SCALABILITY ANALYSIS OF A NEW INTERNETWORK NAMING AND ADDRESSING ARCHITECTURE

Projecte Final de Carrera d'Enginyeria de Telecomunicació

Jordi Pujol Teixidó

Supervisor: Dr. Marcus Brunner

Professor: Dr. Xavier Hesselbach Serra

Escola Tècnica Superior d'Enginyeria de Telecomunicació de Barcelona (ETSETB)
Universitat Politècnica de Catalunya (UPC)

I would like to thank the following members of the Next Generation Internet Group at NEC Network Laboratories (Heidelberg, Germany) for having granted me the opportunity of carrying out my Master Thesis with them and for all the support that they provided to me during those months.

Dr. Stefan Schmid
Dr. Lars Eggert
Dr. Marcus Brunner

TABLE OF CONTENTS

1. INTRODUCTION

This dissertation focuses on the scalability analysis of a new naming and addressing scheme for Internetworks, which is the core and most representative infrastructure of any next generation Internet architecture.

The nowadays Internet has a proofed lack of autonomy and flexibility. For instance, it does not provide any in-built support for mobility, dynamical network composition, multihoming, multisourcing, heterogeneity of networks… and many other features that the final user would expect from it. Therefore, and instead of offering transparent solutions, the system and network administrators have to perform misusages and patches that often break the protocol stack and the architectural rules. The main one among all these limitations, as it is the reason for the others, refers to the naming and addressing scheme for the architecture, which is actually unable to distinguish between identities and topological locations of system elements.

This situation renders the Internet inappropriate for the expansion of both the business and the private use of the network. The limitations imposed by the network do not allow a smart and innovative usage of the resources.

For instance, in Figure 1, a laptop is connected to the network through a wireless LAN hotspot. In addition to this, a second laptop is interacting with the first one through a Bluetooth *ad hoc* network. Even though it is rather unlikely, the second laptop might be using an upper layer application to access the Internet through the first one. However, the networks themselves do not provide any transparent support for this case of network composition. In an ideal scenario, the second laptop could be able connect to the Internet, if allowed by the first one, and make the same use with the same capabilities as if it had a direct connection.

In a second scenario, the network of a train is connected to the Internet through an ISP. Due to the intrinsic mobility of the train, its network looses connection with the first provider and enters the influence zone of a second one. However, the connection is lost as there is no interoperability agreement between providers. These and other items like roaming or

management of handovers, which are related to the mobility, cannot be handled by the Internet itself.

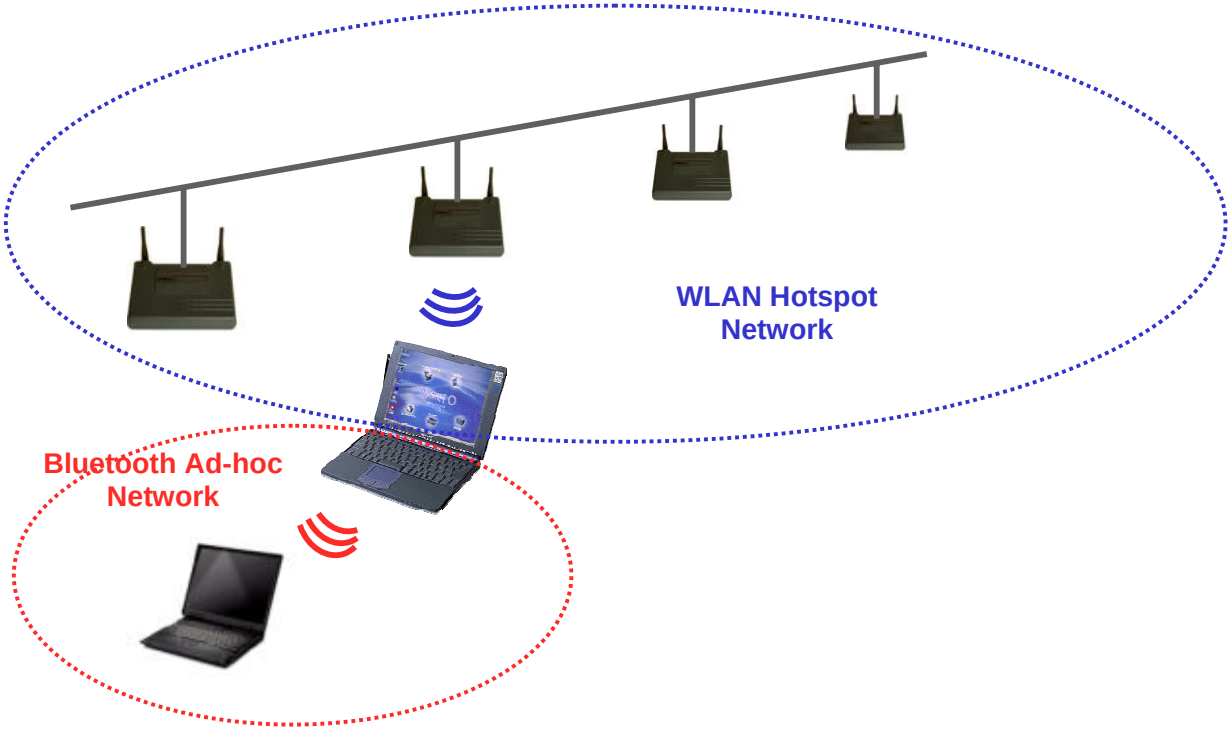

Figure 1.1. Network composition.

As a response, the scientific community claims the necessity of a new evolving Internet with added capabilities to handle the requirements of the information society. One of these proposals is the TurfNet architecture, developed at the Network Laboratories of NEC Europe Ltd. at Heidelberg, Germany. The TurfNet makes use of a hierarchical registration, lookup and resolution scheme for identities while the addresses, which become decoupled, are handled locally. Furthermore, all these procedures follow natural user data flow like paths and have a one-step performance (while the location of an address in DNS typically requires several iterations). Such ideals and characteristics are commonly accepted and assumed by many other next generation Internet architectures.

