually its destination address, and on occasion, its source address. It should then map the values to an outgoing interface object that is the next downstream recipient of this packet. From this we know that node performs the function of receiving and forwarding the packets.

The functionality of the transport layer of the Open Systems Interconnection (OSI) network model is performed by agents in NS-2. Agents perform packet header processing for the protocol involved. There are number of agents already implemented as library routines in NS-2 like UDP, TCP etc. NS-2 also gives the ability of adding a new agent or to modify the agents already defined in the NS-2 library. In our implementation we added new functionalities to the existing UDP agent to facilitate our experiment.

The Application layer in the OSI model is modeled by the application class in NS-2. In real-world systems, applications typically access network services through an Application Programming Interface (API). The most popular of the API is known as sockets. In NS-2 the socket API behavior is mimicked through a set of well defined API functions. These functions are then mapped to the appropriate internal agent functions. For example a call to send(numBytes) causes TCP to increment its send buffer by the corresponding number of bytes. The node, the agent and the application are hooked together by the API functions to communicate with each other

Other requirement that is needed in a simulation model is related to links between two nodes like the bandwidth of the link, the propagation delay and also the type of algorithm for packet processing at the link (e.g. RED (Random Early Detection)). An indication for example 10Mbps, indicate the bandwidth of the link, the propagation delay can also be specified by a parameter like 10ms and the algorithm can be called directly by its name. After having looked at the basic architecture of NS-2, now we will explain how we add ENUM

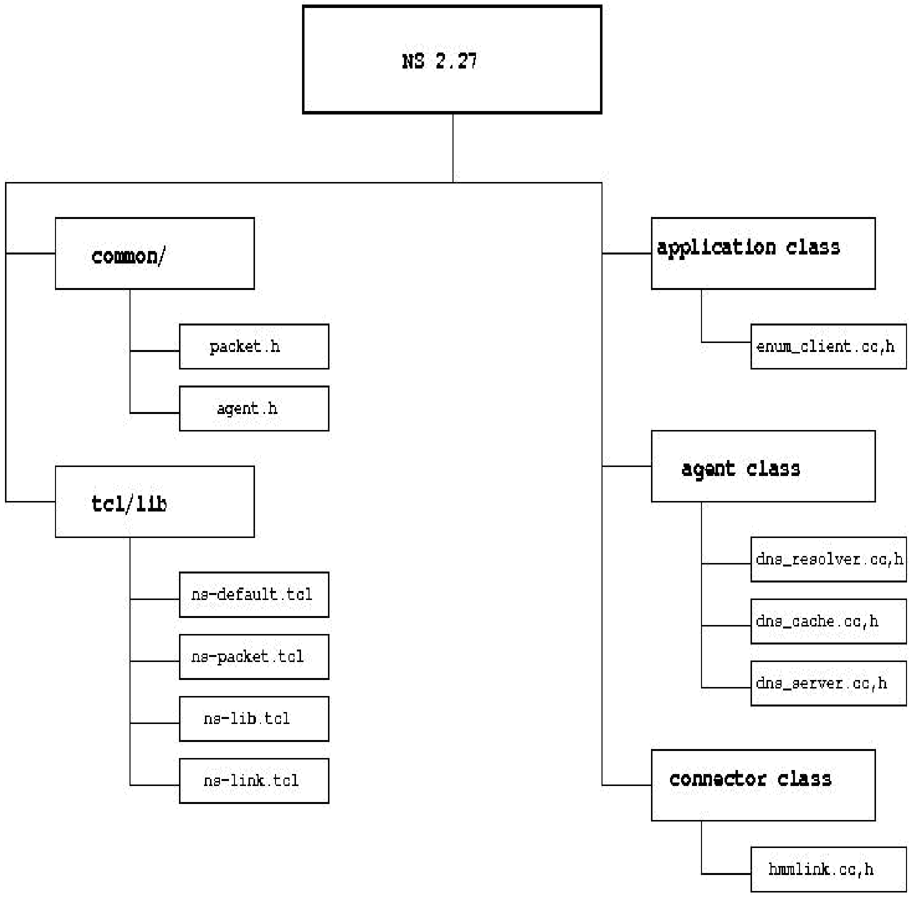


Figure 4.8. Module Implementation in NS-2

extensions to NS-2. The figure (4.8) illustrates the files that were modified (left side) and added (the right side) to integrate the new application. The modifications are done both in C++ data functions and OTcl control functions and the additions were done only in C++.

4.4.1 Modifications to NS-2

common/packet.cc,h : The class Packet defines the structure of a packet and provides member functions to handle a free list for objects of this type. Since we introduced ENUM query/response type packet, we had to describe the corresponding packet format in this class.

common/agent.cc,h : The class Agent supports packet generation and reception. To have a new application and agent running with the NS-2 distribution the agent class is modified to support the member functions of the new agents.

tcl/lib/ns-default.tcl : Default values for member variables, those visible in OTcl only and those linked between OTcl and C++ with bind are initialized in the ns-default.tcl file. Default values for the newly introduced configurable parameters are introduced in ns-default.tcl.

tcl/lib/ns-packet.tcl : To enable the new packet format used by ENUM we had to add the new packet types to ns-packet.tcl. This file is executed when the simulator initializes and creates an array containing a mapping between the class names and the names of the currently active packet header class.

tcl/lib/ns-lib.tcl : This file interprets node configurations specified in NS-2 simulation script. In order to introduce losses at the node we added an extension to accommodate them

tcl/lib/ns-link.tcl : The modifications in the link that is introducing loss and delay parameters in the queue for the IP link is done here.

In the forthcoming subsections we will see how new modules were integrated to NS-2.

4.4.2 ENUM client Implementation

NS-2 follows a class hierarchy [7] in C++ as well as in OTcl. Any new objects instantiated are inherited from the NS-2 class hierarchy. The new Enum client module added to NS-2 inherits the base class "Application". There are two basic types of applications in NS-2: *traffic generators* and *simulated applications.* The Enum client falls into the traffic generators category. This module accesses a configuration file (enum.*.conf), to generate a sequence of E.164 type numbers. The configuration file is of the format ”336 41000000 5000 500” where

- 336 41000000 represents Starting Number
- 5000 represents Size and
- 500 represents Average Inter arrival Time

From the ”starting number” enum client generates 5000 different E.164 queries and add e164.arpa with each query. Finally ”.” are introduced between each number such that a complete FQDN is generated. A packet is created to add this generated data. The packet is of following format

Packet ID	Resource Record	Domain Name
For each client it starts from ’0’ and increases by an order of ’1’. So there are two patterns of sequence number	At present it is kept at ’3’	3.4.9.0.0.0.1.4.6.3.3.e164.arpa

The time interval between each packet is also generated here. Since, we did not have real traces of ENUM, these requests were sent to the resolver at a random interval. At present this random interval is fixed to the same interval that we used in the testing of real ENUM architecture (as in chapter 3). The distribution chosen for traffic generation in the simulated model may not quite adhere to the real ENUM scenarios of traffic generation. In the future when real ENUM traces (not experimental as we got from the measurements in chapter3 but real user generated traces) are available, the configuration parameters for the time interval between two distinct packets can be modeled accordingly.

4.4.3 Resolver Implementation

In NS-2, an application is attached to an agent and the application sends data through the underlying agent or by calling a set of methods defined in the agent. Enum client is the application which generates the traffic and it needs an agent to carry the traffic to its appropriate destination as well as to receive data. Since DNS uses UDP, the simulation model also uses the same as the transport agent. UDP is already present as a library routine in NS-2.

To perform the resolver operations, a new *dns_resolver agent* is created as the sub class of the UDP agent. dns_resolver agent is necessary to create a DNS type query from the data received from the application (enum_client) and also facilitate in receiving a DNS-type response from the local cache DNS server. It is necessary to explain the different hash tables used for this module before going into the implementation details.

The hash tables are necessary to store and do the look up of corresponding information to the queries. Two different hash tables were used in this module:

- The first hash table (htable1) contains the packet sequence number as ”key” and complete DNS query packet information as ”value”. The value information is packed as a structure which contains the following fields:
  - The timeout value (at what time the same query will be called again)
  - Type of the packet (whether it is a query or response)
  - The string (e.164 string format)
  - The number of iteration (explained in the Timer algorithm.)
- The second hash table (htable2) contains the packet sequence number as ”key” and the time when the packet is send to the cache server as ”value”.

The procedure that is followed at the resolver agent is as follows (figure 4.9):

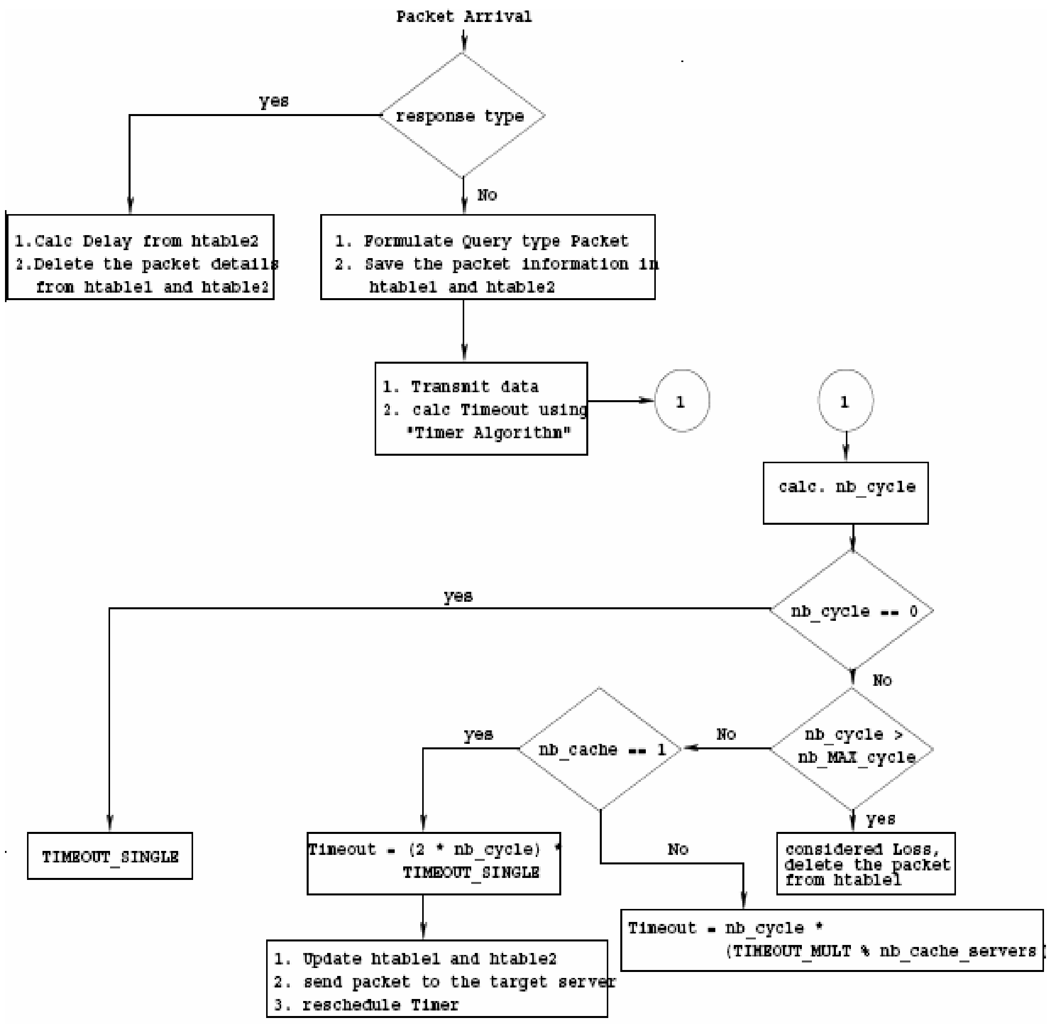

Packet Arrival
response type
yes
No
1.Calc Delay from htable2
2.Delete the packet details from htable1 and htable2
1. Formulate Query type Packet
2. Save the packet information in htable1 and htable2
1. Transmit data
2. calc Timeout using "Timer Algorithm"
1
1
calc. nb_cycle
nb_cycle == 0
yes
No
nb_cycle > nb_MAX_cycle
No
yes
nb_cache == 1
yes
No
TIMEOUT_SINGLE
Timeout = (2 * nb_cycle) * TIMEOUT_SINGLE
considered Loss, delete the packet from htable1
Timeout = nb_cycle * (TIMEOUT_MULT % nb_cache_servers)
1. Update htable1 and htable2
2. send packet to the target server
3. reschedule Timer

Figure 4.9. Implementation of resolver in NS-2

A new Packet arrives On receiving a packet, the type of the packet is identified to check, whether it is a "response type" or "query type" packet.