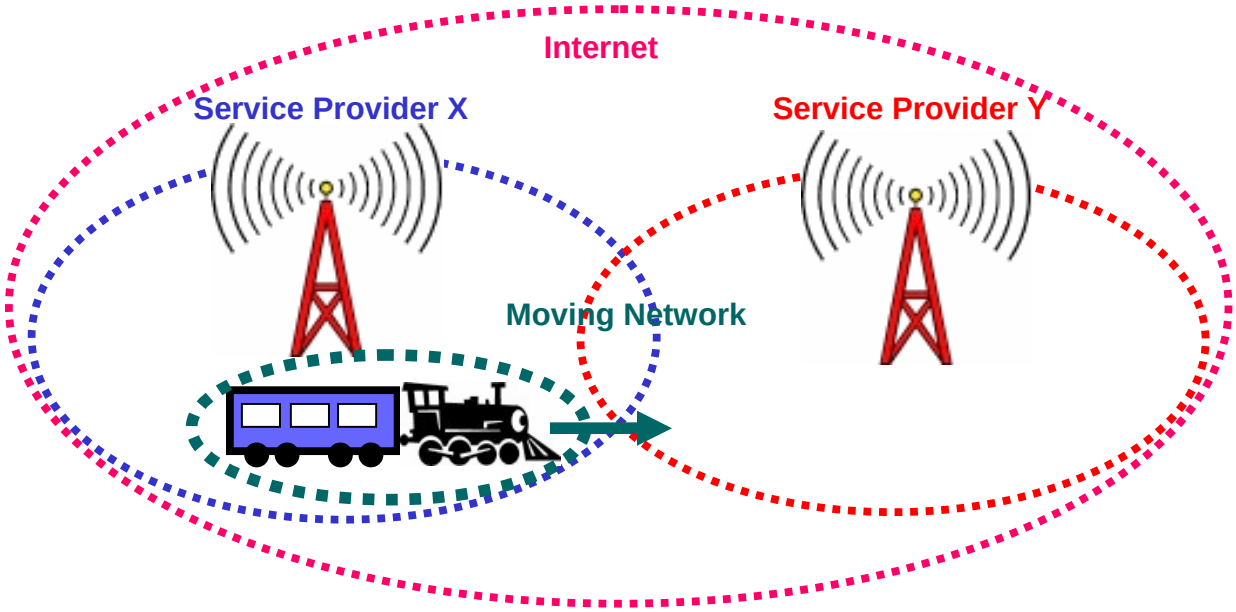

Figure 1.2. Scenario of user mobility.

Nevertheless, and although it could seem paradoxical, some of these new Internet architectures could not be able to supplant the current Internet. The all of them share an in deep convincement that main structural changes have to be applied to improve the supported network features. Furthermore, most of the solutions and strategies that they suggest are common, related, evolutions from one another, or could be even complementary. However, even though they have proofed to operate in small-scale networks, they might find an impassable barrier when trying to handle such a huge structure as the real Internet. The operations that they define have to be large scale certified.

The key contribution of this work is a novel scalability analysis methodology to test the feasibility of deploying a next generation Internet architecture. It allows measurements on registrations, lookups, resolutions… in concordance with the naming and addressing structure that rules the architecture. Such an analysis provides quantification for the possible bottlenecks and weaknesses of the system, and a criterion to improve the architecture itself, as a result of the obtained feedback.

First, this thesis meant a study of the Internet characteristics in order to develop this work. On one hand, the real topology of the network in terms of Autonomous Systems provides the scenario in which to check the new protocols in an Internet like environment.

On the other hand, a communication pattern models the likelihood of connection between hosts that are at a certain distance in the network.

The statistical and matrix like characteristics of these data suggested the usage of MatLab as the programming language to final implementation of the system.

After providing the necessary mathematical definitions for a generic view of any architecture, this dissertation chooses a hierarchical approach for the naming and addressing scheme. This solution resembles the TurfNet architecture, which is, actually, the scenario of final exemplification of the methodological procedures herein developed.

As a first result, and concerning the evaluation of this concrete proposal, it shows to perform efficiently under the proper conditions. Thus, the TurfNet turns up as a feasible architecture for an Internet size. In addition to this, all its possible scenarios of application at lower scale become automatically proofed.

Apart from this, and as a second result, this dissertation suggested some improvements on the proceedings of the architecture, related to the horizontal composition of networks through the peering connections, and the vertical composition through the provider to customer ones.

Finally, notice that no other previous research study had checked the scalability behavior of networks in this way before. Hence, and as the main research advance of this thesis, it presents a brand new approach from which any new proposal for a new Internet scenario could beneficiate by taking it as a starting point for scalability purposes.

1.1- TASKS

The required tasks that were carried out to develop this thesis were organized according to the following schedule:

- Study the status of the new Internet architectural proposals.
- Study of the TurfNet architecture and its related possible working lines.
- In detail study of the Internet characteristics.
- Model proposal.
- Installation of the working environment and learning the syntax of MatLab.
- Implementation of the model in MatLab and obtaining preliminary results.
- Propose improvements for the TurfNet.
- Final results for the TurfNet.
- Write a TurfNet related paper, [PUJOL2005].
- Abstraction and generalization of the work, as an extension of the study.
- Refinement of the model and new results.

1.2- OUTLINE

After this brief introduction on the contents of this thesis, Chapter 2 gives a detailed overview of the next generation Internet architectures and their theoretical principles. This includes sorting out what the Internet lacks for the future user requirements. Then, it shows the brand new functionalities that are added to the system and their effects over the network performance. Among them and as its key issue, the necessity of a novel naming and addressing scheme.

Chapter 3 discusses a particular case of these new architectures: the TurfNet. It was the starting point for this work and, thus, it deserves special attention and a separate chapter. Furthermore, it provides an example on how do the new Internet proposals manage to overcome the limitations of the nowadays Internet. In addition to this, some of the results of the thesis were used as a means of feedback for the analysis and improvement of this architecture. Hence, there is a personal contribution therein.

Chapter 4 models the structural naming and addressing architecture in terms of already existing elements of the Internet. Thus, this includes an in detail study of the Internet characteristics, that was one of the main task of this thesis. In the last section of this chapter, there is the final mathematical description of the system.

Chapter 5 implements the previously described model. Due to the high amount of operations that are involved in the simulations (as they apply to the whole Internet), this chapter pays special attention at using strategies to reduce the running time for the final pseudo code.

Chapter 6 discusses experiments that evaluate the performance of the naming and addressing architecture under different values for the possible parameters. Moreover, it suggests and validates some strategies to achieve a higher level of performance and reduce the operational costs in the possible bottlenecks of the architecture. The chapter concludes that the system is feasible.

Chapter 7 opens several discussions on items that apply to the field of Internet architectures. This includes, mainly, suggestions for real implementation of some technical aspects of the system that would be necessary when deploying the architecture in a real scenario. Furthermore, it specifies some future work.

Chapter 8 covers related work in three main areas. The first one is this scalability analysis itself. It can be told in advance that no other similar study had been done yet by any other of the next generation Internet proposals. The second area is that of the Internet characteristics analysis. At last, we discuss other work related to the creation of a new Internet.

Finally, Chapter 9 summarizes and concludes this work.

2. NEXT-GENERATION INTERNET ARCHITECTURES

2.1- INTRODUCTION

This chapter lists several main items that render the nowadays Internet inappropriate for the new requirements of the society. Additionally, it introduces new concepts that would allow a next generation architecture to overcome any current lack, which have their basis on the naming schemes.

[CLARK2002] claims and demonstrates that in the current Internet there are multiple scenarios driving into tussles. These fights are the result of the incompatibility of different elements that were not considered in the original design of the Internet, as it was not business oriented. Nowadays, the network has become popular and now that these new commercial, governmental, community and user scenarios have appeared, the architecture is neither able to fairly provide the proper resources to each situation nor to avoid the collision of some of them. The user expectancies can be hardly ever accomplished.

The characteristics and definitions showed in this chapter have been mainly extracted from the design principles manifested by [BRADEN2000], [BALAKRISHNAN2004], and the Ambient Networks project [AMBIENT] (co-sponsored by the European Commission). The remaining is a brief of them.

2.2- NAMING VS ADDRESSING

According to [AMBIENT], naming is a function that contributes to the overall internetwork. The new models are based on the decoupling of identifiers and locators. While identities allow naming, the locators are used for addressing; *i.e.*, identities are typically resolved into locators, which are addressable.

This brand new approach provides the architecture with the new required features. For instance, the multihoming and multisourcing can be achieved once an identity has several concurrent locators.

Furthermore, the decoupling has inherent mobile (both geographical and network related) concerns. As the identity is not attached to a fixed address, it is granted the ability of changing location by acquiring an additional locator. Otherwise, the same element would be bound to the fixed address space and confinement of its network subset.

Performing this distinction between identities and locators is already considered and applied by architectures such as TurfNet [SCHMID2004], DOA [WALFISH2004], HIP [MOSKOWITZ2005], multi6 [ABLEY2003], and SNF [JONSSON2003] among others.

The identification requires defining which elements of the topology have to be identified (*i.e.*, the scope of this nomenclature). The final choice over these elements can have implications on the naming and addressing characteristics and schemes that can be used. Therefore, this should be one of the first decisions to be taken by any architecture designer.

The possible identifiable objects are:

- User or organization identity. Provides legal identification.
- Subscriber. It considers service payment issues.
- Network identity.
- A general purpose persistent over time service or data identity.
- Session identity.
- End-point identity. It can be a user, a network card,…
- Locator or routable address. The other identities will be resolved into these locators.

It is quite noticeable that this list has a high level of abstraction behind. Hence, it can apply to any concrete architecture. In some cases, not the all of them require being identified as a mandatory condition.

On the other hand, the naming functions that apply are:

- Name translation, to allow joining a network with different address space or protocol.
- Name resolution, to obtain physical locations out of identities.
- Name allocation, to assign identities to objects.
- Name registration, to assign locators to identities.

2.3- COMMON PRINCIPLES

This section briefly enumerates the most common possible axioms of any next generation Internet architecture*:*

- Packet switched networks, for its ease of management and its performance efficiency.
- Usage of a global namespace for the identification of networks.
- Flexibility in business models: allow dynamic composition of networks based on autonomic performance.
- Autonomous network domains, which break the architectural homogeneity.
- Inter-domain control interface, as network control for the naming functions must cross multiple domains. Additionally, its management must be distributed.
- Remove the usage of NATs as the architectural solution to both multiplex addresses (which renders the system unable to support detaching identities from locators) and to hide internal network information (this breaks the execution of some protocols, such as FTP, and the packet integrity).
- In-built mobility proceedings, for both entities and whole networks.
- Ability to delegate the communication to third parties.

2.4- PROPOSAL FOR A HIERARCHICAL APPROACH

As shown in this chapter, the main elements that define any architecture is the identification system. The rest of the structural elements are defined accordingly to take the

maximum profit of the system resources. Hence, the procedures related to identities and addresses can be extracted and treated as the core of the new architecture. Any other consideration would be a particularization of the system due to some conditionings. In an extreme case, these conditionings can even turn into impediments and limitations.

Apart from introducing this scalability analysis, this dissertation aims to show the performance of hierarchically based naming and addressing architectures. They would considerably differ from the nowadays structure, with a non-decoupled address that is used to identify the host. Additionally, the system uses a hard state approach that limits the mobility and composition of its elements. On the other hand, the new hierarchical scenario would allow a high degree of flexibility by providing clear registration, lookup and resolution of identities into addresses for the communication process. These procedures would follow a natural data flow like path, instead of the non-conventional method that DNS applies (which is somehow, as an overlay network on top of the real topology).

Chapter 3 provides a description of a next generation Internet architecture that perfectly fits these hierarchical schemes: the TurfNet. Moreover, having an example is helpful in order to provide a better understanding of all these new concepts. Chapter 4 provides the definitions to analyze the scalability of any architectural proposal when deploying a large Internet-like network. Its generic behavior is finally instantiated by a hierarchical solution that is the main basis for TurfNet, among other similar architectures. Actually, TurfNet is the main example over which the system is implemented in Chapter 5.

3. TURFNET

3.1- INTRODUCTION

The herein presented scalability analysis generically applies to any new architecture that follows hierarchical rules for the addressing and looking up schemes. However, its main application is for the TurfNet architecture. TurfNet is one among the proposals that are being worldwide developed attending the requirements of a new Internet that the current structure relying on IPv4 is unable to handle.

TurfNet has been developed at NEC Europe Ltd. Network Laboratories, Heidelberg, Germany. Moreover, TurfNet is a candidate for Ambient Networks [AMBIENT], and thus partially funded by the European Commission under its Sixth Framework Programme.

This chapter provides the basic guidelines of the TurfNet. For a full understanding of the architecture and its particularities, it is strongly recommended to review [SCHMID2004] and [SCHMID2005].

3.2- ARCHITECTURE

A TurfNet is defined as a completely autonomous network domain.

As we also call TurfNet to the whole architecture, in order to distinguish both, a single TurfNet is also called Turf, if required.

The key point of each Turf domain is the autonomy. To achieve it, each Turf is able to take care of its local communications. This includes local naming, addressing, routing, resolution… features, as well as all the elements that provide control and added functionalities over them.