Response type packet In this case the packet contains a response for the query it requested. The sequence number of the packet is parsed from the received packet and the sequence number is searched in the htable1.

- If it is found then the global response time is calculated from subtracting the value of the received time - send time. The send time data is obtained by looking at the second hash table (htable2) with the sequence number of the received packet.
- If the sequence number is not found, packet is ignored since the response has been received already and the values are deleted from the hash tables.

Query type If the packet received is not of the response type, then it is from the enum_client. In this case, a new packet is formulated with the packet id as increasing sequence number (which is unique), the URI (the E.164 identifier is converted to URI (as in subsection 2.5.2)) and a field for calculating the number of hops. The hop value increases with every new node the particular packet traverses. The time that the packet is sent is updated in htable2. The formulated packet is sent to the corresponding cache server.

While instantiating the dns_resolver object, the number of cache servers (nb_cacheservers) that is accessible by the resolver is obtained from the resolv.conf configuration file [Fig. 4.4]. If a new cache server is added, the resolv.conf file is updated.

Once the query is sent to the cache server a timeout is calculated for the query (in case the packet gets lost or delayed). After the timeout, if a response for the query is not received, the query is resent to the same or different cache server. The timeout is calculated by the Timer Algorithm.

Timer Algorithm Every time a request is sent from the resolver to the cache server, it is calculated as iteration (nb_iteration). If there are two cache servers and when a new query is generated from the resolver, it is the first iteration. Once the same query is sent to all the different accessible cache servers it is calculated as one cycle (nb_cycle). We used the BIND configurations for fixing the number of maximum servers (NB_MAX_SERVERS) to 3 and number of maximum cycles (NB_MAX_CYCLES) to 4.

By using the algorithm, timeouts and the target server to which the query packet is to be sent are calculated depending on the number of cache servers, iterations, and number of cycles. After every time the timer algorithm is called the nb_iteration is updated in the hash table (htable1) for the particular packet.

```
   input  : TIMEOUT_SINGLE = 5 (in seconds);
   TIMEOUT_MULTIPLE = 10 (n seconds);
   NB_MAX_CYCLES = 3;
   NB_MAX_SERVERS = 4;
   output: The TTL value and the target local cache server to send the data
1  nb_cycle = iteration/nb_cacheservers (Convert the result to Integer Value);
2  if nb_cycle == 0 then
3      Timeout = TIMEOUT_SINGLE;
4      Target Server = Iteration mod nb_cacheservers;
5  if nb_cycles > NB_MAX_CYCLES then
6      The packet is considered lost;
7  if nb_cycle > 0 then
8      if nb_cacheservers == 1 then
9          TIMEOUT = 2*nb_cycle*TIMEOUT_SINGLE;
10         Target Server = Iteration mod nb_cacheservers;
11     else
12         TIMEOUT = nb_cycle*TIMEOUT_MULTI*nb_cacheservers;
13         Target Server = Iteration mod nb_cacheservers;
14     end
```
Algorithm 1: Algorithm for calculating Timeout

4.4.4 Cache Server Implementation

As discussed in the previous subsection here also it is necessary to explain the hash tables present in this module before going into the implementation part. There are three hash tables used by the cache server module:

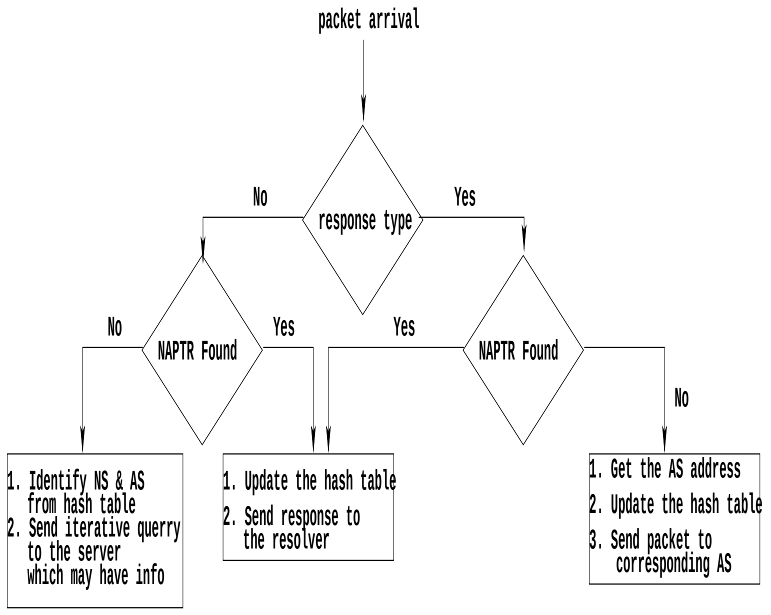

packet arrival
response type
No
Yes
NAPTR Found
No
Yes
Yes
NAPTR Found
No
1. Identify NS & AS from hash table
2. Send iterative querry to the server which may have info
1. Update the hash table
2. Send response to the resolver
1. Get the AS address
2. Update the hash table
3. Send packet to corresponding AS

Figure 4.10. Implementation of Cache Server in NS-2

- Name server (NS) Hash table - to give information about a segment of the database
- Authoritative server (AS) Hash table - to store the information pertaining to a particular domain, exclusive of any sub domains that have been delegated to their own authoritative servers and
- NAPTR hash table - to store NAPTR RRs corresponding to E.164 number.

The cache agent module is also developed as the sub class of the NS-2 UDP agent. On instantiation of the cache object, it updates all the three hash tables with the data from the configuration file; db.cache(see Fig. 4.4). The cache agent when receives the query packet from the resolver - have to follow successive referrals querying the different authoritative server until it receives a response for the query.

The cache Agent module works as follows:

A Packet Arrives In this case the cache node can either receive packets from the resolver (as query) or from the server (as response).

Query type packet When it is a query type packet, the cache server looks up its NAPTR hash table (which contains RRs) to verify whether it has response for the particular query. If it is a cache hit, then it updates the hash table, formulates the response type packet and sends it back to the resolver. On the contrary if it is a cache miss, then it searches in the NS hash table to find the authoritative server. Finally from the authoritative server hash table it gets the address of the AS node which can provide some information about the received request. For every data processing done, the corresponding hash tables have to be updated.

Response type packet In this case the cache server receives packet from any of the AS. If the packet that has been received contains the NAPTR value then it means that the response for the query is obtained. Then the response packet is formulated and send it back to the resolver. On the contrary if the packet contains information about another AS which might have information about the query, the cache server ask the AS for the address for the query and this process is continued until it gets an answer. For every transaction done in the cache server, the stipulated hash table have to be updated.

4.4.5 Authoritative Server Implementation

Similar to the cache agent here also three hash tables are used:

- NS Hash table
- AS Hash table and
- NAPTR hash table

The server agent is also created as the sub class of the UDP agent.

A Packet Arrives When the AS receives the packet, it uses a mechanism to determine whether the packet should be lost or not. The loss rate value for determining this process is obtained from the real measurements and the agent retrieves this data from the configuration file.

If packet not lost Here a server processing delay is introduced. This information is also obtained from the real measurements. A Gaussian distribution function is used to calculate the delay. Then it looks into its hash table either to send the server address for the query or a response which will be used by the cache server to identify the destination which can either give more details or the address of the server.

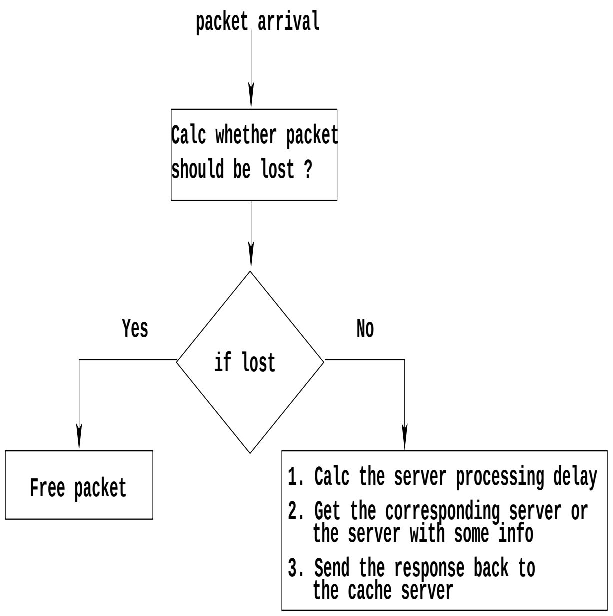

packet arrival
Calc whether packet should be lost ?
if lost
Yes
No
Free packet
1. Calc the server processing delay
2. Get the corresponding server or the server with some info
3. Send the response back to the cache server

Figure 4.11. Implementation of Authoritative Server in NS-2

4.4.6 IP link Model

The performance of the IP links between the resolver, the cache and authoritative servers can affect the global DNS response time. We decided to measure and model the two most important metrics which are the delay and the loss on these links. The model developed should reflect the features of an IP link. So we designed an asymmetric Internet cloud model (client-server nature of the Internet), where the delay and loss are correlated. Since we want the internet cloud to reflect the dynamicity and long-term dependence of the internet traffic we decided to use Hidden Markov Models (HMMs). HMMs are a great trade-off between simplicity and realism (Poisson modeling is easy to deal with, but not realistic enough, on the other hand fractals or wavelet analysis are an excess of what is required). Our model of an IP link has four HMMs, one for the delay and one for loss on both direction of the IP link as explained in chapter 3[Fig. 3.13].

To validate the model, we ran simulation of the HMM driven process and applied the methodology (Expected - Maximization algorithm). The parameters of the HMM found through this methodology were very close to the measured data, thus validating the IP link model.

In NS-2, Connector class is used to link two nodes (for e.g. a resolver and the cache server). In our simulation model we created a new class called HMMLink which inherits the Connector class. This object is composed of two markov chains: one for loss and one for delay. This object is placed between the objects head_ and enqT_ of the link class in ns-2 [7]

A packet arrives When a packet arrives in the link, the method "recv" in the HMMLink class uses the hidden markov chains to determine whether the received packet is lost or not, and if not, how much the delay is. Two other classes inherits the HMMLink class: HMMdelay and HMMloss

HMMdelay This object calculates the delay for the packet using a configuration file as shown below:

```
1
10    5
1000
Poisson
```

The first line indicates the number of states for the transition matrix while the parameters in the second line are for average and standard deviation delay. The third line is the time between each state in milliseconds and final line is used for identifying the distribution type.

HMMloss This object will calculate whether an arriving packet should be lost or not. The loss configuration file is as shown below:

```
1
0
1000
CBR
```

All the parameters in the loss configuration do the same operation as that of loss, except the second line which indicates the loss rate.

4.5 How the Simulator is Autonomic

According to IBM [53], an autonomic element will typically consist of one or more *managed elements* coupled with a single *autonomic manager* that controls and represents the managed elements. The managed elements can be a hardware resource such as storage or a CPU, or a printer, or a software resource such as a database, a directory service or a large legacy system. At an application level the managed element could be an e-utility, an application server or an individual business.

In our case, the autonomic manager is represented by the configuration files and the managed elements are the DNS directory service and the different servers involved in the French ENUM system. The configuration files monitor the managed elements and constructs and executes plan based on the information from the autonomic manager i.e. the configuration files.

The simulator that we have developed is an autonomic simulation tool. The input is given through the configuration files and the simulator with the help of its configuration files simulates, and depending on high level policies like performance it automatically find the optimized solution.

4.5.1 Testing Autonomicity

We verified the autonomicity of the simulation model from an architectural point. Now in order to verify and assess to what extend, the constructed simulation model is autonomous and to check if the modeled functionality is reflected in the operation of the system, we test one by one the four "self-*" properties that characterize an autonomous system.

Self-Configuring involves autonomic incorporation of new components and autonomic component adjustments to new conditions. Scientific applications often need to change and adapt. For example the input data of our simulation program may need to change because of uncertainty in setting the parameters. The changes may be in the input while generating the ENUM query or in the DNS database or in the topology itself.

As discussed previously (subsection 4.3.3) the simulation model is designed to get all its values from the configuration files. But this cannot really mean self configuring. Most of the self-configuring tasks are for the purpose of optimization. One of the case study (chapter 5) that we can take is setting the TTL value at the cache server on the basis of the average response time.

Response time is calculated in the *resolver* module. Initially the TTL value of the cache server is obtained from the configuration file and after a stipulated period of simulation the average response time for all the response received is calculated. Then the TTL metric is changed and simulation is performed for the same stipulated time as the previous one. Thus different TTL values are compared until an optimized TTL metric for the particular ENUM scenario is identified.

Self-Optimizing : The idea of using self-configuring tasks to optimization can also apply to self-optimizing. We would like to address load balancing of local cache servers for self optimization purpose. This case study can be done only when there is more than one local cache server for the resolver to connect to.

The timer algorithm (Algorithm 1) has been used for load balancing. In this case when the simulation starts the resolver searches the configuration files and sends the request to the first cache server in the configuration. The resolver then waits for the response until the specified timeout. If it obtains an acknowledgement then it continues sending request to the connected cache server. In case of not receiving the acknowledgement within the prescribed timeout, the resolver searches the configuration files for an alternative cache server and continues sending to that until there is a timeout. In this case the simulation model adapts itself depending on the load at the cache servers for achieving its objective-QoS.

To attain the "self-*" behaving properties with existing strictly layered approaches may be possible to certain extent, but will not leverage all the possible optimizations. According to [60], cross layered approaches are better suited to achieve the "self-*" properties targeted by autonomic systems. One of the case studies we have conducted (chapter 5) is using the cross layered approach. Here we calculate the load at the network for the connected cache servers. This is done at the network layer. When the load at the cache server reaches above a threshold, information is send to the resolver, which is at the application layer. The resolver now searches for an alternative cache server in the configuration file and redirects the queries to the alternative cache server. This approach is seen to further optimize the performance when compared to the load balancing done using layered approach with two cache servers.

Self-healing can be accomplished by automatically detecting, diagnosing and repairing localized software or hardware problems. At present in our simulation structure we have not introduced failure mechanisms, but it could become a necessity for testing certain scenarios. For example the operational database is vital in the successful operation of the system, therefore, a standby database is configured so that it can take the role of the primary database in case of failure. The DNS database in our simulation can have master and slave servers. The slave servers can be used in case of failure. Configuring the slave server at run can be done by the ”modify” module.

The modify module has been designed for two purposes:

1. To modify the cache and target server database at run time.
2. In case of introducing failure in the master database, the modify module will be used in changing the configuration files on run, so that the query is send to the newly configured secondary server rather than the failed primary database.

Self-protecting means the system automatically defends against malicious attacks or cascading failures. It uses early warnings to anticipate and prevent system wide failure. We have discussed the DNS attacks in chapter 3 while studying the DNS performance. ENUM scenario also needs protection. Access control for the user generated query can be done. Also the simulator could be used to detect one set of anomalous behavior such as DDoS attacks. This is a future scenario which could be incorporated in our simulation model.

4.6 Validation

As explained in chapter 3 our research methodology had three phases namely: *analysis, simulation* and *optimization*. The goal of the simulator is not to stop within this three phases. The promising solutions obtained from the simulation should be used in a fourth phase which is *implementation*. The results obtained from the simulation models should be implemented as real solutions in real world ENUM scenarios.

In order to use the simulation solutions in the implementation phase one should be confident enough that the obtained simulation results are accurate and meaningful. Validation of a simulation model with real measurements should be done to understand how accurately does the simulation model reflect the operations of a real system [78],[55].

As discussed by Harry Perros [78] one of the most powerful validation methods is called *output validity*. If actual data are available regarding the system under study (which we have done in chapter 3 by measurements), then these data can be compared with the output obtained from the simulation model. Obviously is it true that if they do not compare well, the simulation model is not valid.

In order to satisfy the *output validity* validation method, we first compared the global response time of the real and the simulation model. For this purpose we again state the reference [78] which explains about the need for a method called *Relationship Validity*. It defines that in order to have the structure of a system under study, fully reflected down to its very detail in a simulation model, the model's assumptions should be credible.

Assumptions like topology, bandwidth, generation of ENUM queries and DNS database have been verified by logical checks and also by the NS-2 visualizing tool NAM. Other parameters like the ENUM servers (Tier-0,Tier-1, Tier-2 and Tier-3) processing time has been taken as input from the real measurements. Also the IP link delay and loss values are obtained from real measurements. So to our knowledge *Relationship Validity* has been done.

We used parameters obtained from real measurements into the simulator and compared two kinds of results; for a lower number of request (5000) and a higher number of requests (36000). We compared the cumulative frequency distribution of real measurements and simulated results for validating the simulation model as shown in 4.12.

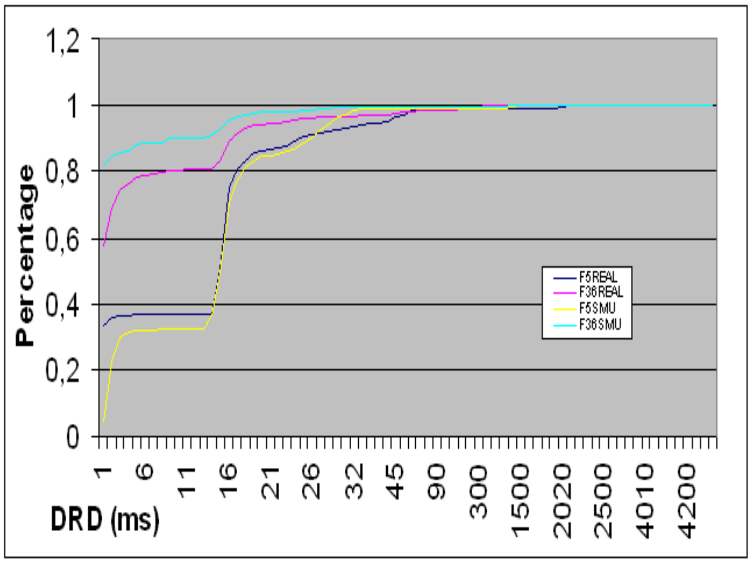


Figure 4.12. Comparison of real and simulation results

In [55] Averil M. Law suggests that a null hypothesis of saying that the real system and the simulation model are same is clearly false, since the model is only an approximation of the real system. So the question we have to ask is whether the differences between the model and the real system are significant enough to affect any conclusions derived from the model. To answer the question, statistical procedures such as confidence intervals can be used.

The confidence interval at 95% interval can be calculated by the formula:

$$Mean \pm (2 * \frac{StandardDeviation}{\sqrt{N}})$$

The standard rate error i.e. $\frac{StandardDeviation}{\sqrt{N}}$ is calculated as 1.05. So in order to have the confidence interval at 95% interval the mean global response time of the simulation results should should be $\pm$ 2*1.05 (i.e.2.10) of the real global response time measurements. The mean global response time of real measurements for 5000 queries estimated at the cache server in INT is 10,53116667. The global response time calculated for 5000 queries by simulation is 10.717. So this proves that the conclusions arrived from the simulation model can be significant in real implementations.

4.7 Summary

The primary contribution of this chapter is to implement the ENUM protocol in NS-2. We have made sure that the simulation model is built in such a way that it gets all its parameters from configuration files. In keeping in mind the heterogeneity of the technologies in NGN, the simulation model is designed to incorporate new application which uses DNS for address mapping.

At its present state the simulation model do not have integrated into it two of the four properties of autonomic computing envisioned by IBM namely; self-healing and self-protecting. The other two properties self-configuring and self-optimizing are implemented. Finally the simulation model is validated with standard validation procedures. The model that we have developed can be used as a platform for studying different ENUM scenarios.

Chapter 5

Optimization using the new simulator

It has been an axiom of mine that little things are infinitely the most important

- Sherlock Holmes in 'A Case of Identity,' by Sir Arthur Conan Doyle

The main objective of building the simulation model is to look at the influence of different metrics (such as Time to Live for different types of Resource Records in Tiers-(0...2) DNS Servers) and also optimization of algorithms at the cache server to improve the global DNS response time. It can be also used to study an optimized delegation model for a particular ENUM scenario. In future it could be used to study the feasibility of different ENUM enabled services with minimal modifications. In this chapter we give case studies of how we can use the simulation model to optimize the ENUM response time, which can be used as recommendation back to real implementations.

5.1 Performance Evaluation with respect to TTL values

In this section we study different TTL values using the developed simulation model and try to come up with solutions which can improve the performance of the ENUM system. Initially we study the impact of different TTLs at the DNS cache server and how it affects the delay occurred by ENUM address resolution.

Classical DNS name servers store the mapping of names to addresses in resource records, each having an associated TTL field that determines how long the entry can be cached by other name servers in the system. For example RFC 1912 recommends minimum TTL values around 1-5 days. Before RFC 1912, RFC 1536 had recommended 1 day as the minimum

TTL for most servers and around 4 days for top-level domains. A recent study [13] shows, however, that a majority of name server's use a default TTL value of 86400 seconds (or 1 day) for their domain.

A large TTL value reduces the load on the name server but limits the frequency of update propagation through the system. A large TTL can be effective in the case of address resolution for domain names in classical DNS. But in the ENUM scenarios it is the personal information that has been queried upon. For example if a user updates his current contact, it's not much useful if others who contact the user can be redirected to the updated contact address only after a day.