Even though all these characteristics are locally handled, not the all of them are. There must be a global naming space to allow interconnection among different Turfs. In this global

space, the elements of the architecture (from here on hosts, even though as explained in Chapter 2 there are different entities) are uniquely identified. Additionally, the operations between Turfs have to be running a common protocol to enable handshaking and interoperability. However, this excludes the nowadays global protocol IPv4 or any other global constraining agreement.

Regarding the lack of a global addressing space, this implies having to apply several mechanisms to map identities in the different Turfs, as explained later.

Additionally, and for business related purposes, Turfs are able to hide what happens inside their boundaries from the rest of the network: their structure, the number of nodes, their interconnections, if their have other sub-domains below… Therefore, rival companies can compete to offer their services in equality of opportunities and, thus, promoting the competence.

Furthermore, the hosts themselves are able to choose whether they want or not to be fully reachable. This means that they can choose if they want to communicate and, thus, to be located from outside (if allowed by the control mechanisms) and even up to which extension. Therefore, they could check to be accessed only in the subsidiary location of the company where they belong and in an upper Turf of the whole corporative site. However, neither another location nor other Turfs from the Internet would be allowed to communicate with that host.

Somehow, this behavior comes already granted by NATs. However, as suggested in Chapter 2, a new Internet architecture like TurfNet must be able to overcome the implicit limitations imposed by the use of NATs.

3.3- INSIDE A TURFNET

The elements of the TurfNet, as shown in Figure 3.2, are:

- TurfControl. It is defined as the both logical and local element of the network that takes care of the operations. We must remark that they are logical, which means that they are not necessarily unique. This is especially interesting to allow the system to have resiliency, as explained in Section 7.4.2. It means that there could be several

TurfControls running in a coordinate way if necessary due to size of the Turfs and number of operations that they have to lead/supervise. Actually, this is the only brand new element that this architecture incorporates. The others already exist in the current Internet or are extensions of those.

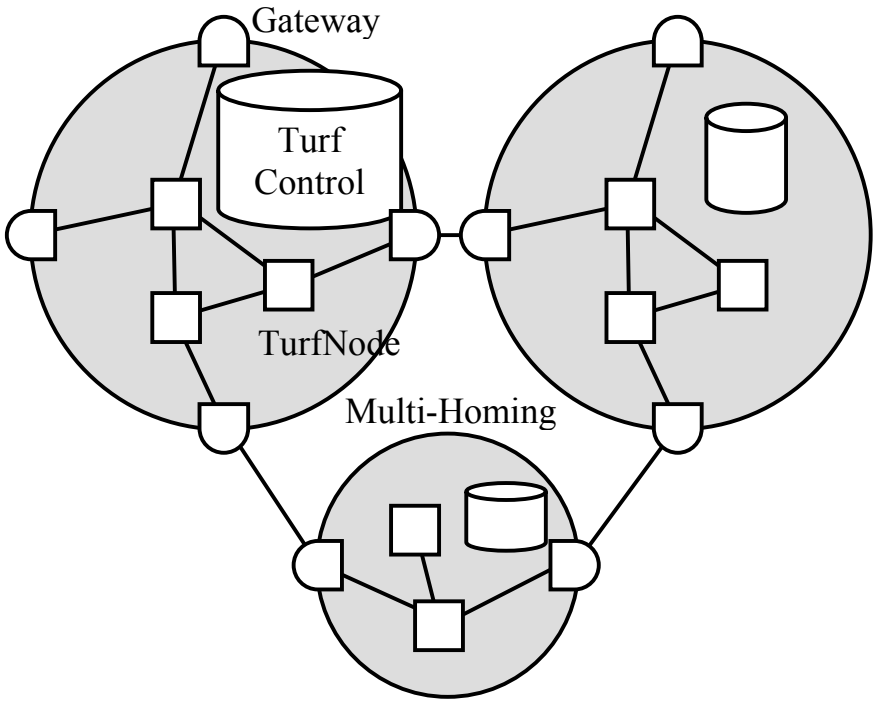

Figure 3.1. Elements of a TurfNet.

It has a dual aim. On one hand, it handles the usual functionalities of a common IPv4 sub-network (address allocation, routing, name resolution…). On the other hand, it must take care of the added functionalities.

- TurfNode. It is each of the entities that are granted a global identity belonging to the global naming system and, thus, are potentially allowed to communicate with other TurfNodes in the Internet. Each TurfNode belongs to a Turf, even though a real network element (*i.e.*, a PC or a user) might have several TurfNodes allowing, thus, multihoming and multisourcing. The communications inside a Turf for any TurfNode

applies to the local policies regarding addressing, location, routing, protocols… according to the local TurfControl.

- Gateways. These are a special type of multihomed network elements. They are not only in two Turfs at the same time (which means that they are fully capable in them for communication) but they also allow traffic from both to transit through them. In this way, they allow inter-Turf communication. These gateways must then provide translation for address space and even for protocols. However, these translations might even not be required. Both Turfs could be using the same protocol or could be sharing a common addressing space in a non-aggressive way. In that situation, the gateways would act as mere routers. Additionally, the TurfControls, as members of the Turf, should inter-communicate through the gateways for the overall protocol to work.

3.4- NETWORK COMPOSITION

As a key point of the improved autonomy, TurfNets can cooperate between them. Turfs communicate through their gateways as they share links among the all of them.

These connections are not static. Hence, the Turfs can dynamically associate or dissociate to provide a means of flexibility to the users and organizations. These functionalities are handled by the system in a totally transparent way.

Obviously, the physical links have to be present (even though the media can be an air transmission via radiofrequency). Therefore, the connections are already present. The TurfNet architecture, through the TurfControls, is able to decide, according to the preferences and policies of the organization, whether to make use of them or not in order to compose with the remote Turf.

Depending on the sort of usage that is made out of the inter-Turf connections, the relations between Turfs can be divided into *vertical composition* and *horizontal composition*, as shown in Figure 3.2.

Finally, Turfs have also the ability of merging into a new Turf.