Small TTL values allow fine grained load balancing and rapid response to changes in the server or network load. The problem is small TTL values could significantly degrade the scalability of the DNS, since more requests would have to be transmitted in the network rather than being served from the local cache name server. Careful consideration is necessary when choosing DNS TTL values to balance responsiveness against extra client latency for ENUM scenarios. Figure 5.1 shows the global response time distribution of the simulation

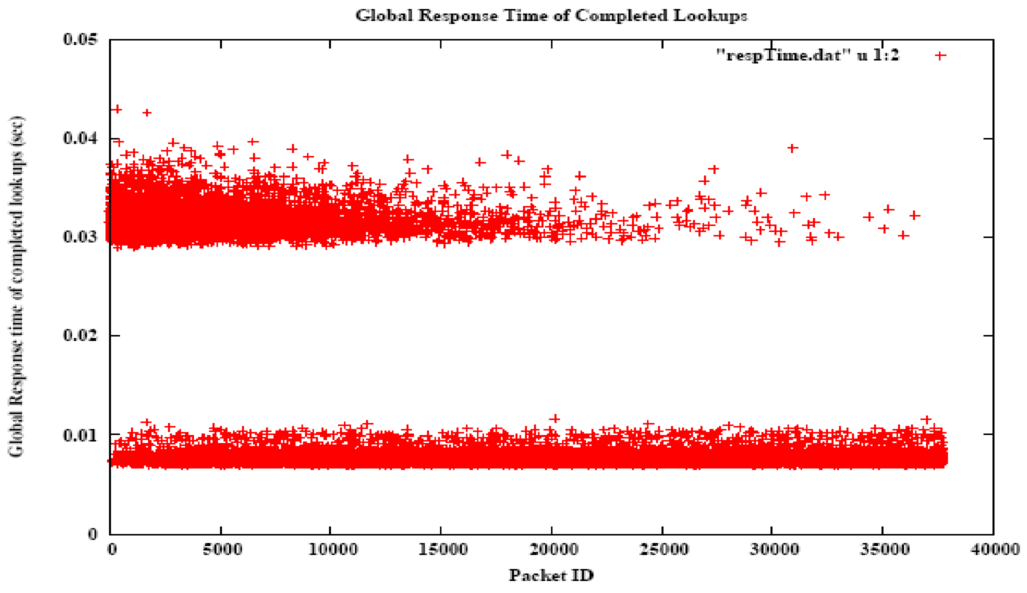


Figure 5.1. Response time distribution with TTL at the cache server as 1 day

results, where the TTL at the DNS cache server is fixed to 1 day. The scenario for the simulation experiments is as follows; we have 5000 different E.164 numbers as queries. The simulation is run for 1 hour. The total query that has been sent during this period is 37725. In an average it is around 10.5 requests per second. Since the input has only 5000 different queries, the same query will be repeated randomly. When the same query is repeated it will be present in the cache with such a large TTL value. If the response is in the cache already available for the query, then it needs only one hop. After a while even if there is a new query, which is not already cached at the DNS cache server, it need not query the Tier0 and Tier1 server, rather it will query directly the Tier2 block. So these queries will result in three hops. The scenario for the ENUM look up sequence for the simulation experiments

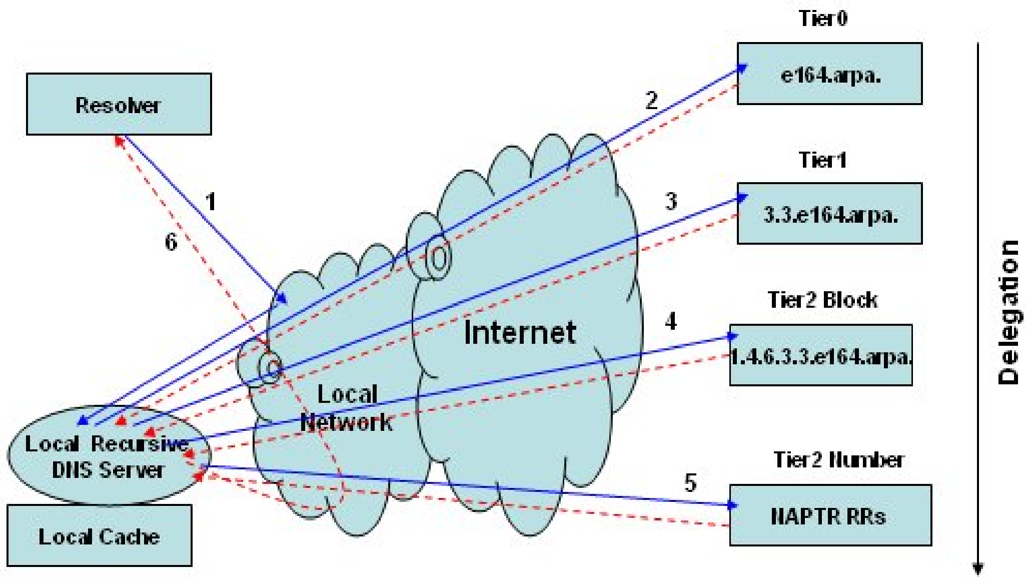


Figure 5.2. Example of ENUM lookup sequence (Requests are indicated in bold lines and response is indicated in dotted lines)

that we have conducted is as shown in figure 5.2:

1. The resolver asks the local DNS cache server for the NAPTR RR corresponding to an ENUM query.
2. Initially the cache server is empty, so the local DNS server asks Tier 0 for information. Tier0 refers to French ENUM server (Tier1).
3. The Tier1 French ENUM server refers to Tier2 block server which allocates chunk of numbers for different servers
4. Tier2 block server knows which server has the NAPTR RRs for corresponding query. So it redirects to corresponding Tier2 Number server.
5. Tier2 number server responds with the NAPTR RRs.
6. The local cache server responds back to the resolver.

With the same scenario as in figure 5.1, we conducted simulation experiments varying only the TTL values (3, 100, and 3600). The results in Fig. 5.3 lead to the following observations.

Observation

- A short TTL has higher average response time rather than larger TTL. This is quite logical.

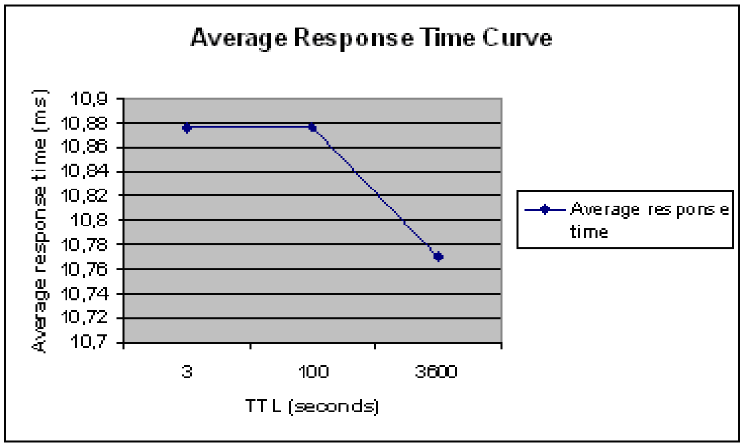


Figure 5.3. Average response time for different TTL values (10.5 requests per second)

- Total queries per second are around 10.5. So there is no loss. It means the DNS cache server is not stressed.
- Since the load on the DNS cache server is not heavy (Fig. 5.3) different TTL values do not make much impact. The difference in the average response time is just 0.1milliseconds between a TTL of 3 and a TTL of 3600.

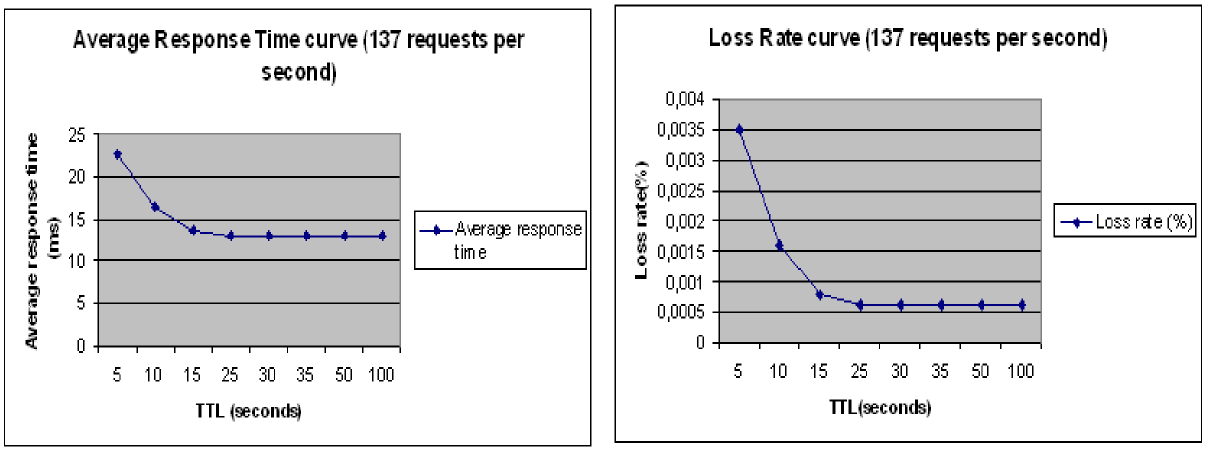


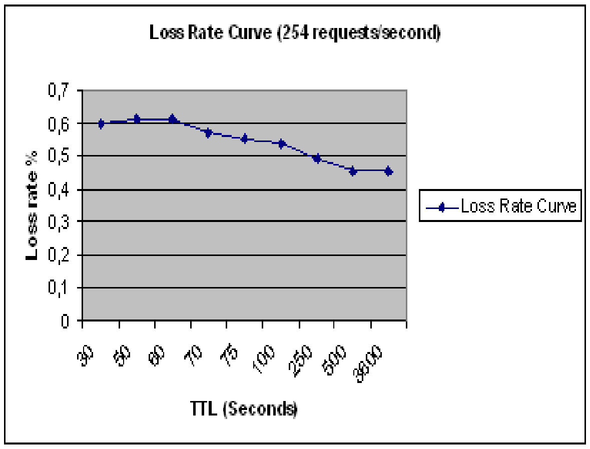
Figure 5.4. Average response time and loss rate for different TTL values (137 requests per second)

The next simulation experiment was conducted with a higher load, around 137 re-quests per second. The number of distinct E.164 queries was also increased from 5000 to 5500. In this case, as the DNS cache server is stressed we observe loss. The simulation experiments were run by varying the TTL values (5, 10, 15, 25, 30, 35, 50 and 100) in this scenario.

Observation

- As in the previous results (Fig. 5.3), a short TTL has higher average response time than larger TTL (Fig. 5.4).
- After a certain threshold the TTL does not make any impact. This can be observed for TTL values above 25 seconds (Fig. 5.4).
- Loss rate and response time are correlated.
- Our observation is that there is no use for increasing the TTL value after the cache server reaches a certain threshold "k".

This threshold depends on the DNS server parameters like the hardware, software and DNS configurations. To confirm our previous observations, we increased the request rate per second (around 254) and simulated. We observe similar results (Fig. 5.5) . Here the smoothening occurs around a TTL of 500 seconds. As the load increases the smoothening is



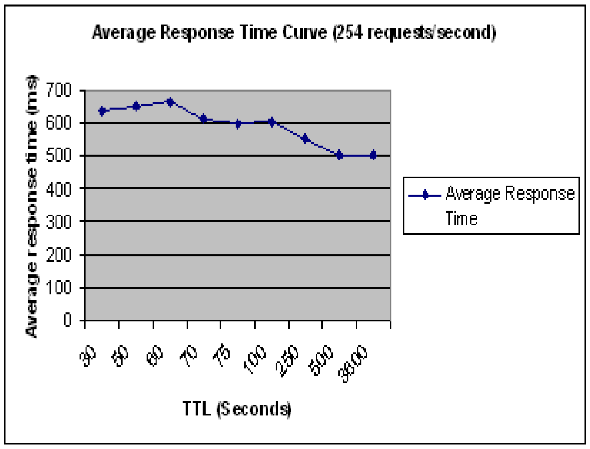


Figure 5.5. Average response time and loss rate for different TTL values (254 requests per second)

delayed. If we see the first results (Fig. 5.3) even though the graph shows a bigger deviation the actual difference is 0.1 milliseconds. So TTL does not play a bigger role in the DNS cache, when the cache server is not stressed beyond the server limits. But when the DNS cache server is stressed it is important to find the threshold point "k"; since after "k" even if we increase the TTL this is not going to improve the performance but on the contrary it is going to reduce the responsiveness of the server to new changes. Our proposition is that

$$Response\ Time \quad \alpha \quad \frac{1}{TTL}\ up\ to\ a\ certain\ threshold\ "k"$$

The "k" value depends on the DNS server configuration. Smoothening of the TTL is delayed when the request rate increases. So it is vital to have higher TTLs when the request rate is high and stop increasing the TTL after the threshold point.