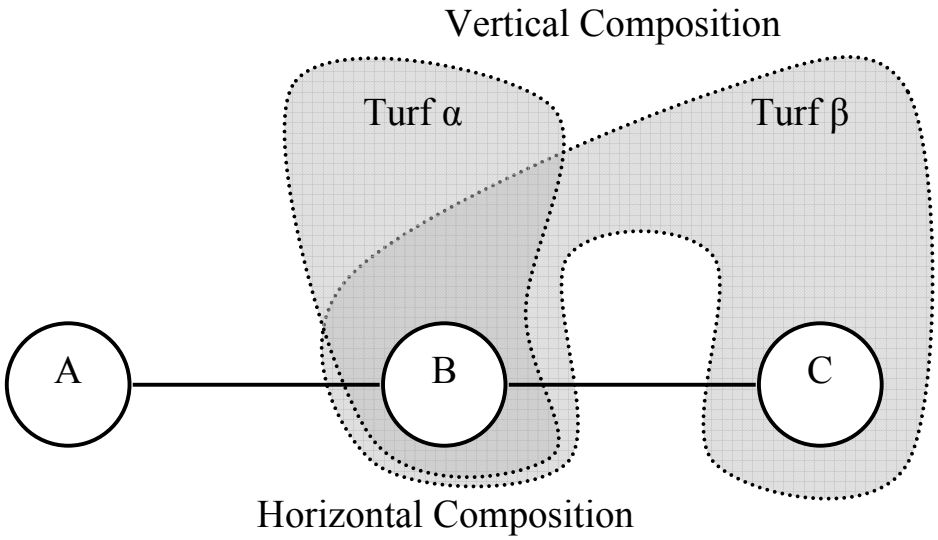

Figure 3.2. Vertical and horizontal composition of networks.

Figure 3.2 shows an abstract example of five TurfNets that compose. α is composing vertically with B. β also composes vertically with B and C. Additionally, B shares horizontal relations with A and C at the same time.

3.4.1- VERTICAL COMPOSITION

Vertical composition is the main mechanism in the TurfNet architecture that provides inter-Turf communication. While the other two are optional, it is mandatory, due to the registration process scheme, that a Turf plays the role of a customer to, at least, a provider Turf. However, a customer Turf will typically have more than a single provider.

Providers will act at their turn as customers for some others. In this way, the provider to customer relations that translate into the vertical composition, weave a hierarchy of Turf levels

where Turfs are located depending on their functionality and degree of interoperability with the others.

As another assumption, there must be a single provider Turf on top of the all of them. This will be justified later on in this chapter when explaining the registration and resolution processes.

The main advantages of the vertical composition are the encapsulation of administrative, control and routing functionalities, as well as isolation of internal structures. The resulting structure will be, furthermore, able to be distributed into more than two single levels, as commonly done. Thus, it overcomes another restriction.

Note that after the feedback provided in Chapter 6 thanks to the analysis and the obtained results, these concepts were redefined and improved.

3.4.2- HORIZONTAL COMPOSITION

The concept of horizontal composition was not initially considered in the basic guidelines of the TurfNet architecture. It appeared as an output during the realization of this work. When studying the characteristics of the current Internet structure (Chapter 4), it showed up that the network domains that shape it (Autonomous Systems) do not only possess vertical-like relations, but in addition to this, they also connect with others that are at the same height, their peers.

Hence, the concept of horizontal composition was abstracted out of the Internet.

Through horizontal composition, Turfs are able to communicate with their neighbors and share information. This information can be roughly used in two ways:

- A Turf can spread its information concerning TurfNodes that have been registered in it
- Otherwise, when a Turf tries to locate a TurfNode, it can propagate the necessary requests to the peers

As an architectural agreement, the information that starts spreading through horizontal

composition does not follow any hierarchical path later. In this way, it avoids overloading the upper levels with an excess of redundant information.

Finally, the Turfs can decide up to which point the information will be forwarded. In the vertical composition, an analog proceeding allows a TurfNode to be reachable up to certain point through its providers. This allows the system to avoid a waste of resources.

A scope parameter provides this means of control on the horizontal composition.

3.4.3- NETWORK MERGING

The dynamic behavior of the TurfNet architecture allows a third possibility. Two Turfs can *merge*. In this way, both Turfs turn into a single network that controls the resulting internal communications and takes care of the external ones.

This implies a unified TurfControl, a single internal communication protocol and a non-colliding address space. It can be done by either prevalence of the protocol and address space of one of both or by an agreement into some other protocol and space.

After a network merging, it is rather unlikely that the two Turfs are able to recover their initial shape. Actually, the process is virtually irreversible as no information is necessarily kept regarding the original Turfs.

3.5- BASIC OPERATIONS

The basic operations of the TurfNet architecture are registration, lookup and communication process. For plenty of details, refer to [SCHMID2004] and [SCHMID2005].

3.5.1- NODE REGISTRATION

Once a host is granted an identity in the TurfNet, it can register in the structure to be

globally reachable by others.

As a basic rule, and unless a node is to be registered just up to a certain point, a customer Turf will propagate the registration of its registered TurfNodes to its providers. In this way, the TurfNode obtains an entry in the local registers of all provider Turfs up to the top and all the necessary state for translation in the Turfs that are being crossed.

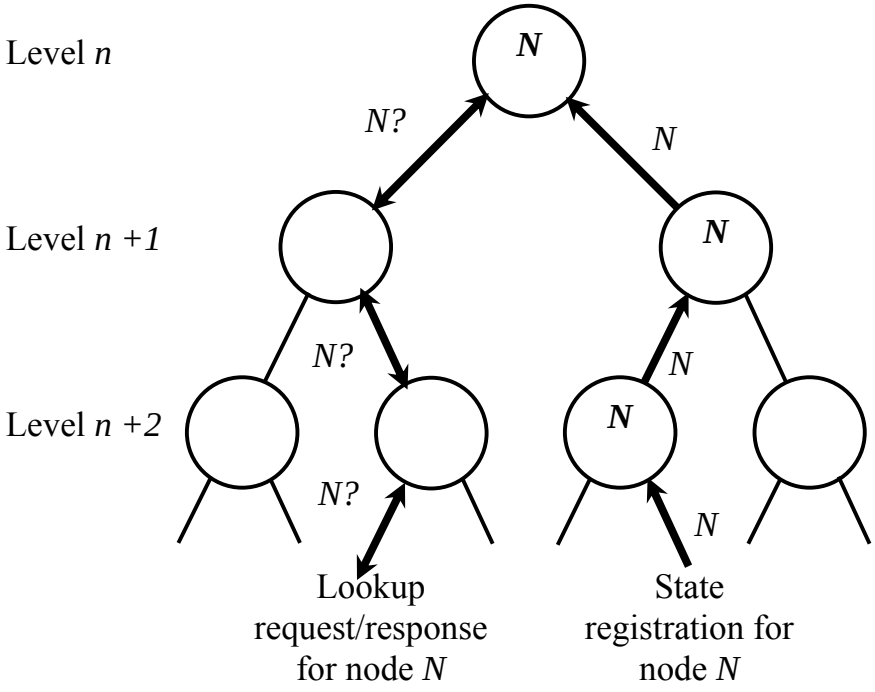

Figure 3.3. Typical lookup resolution and registration procedures.