Our solution is to have an adaptive TTL based on the number of requests. A higher TTL is assigned when the request rate is high and a lower TTL when the request rate is less. This can be done by keeping track of the number of requests at the DNS cache server, periodically collecting the client request rate coming from different resolvers.

Unlike the classical DNS where it is effective when certain well known domain names are given higher TTLs (for example google.com) with respect to lower TTL value, for not so well known domain names, ENUM maps individual users contacts where such policies cannot be adopted. So it is not effective to characterize users in ENUM case. Our proposal is to set adaptive TTL values based on the client request rate, which can improve the DNS cache server performance and in the same time maintaining the responsiveness of the server to external changes.

5.2 Performance Evaluation with respect to number of hops

As explained in section 5.1 a large TTL value reduces the load on the name server but limits the frequency of update propagation through the system. A large TTL can be effective in the case of address resolution for domain names in classical DNS. But in the ENUM scenarios it is the personal information that has been queried upon. Small TTL values allow fine grained load balancing and rapid response to changes in the server or network load, which is the case of ENUM.

Contemporary DNS resolution make effective use of caching, thanks to large TTL values. The cache hit rate is 90% depending on the starting state of the database [43]. In such cases, the response time of DNS query is independent with hop count.

But in case of the ENUM the effective use of caching is unlikely. The first reason is the use of small TTL values to allow rapid response to changes and the second reason is a local name server will not cache all of NAPTR Resource Records with E.164 numbers [43]. The ENUM resolution will have excessively low cache hit rate.

Figure 5.6 shows the simulation results of the number of hops distribution percentage, where the TTL at the DNS cache server is fixed to 36000, 100 and 5 seconds. The scenario for the simulation experiments is as follows; we have 5000 different E.164 numbers as queries. The simulation is run for 1 hour. Since the input has only 5000 different queries, the same query will be repeated randomly. When the same query is repeated it will be present in the cache when there is a large TTL value. If the response is in the cache already available for the query, then it needs only one hop. After a while even if there is a new query, which is not already cached at the DNS cache server, it need not query the Tier0 and Tier1 server, rather it will query directly the Tier2 block. So these queries will result in three hops. Only at the beginning of the simulation when the cache is empty it needs 5 hops, so that it has to query additionally Tier0 and Tier1 server. When the TTL value is less (as in case of 100 and 5 seconds) we observe an increase in the percentage of 3 and 5 hops (In the graph we

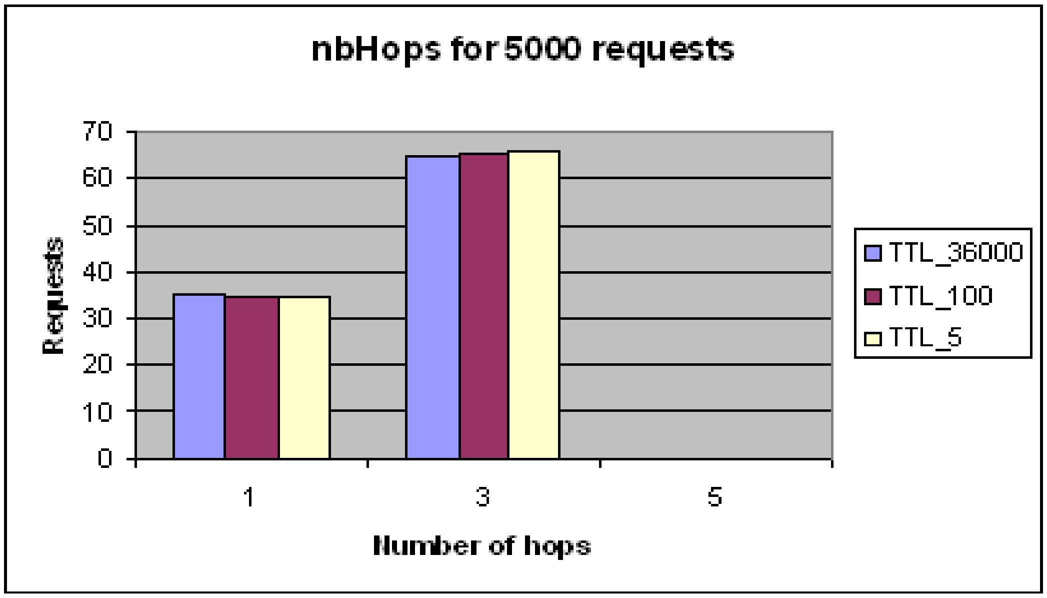


Figure 5.6. Number of hops distribution for 5000 requests

are not able to view the histogram for 5 hops since their percentage are very less (.0399 for TTL 36000, .08 for TTL 100 and .18 for TTL 5).

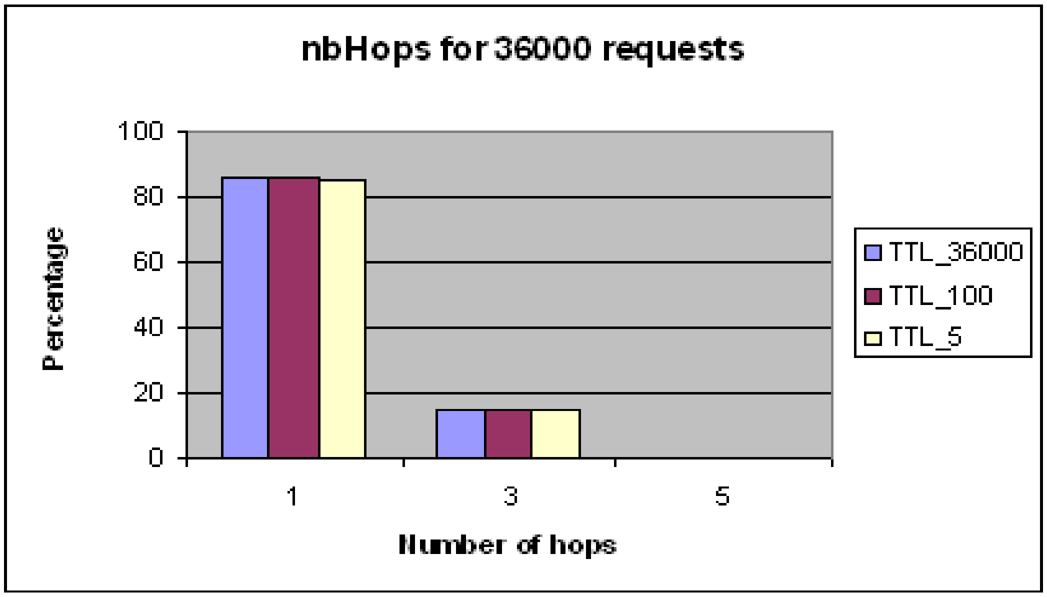


Figure 5.7. Number of hops distribution for 36000 requests

For the same simulation scenario as above we increase the rate of the number of requests sent and the figure 5.7 shows the results for first 36000 requests for different TTL values of 36000, 100 and 5. The only difference between the previous and the present results are that since there are only 5000 different queries given as input, the same queries will be repeated in a higher frequency than the previous results.

The observation from both the figures 5.6 and 5.7 is that when the cache hit rate decreases the number of hops increases. In a real world scenario the cache hit rate of local name server for ENUM query is supposed to be very low relatively to DNS query, and it will never get ahead of DNS's cache hit rate, because the number and size of NAPTR Resource Record per E.164 number should be more and bigger than the resource record for the IP resolution made by the DNS.

As we have explained earlier, the end-to-end delay over a fixed path can be defined by four components *processing delay*, *queuing delay*, *transmission delay* and *propagation delay* . The delay distribution is hop by hop and it increases with each node that the packet traverses. From our experiments in section 3 we have seen the delay between two nodes on our test set up is much smaller due to the dedicated connection between the different machines and the machines used were specifically for our experiments. But this is not the case in real world scenarios. So with every increase in the hop count leads to a strongly deteriorated ENUM look up time and the total response time is influenced due to this.

The results shown in 5.6 and 5.7 does not cause any stress on the local cache server and so there is no loss incurred. We now look into another case where the simulation experiment was conducted with higher load around 137 requests per second. The number of distinct E.164 queries were also increased from 5000 to 5500 and the simulation was run for 1 hour. The results are shown in the tabular column, since certain data are not visible in the graph due to their small values.

nb of hops	**nb of requests** TTL_5	**nb of requests** TTL_10	**nb of requests** TTL_15	**nb of requests** TTL_25	**nb of requests** TTL_30	**nb of requests** TTL_50
1	484479	483831	484410	484600	484694	484651
2	50	53	9	10	12	12
3	10892	10836	10842	10946	10801	10973
4	166	103	55	43	40	39
5	52	30	23	13	11	8
6	30	1	2			
7	1					

Table 5.1. Number of hops for different TTL values where the request that is sent to the cache server for each second is around 137

Observation

- On all the TTL values we observe a loss rate even though it is very small. The average response time and loss rate for this simulation result is shown in Fig. 5.4
- If there is no loss incurred for resolving the NAPTR RR for an E.164 number, the maximum number of hops that the request packet will traverse according to the dele-

gation model of the French ENUM design is 5. The scenario (from Table 5.1) where the number of hops goes above 5 is due to the packets being dropped at any of the server and the resolver resend the packet

- As seen in graph 5.4 a shorter TTL has higher average response time. The new observation is that shorter TTL results in more number of hops (Table 5.1)

To improve the total response time it is necessary to lessen total hop count, especially caused by ENUM look up. The DNS look up time can be controlled by effective caching at the local name server. Name server for ENUM is bound with geographical location or operator based, so it is difficult to lessen hop count.

Usually when a local name server receives a query about NAPTR RR from a host, it first searches its cache. If it finds appropriate NAPTR resource records for the query, it responds to the host and ENUM resolution is over. Otherwise the local name server forwards the query to the Tier1 name server, which returns a referral response, composed of the address of Tier2 block name server. Then it is again referred to the Tier2 number name server which builds the response packet containing the NAPTR RRS and sends it back to the local name server. As the local name server receives a response, it caches and forwards the response to the host.

What we propose here is a change in the French ENUM Tiered design. If we run the local name server to represent each local area according to the geographical regions or based on the operators. For example each of the operator is assigned a chunk if numbers as we have explained in Chapter 3. In the same manner we propose separate servers for each region. For example for Paris the PSTN number starts with ”01” and for Toulouse it starts with ”05”. If the local cache server has information about these number it can directly send the queries to the corresponding Tier-2 Bloc server which possesses the NAPTR RRs corresponding to the E.164 number, thus reducing the number of hops the query packet will traverse. The proposed change in the design to point to the representative Tier-2 Bloc server for the resolution of the French E.164 numbers results in better ENUM resolution process as shown in the figure 5.8.

5.3 Performance Evaluation with respect to load balancing

In this example initially we use our simulation tool to study at how much load, optimization or increase of resources is needed at the cache server for a particular scenario.

In the real measurements we made, the test platform had dedicated connections with our partners (SFR, Orange and France Telecom). For our stress tests we had high bandwidth. Due to this we were able to see loss only after around 9000 requests per second. While validating the simulation model we also used high bandwidth (100Mb) for our links. But

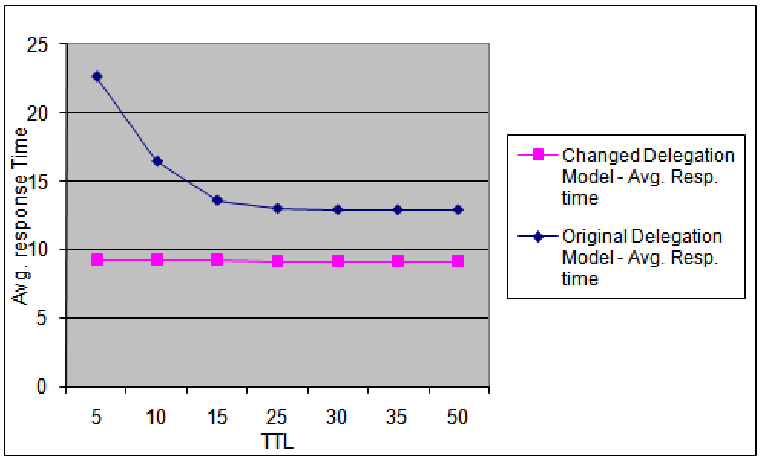


Figure 5.8. Difference between the original French ENUM delegation model and the modified delegation model

that need not be the case in real world scenarios. So the configurations for the simulation topology were altered with bandwidth of 10Mb between the resolver and the cache and bandwidth of 3Mb between the cache and the authoritative servers.

Once the query is sent to the cache server a timeout is calculated for the query (in case the packet gets lost or delayed). After the timeout, if a response for the query is not received, the query is resent to the same or different cache server depending on the algorithm that is explained in the timer algorithm (Algorithm 1 in section 4.4.3). The timeout values are calculated as per the following procedure in the table here:

nb_cache_servers/nb_cycle	1	2	3
0	5	2*5	3*5
1	10	2*5	3*5
2	20	2*10	3*6
3	40	2*20	3*13

Table 5.2. Calculation of the timeout depending on the number of cache servers

In the first experiment we used only one cache server. From the graph 5.9 for one cache server we observe that once the number of requests per second reaches around 250, the response time is getting more than 200 ms which is more than the average PSTN response time [3]. But for two cache servers using the same bandwidth and topology as in the case of one cache server above, there is a decrease in the response time latency to much less than 200 ms. It is quite obvious that by increasing the resources in a system, the performance will be improved. What we want to show by this example is, depending on the hardware/software configurations of the authoritative and the cache server, the characteristics of the IP links and depending on the average arrival requests, one can arrive at the resource requirements for a particular ENUM implementation with the help of our simulation tool.

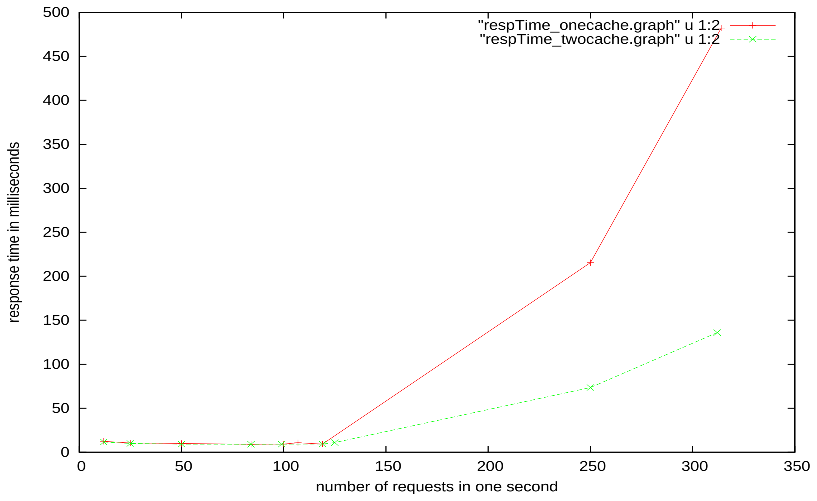


Figure 5.9. Response time comparison for one and two cache servers

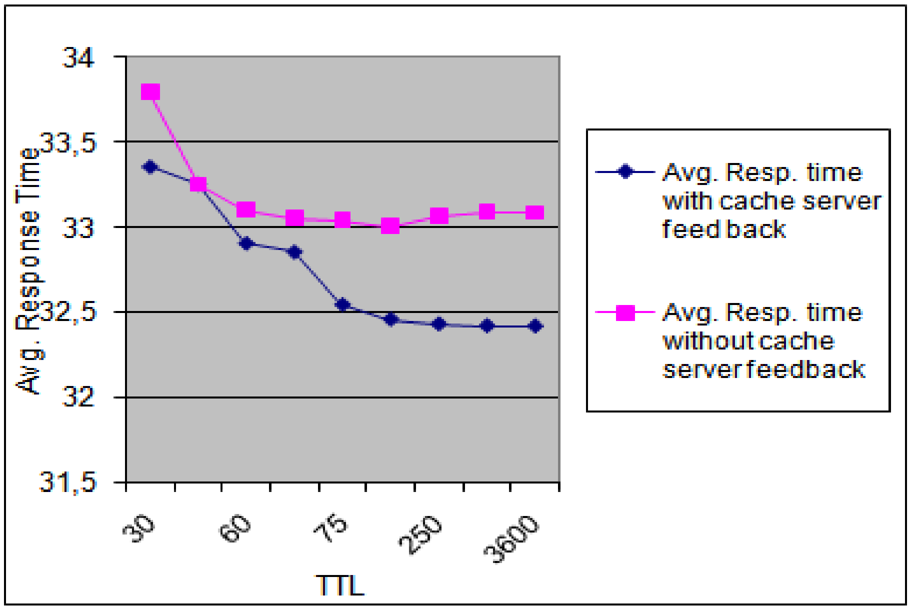


Figure 5.10. Response time comparison with and without asynchronous feedback

Layered architectures are not sufficiently flexible to cope with the dynamics of next generation communications. Research in wireless and adhoc networks have already proved that cross-layer architectures may provide better QoS[36]. Cross-layering allows interaction between two or more non-adjacent layers in the protocol stack. We run this experiment with a cross layer feedback between the network layer at the cache to the application layer at the resolver.

In this case, as an addition to the previous test, instead of waiting for the cache server to get overloaded, we gave a acknowledgement feedback from the cache server to the resolver, when the bandwidth of the current cache server is above a threshold value. Once the resolver receives this acknowledgement, it searches the configuration file for any other alternative cache server. In our case we have two cache servers for the experiment. Hence, it redirects the query packet to the alternative cache server. Using this technique we compare the results (Fig. 5.10) with and without the feedback from the cache server. The results do prove that the cross layered approach has a better performance.

5.4 Summary

In this chapter we demonstrate how the simulation tool could be used to study address mapping scenarios and will be helpful to give promising solutions. Here we run simulations to study different factors affecting the response time.

We started with evaluating the effect of TTL on the global response time. Higher TTL values are not recommended in ENUM as in the classical DNS and on the contrary smaller TTL values can increase the load on the network and the name servers. From our experiments we propose a way to fix adaptive TTL values depending on the load coming to the cache servers. In the second experiment we propose a design change in the existing French ENUM model to reduce the number of hops and in turn thus reducing the global response time. Finally we demonstrate how load balancing using asynchronous feedback could further optimise the ENUM resolution time.

Chapter 6

Conclusion and Future Work

I just want to say one word to you-just one word-plastics
- From "The Graduate" 1967.

6.1 Conclusion

In this thesis we have identified one of the key issues in the NGN, i.e. the lack of an universal addressing system for communication between different network architectures, services and applications. Our overall approach is to build an autonomic simulation platform which can be used to study and optimize addressing issues in NGN. For this purpose we identify an appropriate technology that can be used to address mapping between different types of networks. Finally by analyzing an empirical model we build a simulation model, which we then use to study and optimize address mapping scenarios between PSTN and IP. This final objective of this thesis is to use the autonomic simulation platform to study different NGN addressing scenarios, find out how we could improve the performance and put forth the findings as recommendations back to real world implementations.

In this chapter, we summarize the findings and contributions of our work, present concluding discussions, and describe opportunities for further explorations.

6.1.1 Summary

This thesis identifies three fundamental challenges:

1. The convergence towards NGN requires that different network users using different network technologies, can communicate with each other and access resources on distinct

network infrastructures. This requires the interworking of different Naming, Addressing and Numbering systems. Another main issue of NGN convergence is the multiple PCIs for a single person and identifying which is the best way to contact someone.

2. The methodology to study the performance of a real system. The problem here is that is not always possible to study a real system and come up with promising solutions. There are several reasons for this, like the size, studying a new system which is not yet built, etc. The feasible method is to scale down the real systems and experiment via theoretical or simulation studies

3. In NGN where numerous entities are involved to realize the service functionalities, our belief is an autonomous approach is needed to deal with the complexity. Any simulation model built to study NGN should be able to accommodate new services or application and also adapt dynamically to changes which are not foreseen. In order to study a rapidly changing system, building a static simulation model will be short sighted.

This dissertation analyzed the problems posed by the above challenges and used a combination of literature study, empirical measurements, development of new tools, building analytical and simulation models to solve them.

ENUM Protocol

The first contribution has been the identification of ENUM to study NGN addressing issues. ENUM has been chosen for our work because:

- ENUM can be used for interworking between telephony (and other carrier services) and IP services (including IP telephony and multimedia). Using ENUM the end user is identified by a E.164 number that can be associated with variety of networks in PSTN or IP.
- ENUM uses the existing DNS infrastructure for address mapping. DNS is already a proven method for address mapping in the Internet. It has been proven effective since its scalable, reliable and anyone can use it.
- It solves the problem of number portability. It enables the originating administrative domain to do an All Call Query (ACQ) to find the destination network.
- Using ENUM it is possible to use a single PCI to communicate with other different PCIs associated with the initial identifier.

ENUM has been chosen, after studying the possibility of a new protocol and then surveying the major existing technologies used for address mapping across network boundaries - TRIP, UPT, UCI, H.323 and SIP.

6.1.2 Methodology in studying the empirical model

A new approach was used to study the empirical French ENUM model. This study was done in order to get metrics like loss rate and response time from the empirical model, so that this parameters could be used to build the simulation model, which resembles the real system, and also to validate the simulation system as an approximation of the empirical one.

To obtain these parameters we made *measurements* on the different servers in the French ENUM model. These measurements were made in different phases which are explained here:

Measuring the capacity of the servers used This experiment was conducted to test the CPU usage of the server used in the French ENUM system. To understand how much load these servers can withstand under stress, measurements were conducted in the Tier1, Tier2 Bloc and Tier2 number servers. The stress test was conducted using the tool we developed to generate queries of ENUM format.

Analyzing the local DNS servers This experiment was done to measure loss rate and average response time locally for the three target servers (Tier1, Tier2chunk and Tier2number server). The existing tool Queryperf was modified and was used to stress the local DNS Cache Server with queries of ENUM format. The load to the cache server was done with a wide range starting from a low to high QPS rates.

On observing the measurement results one notes that in the beginning the loss rate and average response time follows an exponential curve(y = $ae^{\alpha x}$) while after a certain threshold it follows a linear pattern(y = ax + b). This threshold value depends on the technical characteristics of the server (Hardware and DNS Software used on the machine). Curve fitting techniques were employed to separate the exponential and linear part.Then we proved that our observation is correct by linear and exponential regression techniques. We identified parameters, which when applied into the exponential and linear equations, best fit exponential and linear part of the real world measurements. Thus parameters(α, a, and b) were calculated for all the three target servers to be used in the simulation tool.