Hence, the system is clearly hierarchical. Figure 3.3 shows a basic theoretic scenario for registration. Node N submits its registration request to the local Turf. Then, this local Turf forwards the request so that the TurfNode is registered in all the parent Turfs in the hierarchy up to its top.

Figure 3.4 is the starting point of an example to illustrate the operability of the architecture. Node A asks for registrations to its local TurfControl. As the local Turf is a customer to another Turf, it has to submit the registration query to its provider in order to allow

node A to be fully registered. The TurfControls of both Turfs take care of this operation. This is shown in Figure 3.5.

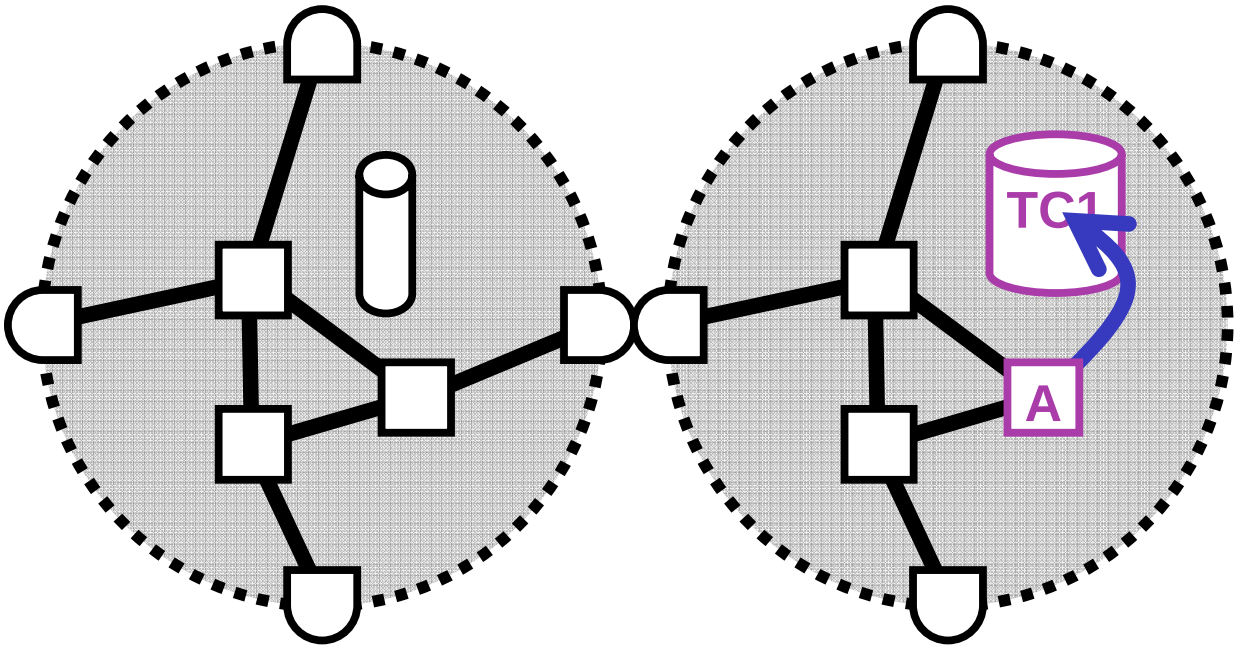

Figure 3.4. Registration.

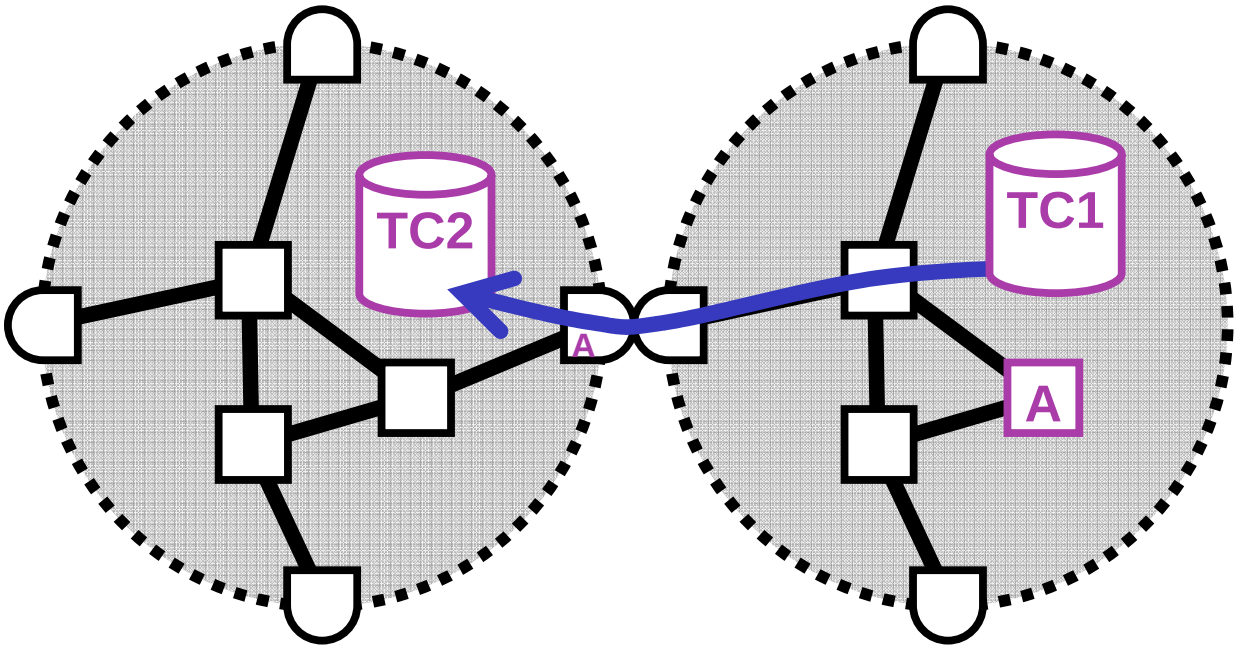

Figure 3.5. Lookup request for connection and creation of state.