Analyzing the IP links Measurement and modeling of the two important metrics (loss and delay) which impact the performance of the IP links connecting the resolver (i.e. the client where the query is generated), the cache server and the authoritative servers (Tier1, Tier2chunk and Tier2number) was done. We wanted the model to reflect the features of an IP link, so an asymmetric Internet model (client-server nature of the internet) was designed, where the delay and loss are not correlated. HMMs were used to reflect the dynamic and long-term dependence nature of the Internet traffic. The model for an IP link has four HMMs one for the delay and one for the loss on both the directions of the IP link. The model was validated using simulation.

The values (loss and delay) obtained from this experiment was used as input in the simulator to simulate loss and delay for the IP links connecting the different nodes of the simulation topology.

Global response time measurements While the stress tests of the local DNS server was proceeding we calculated the global response time. Global response time cumulative distribution function was obtained for 5000 and 36000 queries, which has been used to compare with the simulated global response time results.

6.1.3 Building the simulation model and validation

As to our knowledge we were the first to create ENUM DNS simulation model on the network simulator (NS-2). About 1500 lines of code were written to simulate the complete ENUM DNS set up with real DNS configurations. We used the parameters obtained from our analytical models into the simulator.

In order to use the simulation solutions back in real implementations one should be confident enough that the obtained simulation results are accurate and meaningful. *Validation* of the simulation model with real measurements were done to understand how accurately the simulation model reflects the operations of the real system.

We first used the *output validity* method to perform the validation. For this validation method we compared the global response time cumulative frequency distribution data of both real measurements and simulated results. The graph of the results proves that the the real and the simulated data are almost similar.

Further, in order to answer the question whether the difference between the model and real system are significant enough to affect any conclusions derived from the model, we used statistical procedures. The global response time results of the simulation were inside the 95% confidence interval when compared with the global response time of the real measurements. Thus from the validation procedures that we made, we conclude that our simulation model is a good approximation of the real systems and any promising solutions obtained from the simulation study could be used back in the real implementations for optimized design and performance of such systems.

The simulation model was built keeping in mind the heterogeneity factors of the NGN and the evolving services and applications that need address mapping between different types of networks. All the input parameters can be obtained from configuration files, which enables trying new scenarios, add new applications and technologies and run simulations. An autonomic approach was taken in building the simulation model keeping the mind the four pillars of autonomic computing - self-healing, self-protecting, self-configuring and self-optimizing. We have demonstrated in section 4.5.1 how the simulation model built satisfies these properties.

6.1.4 Optimization using the new simulator

In addition to solving the specific problems discussed above, we also use our simulation model to study French ENUM convergence scenarios and and we project how this simulation model is useful to study and optimize real address mapping models. We propose three recommendations which we think could help to reduce the ENUM resolution time. They are:

1. The simulation run with different TTL values show that the response time is inversely proportional to the the TTL value until a certain threshold. Our solution is to have an adaptive TTL based on the number of requests. A higher TTL is assigned when the request rate is high and a lower TTL when the request rate is less. This is done by keeping track of the number of requests at the DNS cache server, periodically collecting the client request rate coming from different resolvers. This experiment shows the ability of the simulator to dynamically incorporate data into running simulations which is an example of dynamic data driven application systems [28].

2. To improve the total response time it is necessary to lessen total hop count, especially caused by ENUM look up. The DNS look up time can be controlled by effective caching at the local name server. Name server for ENUM is bound with geographical location or operator based, so it is difficult to lessen hop count. What we propose here is a change in the ENUM Tiered design. If we run the local name server to represent each local area according to the geographical region or operator. In such case if the cache is empty and any query with +33 is forwarded to the Tier-2 Bloc server rather than the root name server (Tier0 i.e e164.arpa.). The proposed change in the design to point to the representative Tier-2 Bloc server for the resolution of the French E.164 numbers results in better ENUM resolution process.

3. In the final case study conducted, we initially point out how our simulation model could be used to to study at what load for a particular scenario, optimization or increase of resource is needed. For further optimization, we use a cross-layered approach wherein instead of waiting for the cache server to get overloaded and waiting for a timeout, we gave a acknowledgement feedback from the cache server at the network layer to the resolver at the application layer, when the current cache server reaches a threshold limit. Once the resolver receives this acknowledgement, it redirects the packet to the alternative cache server(the identity of the cache server is obtained from the configuration files) . Using this technique we compare the results with and without cross layer approach using the same scenario. The results prove that cross layer feedback technique improves the performance in this scenario.

6.1.5 Autonomicity

In NGN as we have explained before, involves heterogeneous transport networks applications and services. Nowadays, the user needs are also becoming increasingly demanding and customized. In order to satisfy these requirements, network has to integrate different properties like reliability, QoS, dynamicity, mobility, service adaptation etc. As explained in this article [80] an adaptive and dynamic selection of control mechanism, taking into account the current traffic situation, is necessary to optimize the network resources and come up with a more important number of user expectations associated with QoS.

To realize such functionalities, it is necessary to be able to configure automatically the network in real time and therefore enable all the equipments in the network to react to any kind of changes in the network. In our simulation tool we have kept all the configurations separately from the core simulator program. This enables each of the entities in the simulator to access these configuration data to use or to make changes back in the configuration so that dynamic adaptation is possible. In chapter 4 and chapter 5 we have explained how this has been done with different simulation scenarios.

This dissertation does not strive to solve the entire problem of autonomic computing, but instead focuses on select autonomic computing capabilities for NGN addressing scenarios. We acknowledge autonomic simulations are difficult to achieve, they are still more difficult than cross-layer methods. Using the developed simulator, we conduct autonomic simulation based on parameters like TTL and also adaptive feedback based on the network constraints.

6.2 Availability

The software for the simulation platform described in the dissertation are available online.

- HMM modules for NS-2 are found in the hmm directory
  (www-rst.int-evry.fr/∼balakric/numerobis_final/hmmlink)
- C++ code for other DNS Modules are found in the C++code directory
  (www-rst.int-evry.fr/∼balakric/numerobis_final/dnsmodules/C++code)
- The simulation script is found in the tclcode directory
  (www-rst.int-evry.fr/∼balakric/numerobis_final/dnsmodules/tclcode)

The procedure to be followed are:

- Change the library files of NS-2 by the following steps
  (www-rst.int-evry.fr/∼balakric/numerobis_final/libraryfiles)
  - cp ns-lib.tcl.enum ../tcl/lib/ns-lib.tcl
  - cp ns-link.tcl.enum ../tcl/lib/ns-link.tcl
  - cp ns-default.tcl.enum ../tcl/lib/ns-default.tcl
- Modification of NS-2 "Makefile.in" by following steps
  - add hmmlink/hmmlink.o
  - add dnsmodules/dnspool.o
  - add dnsmodules/dns_resolver.o

    - add dnsmodules/dns_cache.o
    - add dnsmodules/dns_server.o

- Unzip the config files in the directory wherever it is convenient unzip it by "tar xvzf dnsmodules.tar.gz"

Note

- The file extension in the tcl script "numerobis.tcl" from where the config files are called should be changed according to the place where the config files are untared
- Similarly the file extensions in dns_resolver.cc, dns_cache.cc and dns_server.cc should be modified accordingly to where the config files are untared

6.3 Future Directions

There are several interesting directions for future work based on the work described in this dissertation. Some of these are extension of our work, while some are motivated by the general problems in NGN addressing.

Privacy and Security In ENUM all communications associated with each E.164 number are generally stored in publicly accessible database. By querying the DNS database any individual requesting information on a random E.164 number will be able to access all communication services (such as email, fax number, telephone number etc.) associated with that number. This information can be misused by malicious persons or organizations.

Unless strict authentication procedures are developed, the introduction of ENUM may also provide opportunities for some service providers to insert telephone numbers in the DNS without an end users consent, thus potentially redirecting calls to these numbers via their own networks. This would allow service providers to collect transit revenues illegally [**?**].

DNS Security Extensions (DNSSec) is one such extension to DNS which adds security to the DNS. DNSSEC was designed to protect the Internet from certain attacks, such as DNS cache poisoning, by providing:

- Authentication of origin of the DNS data.
- Data integrity and
- Authenticated denial of existence.

All answers in DNSSEC are digitally signed and can be checked. Due to this checking the process is time consuming and leads to an increase in overall response time. This could be an area where we could use the simulation platform to incorporate this extra security modules and study their behavior for improving the global response time.

Experimentation with different scenarios ENUM can be sold only as an application not as a technology and this application have to be a killer one to be successful. It has to be accepted not only by professional users (e.g. Enterprise ENUM) but also by general users (User ENUM). For a general user to accept a new service, he should be comfortable with it. For e.g. if an ENUM enabled service takes more time for a call set up than the ordinary telephone connection which he is normally used to, he will not embrace this application even though it provides him with much advanced services.

Our work has set up a basic platform where we can test different models or different types of ENUM applications on it. The simulator that we have developed can help to study the feasibility study of different ENUM enabled services with very little additions made on the existing simulator. The approach we have taken to study the ENUM French model can also be used to study other models and use those parameters into the simulator to have an emulation of the real world scenario.

Emulation and Simulation Emulation refers the ability to introduce the simulator into a live network. Instead of the enum client generating the enum queries, we would like to have real traffic being introduced to the simulator and be affected by the objects (like resolver, cache and authoritative servers) within the simulation. This would give us more real results to study and optimize.

If i have seen further it is by standing on the shoulders of Giants

-Sir Isaac Newton, letter to Sir Robert Hooke, Feb. 5, 1676

Bibliography

[1] Bind (berkeley internet name domain) a dns server implementation from isc (internet systems consortium) http://www.isc.org/index.pl?/sw/bind/.

[2] Dns survey (http://mydns.bboy.net/survey/).

[3] Getting underway with enum: Building a dns infrastructure for the convergence of telecommunication networks, nominum white paper.

[4] http://cr.yp.to/djbdns.html.

[5] http://enumdata.org/.

[6] http://sourceforge.net/projects/libpcap/.

[7] *The Network Simulator ns-2: Documentation: www.isi.edu/nsnam/ns/ns-documentation.html.*

[8] Tiphon release 4: Etsi technical specification 101 321 http://www.transnexus.com/osp

[9] Vint : Virtual internetwork testbed, http://www.isi.edu/nsnam/vint/index.html.

[10] www.packetfactory.net/libnet/.

[11] Itu-t recommendation f.850: Principles of universal personal telecommunication (upt), March 1993.

[12] Itu-t recommendation f.851: Principles of universal personal telecommunication (upt) - service description, February 1995.

[13] Itu-t recommendation h.323, "packet-based multimedia communications systems", February 1998.

[14] Itu-t recommendation f.850: Principles of universal personal telecommunication (upt) - service description, March 2000.

[15] How to measure the performance of a caching dns server (white paper) from, 2002.

[16] S. Mohan A. R. Modarresi. Control and management in nextgeneration networks: Challenges and opportunities. In *IEEE Communications Magazine, pp. 94102.*, Oct. 2000.

[17] M. Allman and V. Paxson. On estimating end-to-end network path properties. In *in: Proceedings of SIGCOMM*, September 1999.

[18] F. Heckenast T. Kovacs K. Varga A. Varjasi N. Benyo, B. Hatwagner. Novel communication services based on enum technology. In *INES*, 2005.

[19] T. Berners-Lee. Uniform resource identifiers (uri): Generic syntax. rfc-2396, August 1998.

[20] Laura Bertolotti and Maria Carla Calzarossa. Models of mail server workloads. In *Elsevier Science*, 2001.

[21] Jean-Chrystome Bolot. Characterizing end-to-end packet delay and loss in the internet. *In Journal of High Speed Networks*, 2:305–323, 1993.

[22] BUGNAET Thomas BOSSANT Pierre-Yves. Numerobis sp4 livrable 11.2 : Performance des liens ip. Technical report, Institut Nationale Des Télécommunications, 2004.

[23] Thomas Bugnazet. *Métrologie dans les réseaux IP, Performance du DNS & ENUM.* PhD thesis, Université Pierre et Marie Curie, Novembre 2006.

[24] Randy Bush. The dns today: Are we overloading the saddlebags on an old horse? Technical report, 49th IETF Meeting Plenary Presentation, December 2000.

[25] Gerard Hooghiemstra Charles Bovy, Harry Mertodimedjo and Piet van Mieghem. Analysis of end-to-end delay measurements in internet. In *in In Pro-. ceedings of the Passive and Active Measurements Workshop*, 2002.

[26] D. Knigth C.S. Lee. Realization of the nextgeneration network. In *IEEE Communications Magazine, pp 3441.*, Oct. 2005.

[27] Hanlin Dalgic, Ismail; Fang. Comparison of h.323 and sip for ip telephony signaling. In *Proc. SPIE Vol. 3845, p. 106-122, Multimedia Systems and Applications II, Andrew G. Tescher; Bhaskaran Vasudev; V. Michael Bove; Barbara Derryberry; Eds.*, November 1999.

[28] Frederica Darema. Dynamic data driven applications systems - new drivers for networking and communications. Globecom, November 2007. Invited talk.

[29] E. O. Elliott. Estimates of error-rate for codes on burst-noise channels. *Bell Syst. Tech. Journal*, 42:1977–1997, 1963.

[30] T.Berners-Lee et al. The world wide web. In *Communications of the ACM*, 1994.

[31] P. Faltstrom. E.164 number and dns.rfc-2916, September 2000.

[32] P. Faltstrom. The e.164 to uniform resource identifiers (uri) dynamic delegation discovery system (ddds) application (enum). rfc-3761, April 2004.

[33] Andreas Fasbender and Peter Davis. Measurement, modelling and emulation of internet round-trip delays. In *Proceedings of the 8th International Conference on Modelling Techniques and Tools for Computer Performance*, 1995.

[34] S. Kalidindi G. Almes and M. Zekauskas. A one-way delay metric for ippm. rfc-2679, september 1999.

[35] S. Kalidindi G. Almes and M. Zekauskas. A one-way packet loss metric for ippm. rfc-2680, September 1999.

[36] Vincent Gauthier. *Méthodes et Algorithmes transcouche pour les réseaux ad hoc sans fil*. PhD thesis, l'INT en co-habilitation avec Université Paris VI, 2006.

[37] E. N. Gilbert. Capacity of a burst-noise channel. *Bell Syst. Tech. Journal*, 39:1253–1265, 1960.

[38] Gérard Heburtene. A comparative survey on simulation packages. Technical report, Institut Nationale des Télécommunications, 2001.

[39] H.Heffes and D.Lucantoni. A markov modulated characterization of packetized voice and data traffic and related statistical multiplexer performance. *IEEE journal select Areas Communication*, SAC-4:856–868, 1986.

[40] Helmut Hlavacs and C.W. Ueberhuber. Performance evaluation by simulation. Technical report, Vienna University, 2001.

[41] Gerard Hooghiemstra and Piet Van Mieghem. Delay distributions on fixed internet paths. In *Delft University of Technology*, 2001.

[42] G. Huston. Management guidelines & operational requirements for the address and routing parameter area domain ("arpa"). rfc-3172, September 2001.

[43] Youngsong Mun Hyewon K. Lee. Performance evaluation of enum directory service design. In *International Conference on Computational Science*, 2004.

[44] David Karger 11. Frans Kaashoek I. Stoica, Robert Morris and Hari Balakrishnan. Chord: a scalable peer-lo-peer lookup service for internet applications. In *in the Proceedings ofACM SIGCOMM*, 2001.

[45] C.A.N. Soules R.W. Wisniewski D.M. Da Silva O. Krieger M.A. Auslander D.J. Edelsohn B. Gamsa G.R. Gander P. McKenney M. Ostrowski B. Rosenburg M. Stumm J. Appavoo, K. Hui and J. Xenidis. Enabling autonomic behavior in systems software with hot swapping. In *IBM Systems Journal*, 2003.

[46] L.M. Browning J. Jann and R.S. Burgula. Basic building blocks for autonomic computing on ibm pseries servers. In *IBM Systems Journal*, 2003.

[47] I. Matta J. Liu and M. Crovella. End-to-end inference of loss nature in a hybrid wired/wireless environment. In *Modeling Optimization in Mobile, Ad Hoc, Wireless Networks*, 2003.

[48] G. Camarillo A. Johnston J. Peterson R. Sparks M. Handley E. Schooler J. Rosenberg, H. Schulzrinne. Sip: Session initiation protocol. RFC-3261, June 2002.

[49] Hari Balakrishnan Robert Morris Jaeyeon Jung, Emil Sit. Dns performance and the effectiveness of caching. *IEEE ACM Transcations*, 10:589–603, 2002.

[50] R. Jain. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling.* Wiley- Interscience, 1991.

[51] Schulzrinne H Jiang W. Modeling of packet loss and delay and their effect on. In *The 10th International Workshop on Network and Operating System Support for Digital Audio and Video*, June 2000.

[52] T. Towle K. Knightson, N. Motita. Ngn architecture: General principles, functional architecture, and implementarion. In *IEEE Communications Magazine pp.4956.*, Oct. 2005.

[53] J. Kephart and D. Chess. The vision of autonomic computing. *Computer Magazine*, IEEE:41–50, 2003.

[54] Brad Knowles. Domain name server comparison: Bind 8 vs. bind 9 vs. djbdns vs. ??? Technical report, Snow BV, 2002.

[55] Averil M. Law. How to build valid and credible simulation models. In *Proceedings of the Winter Simulation Conference*, 2005.

[56] Averill M. Law and David Kelton. *Simulation Modeling and Analysis.* Mc-Graw Hill, 1991.

[57] Zhen Liu Li Zhang and Cathy Honghui Xia. Clock synchronization algorithms for network measurements. In *In Proceedings of Infocom*, 2002.

[58] C. Liui and P. Albitz. *DNS and BIND.* Oreilly, 4th edition, April 2001.

[59] E. Scholler J. Rosenberg M. Handley, H. Schulzrinne. " sip: Session initiation protocol", rfc2543,. IETF,, March 1999.

[60] Simon Dobson M.A. Razzaque and Paddy Nixon. Cross-layer architectures for autonomic communications. *Journal of Network and Systems Management*, 15(1):13 to 27, March 2007.

[61] Lennart Maris. Infrastructure enum - implementation options for the netherlands. Master's thesis, Eindhoven University of Technology, 2006.

[62] M. Mealing. Dynamic delegation discovery system (ddds) part one : The comprehensive ddds standard. rfc-3401, October 2002.

[63] M. Mealing. Dynamic delegation discovery system (ddds) part three : The dns database. rfc-3403, October 2002.

[64] M. Mealing. Dynamic delegation discovery system (ddds) part two : The algorithm. rfc-3402, Orctober 2002.

[65] M. Mealling. Dynamic delegation discovery system (ddds) , part four: The uniform resource identifiers (uri) resolution application. rfc-3404, October 2002.

[66] M. Mealling and R. Daniel. The naming authority pointer (naptr) dns resource record. rfc-2915, September 2000.

[67] Dowdy L.W Menasce D.A, Almeida V.A.F. *Capacity Planning and Performance Modeling : From Mainframes to Client-Server Systems.* Prentice Hall, 1994.

[68] D. L. Mills. Network time protocol (version 3) specification, implementation and analysis, network working group. rfc-1305, 1992.

[69] Paul Mockapetris. Domain names - concepts and facilities. rfc-1034, November 1987.

[70] Paul Mockapetris. Domain names - implementation and specification. rfc-1035, November 1987.

[71] Evi Nemeth Nevil Brownlee, kc Claffy. Dns measurements at a root server. In *in Proceedings of IEEE GLOBECOMM*, 2001.

[72] A.Bradley P. Barford, A. Bestavros and M.Crovella. Changes in web client access patterns, characteristics and caching implications. In *World Wide Web*, 1999.

[73] Mark Schleifer Paul Vixie, Gerry Sneeringer. Events of 21-oct-2002[dos attacks on root name servers], November 2002.

[74] V. Paxson. End-to-end routing behavior in the internet. In *in Proceedings of SIGCOMM*, August 1996.

[75] V. PAXSON. On calliberating measurements of packet transit times. In *In Proceedings of the ACM SIGMETRICS*, 1998.

[76] Vern Paxson. *Measurements and Analysis of End-to-End Internet Dynamics.* PhD thesis, University of California, Berkeley., 1997.

[77] Vern Paxson and Sally Floyd. Wide-area traffic: The failure of poisson modeling. In *IEEE/ACM Transactions on Networking*, 1995.

[78] Harry Perros. *Computer Simulation Techniques : The definitive introduction!* NC State University, 2007.

[79] Anant Kumar Peter B. Danzig, Katia Obraczka. An analysis of wide-area name server traffic: a study of the internet domain name system. In *in Proceedings of the ACM SIGCOMM*, volume 22, October 1992.

[80] Guy Pujolle. An autonomic-oriented architecture for the internet of things. In *John Vincent Atanasoff Symposium (163-168)*, 2006.

[81] Strohmann Rosalind Stevens. Consumer protection and quality ofservice (qos), including network neutrality and cyber-security issues. OFCOM, March 2007.

[82] Salama H. Squire Rosenberg, J. M.: Telephony routing over ip (trip), draft-ietfiptel-trip-04.txt, November 2000.

[83] Schulzrinne H. Rosenberg, J. A framework for telephony routing over ip, June 2000.

[84] Kavé Salamatian and Sandrine Vaton. Hidden markov modeling for network communication channels. In *ACM SIGMETRICS Performance Evaluation Review*, 2001.

[85] Paul J. S´anchez. As simple as possible, but no simpler: A gentle introduction to simulation modeling. In *Proceedings of the Winter Simulation Conference*, 2006.

[86] Thomas BUGNAZET Sandoche BALAKRICHENAN and Monique BECKER. Studying enum performance with modeling and simulation. In *Proceeding os the Asian Modeling Symposium*, 2007.

[87] Donald F. Towsley Sara Alouf, Philippe Nain. Inferring network characteristics via moment-based estimators. In *INFOCOM*, 2001.

[88] S.Balakrichenan, T.Bugnazet, and M.Becker. Studying enum performance with modeling and simulation. In *Proceedings of the Asian Modeling Symposium*, March 2007.

[89] Paul Skelly Sue B. Moon and Don Towsley. Estimation and removal of clock skew from network delay measurements. In *Proceedings of IEEE Infocom*, March 1999.

[90] A. Brusilovsky V. K Gurbani, X.H.Sun. Inhibitors for ubiquitous deployment of services in the nextgeneration network. In *IEEE Communications Magazine, , pp. 116121*, Sept. 2005.

[91] Paul V.Mockapetris. Telephony's next act. In *IEEE Spectrum*, April 2006.

[92] Don Towsley Wei Wei, Bing Wang. Continuous-time hidden markov models for network performance evaluation. In *Elsevier Science*, 2002.

[93] Duane Wessels. Is your caching resolver polluting the internet? In *In Proceedings of the ACM SIGCOMM workshop on Network troubleshooting*, August 2004.

[94] W.Wimreuter. Elements for transition : Enum and p2p complement the beasts. 44. DFN Betriebstagung, February 2006.

[95] S. Parekh Y. Diao, J.L. Hellerstein and J.P. Bigus. Managing web server performance with autotune agents. In *IBM Systems Journal*, 2003.

[96] D. Karrenberg G. J. de Groot E. Lear Y. Rekhter, B. Moskowitz. Address allocation for private internets. rfc-1918, February 1996.