3.5.3- NAME RESOLUTION

When a TurfNode tries to locate another node, it first performs a local resolution. If not possible to find it, the TurfControl takes care of the enquiry and submits it up through the hierarchy as shown in Figure 3.3. Once the location of the identity has been resolved, the solution returns to the seeking TurfNode.

As the search is hierarchical, only those nodes that have registered up to the top provider will be globally accessible by any other. In addition to this, the system should typically have a single provider, as all the registering information must fit into a single Turf able to resolve the requests that lower levels could not answer.

Nevertheless, the horizontal composition will allow (as shown in Chapter 6) overriding the usage of a unique top provider. In addition to this, the peering connections of the horizontal composition can improve, as shown in Figure 3.6 (a more realistic scenario), the performance of the lookups for resolution of identities.

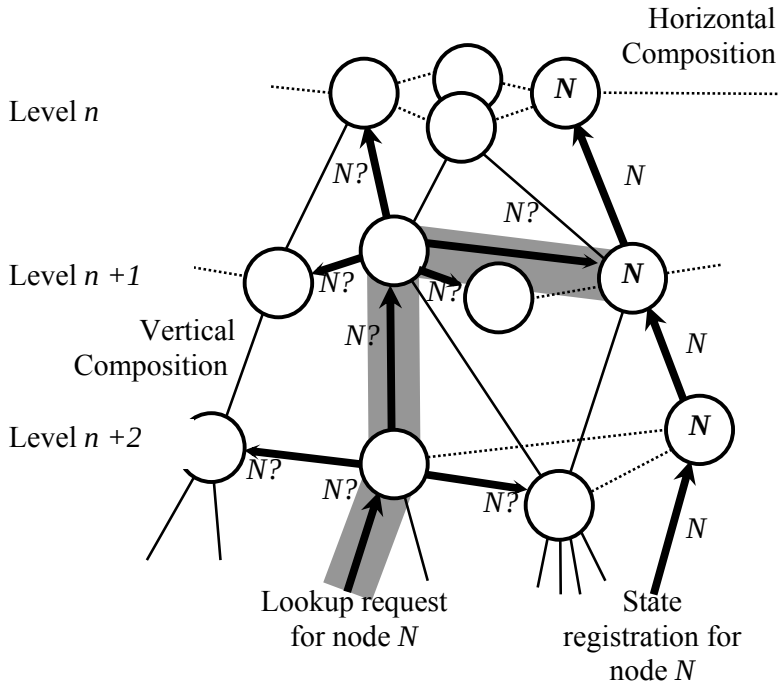

Figure 3.6. Optimization for the lookup and resolution process.

Back to the application example in Figure 3.7, now the TurfNode B, belonging to the provider, asks for the location of the A identity. As A was already registered in there, the TurfControl is able to resolve the location. The communication of the TurfNode B is redirected to the Gateway, where the necessary translation state for B is created.

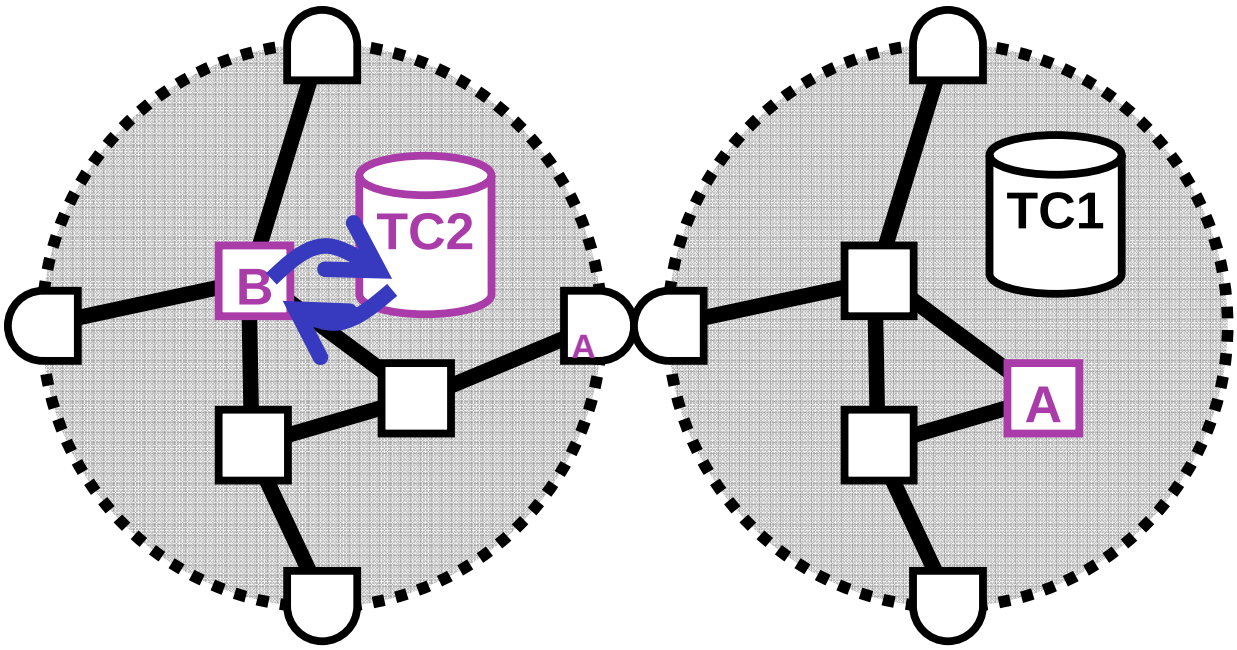

Figure 3.7. Resolution of the request.

3.5.3- PACKET RELAYING

Finally, once all the necessary state has been created through the communication path, the data flow between nodes in both Turfs can start. The necessary protocol and address translation will be provided by the Gateway.

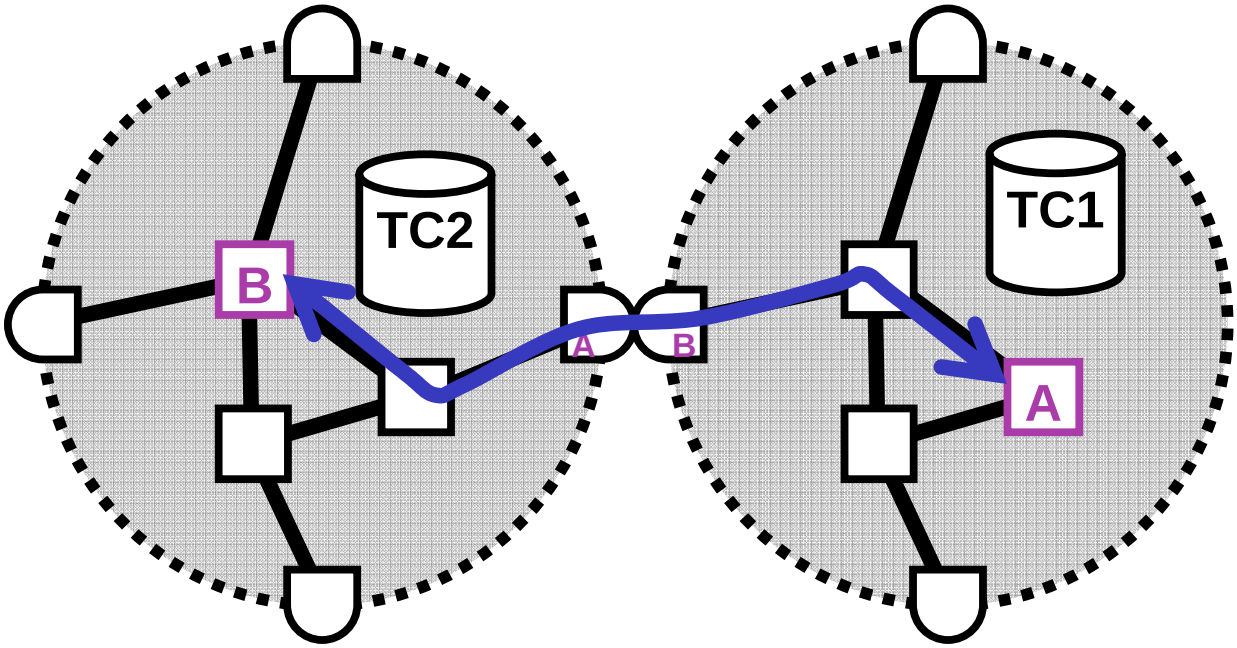

Figure 3.8. Creation of the additional state and begin of the communication data flow.

4. MODEL

4.1- INTRODUCTION

The main issue of this chapter is discussing how to model the overall Internet in a way that it enables the simulation of the key aspects of any new architecture. Therefore, it must be able to support all key elements it has and, on the other hand, get rid of those aspects from the protocol which are irrelevant in this scalability approach (*i.e.*, those which are an important part of it when implementing the whole architecture while not when performing a scalability analysis).

First, this chapter provides an obvious justification on why to perform a simplification of the Internet as an initial stage, instead of performing one of the usual protocol development related solutions, which are:

- Running a relatively small implementation in a laboratory.
- Developing a version of the protocol under a well known and commonly accepted by the scientific community simulation environment, such as OPNET, Network Simulator,… and building the required scenarios by this analysis to run in there.
- Having a simulation scenario like the one built but including all features allowed by the protocol.

Second, it explains the way in which this model is built. For this to be done, it required an in detail study of existing work related to the Internet structure. Several algorithms and data were considered to address the requirements. This mainly involves being able to fit the architectural proposals into the model which derives from the study. Moreover, the key elements of the model had to be analyzed. In this way, they could be used as the system parameters. The resulting behavior of the model will depend on how these parameters are modified to perform the analysis.

Then, it summarizes the parameters of the analysis. The changes performed over their values/typologies allowed this thesis to validate the dependency of the outputs on each of them, as shown in the following chapters. Furthermore, the metrics that were chosen to represent the system are introduced and it is provided a justification on why they are the meaningful ones and not others.

It has to be pointed out that until this moment no assumption over the new Internet architecture is taken at all. Therefore, this whole developed methodology would apply to any other new approach for the Internet. Once here the TurfNet architecture features are introduced in order both to test it for scalability and to check the methodology itself for any other case. This means that the obtained results do not only apply to this concrete new Internet architecture but to any other which might be roughly similar as well. To do this, the existing analogies between the discovered-by-the-study Internet key elements and the TurfNet definitions have to be checked. These parallelisms drive into a set of translations between the general terms and those that apply to the considered case. For instance, and as shown later, the implicit particularities of TurfNet imply having to introduce some new elements into the model for the scalability study to come to terms with the TurfNet and vice versa.

At last, it provides the background for the real implementation of this analysis under code in accordance to Chapter 5. As explained, it was made under the MatLab7 environment due to its matrix treatment facilities for dealing with the data sets, which made it the preferred candidate for these mathematical purposes.

4.2- PERFORMING SIMULATIONS

Running an analysis of a new protocol (or even a single part of it) provides a means of checking its feasibility and obtaining the required feedback for the redefinition of some of its features. Therefore, there are several clear benefits in this process. However, the steps that have

to be taken can differ from one sort of study to another both related to the same architecture. The main reasons for this are the existing implicit constraints in each different research line.

In the case being focused on, which is scaling, and apart from the description of the architecture, the main guidelines are just some questions such as: “Does the X architecture scale?” or “Up to which point is it feasible to scale it?”.

In order to not waste resources, any step in the definition of the process has to be looked through a pragmatic prism: “Do the steps required in these proceedings turn into noticeable benefits regarding the quality of the results?”.

In this way, this drives into the final criterion that allows judging any possible parameter, metric, constant, assumption… up to the level of wondering which the most suitable environment for this work is. This objectivity is applied in the remaining to discuss the different proposals for building a solution.

While a small implementation of any next generation Internet architecture can be running in a laboratory or in an already existing simulator, this would be absolutely useless to provide any clue on how this new architecture would work under the whole Internet. This means, up to which point it scales. Therefore, a simplified model has to be used for any different Internet-like approach. This model must include all the required elements and their interrelations. Furthermore, all necessary parameters must fit into this model, which must be suitable for extracting the proper results.

As already commented before in the introduction of this chapter, there are three main approaches to face and perform any network viability study. The following paragraphs provide a brief on each of them, their pros, their cons and the reasons for which they were useless or at least unsuitable due to the inner characteristics of this dissertation, which were an implicit result of scaling.

a.) Laboratory implementation

This requires coding the new features of the architecture and running it under different situations to extract the conclusions. The main benefits of a proper implementation are having exact values. In addition to this, the implementation can be reused for other purposes such as having a demonstration of the protocol or performing other studies within it. However, on the other hand:

- It loses sight of the pursued results. Even though getting plenty of them, only a small fraction would be of interest. They would need to be combined and analyzed. This also means that while all the features have to be implemented to have it running only some results will be of interest regarding scalability.
- The amount of machines to get to the high numbers is not feasible. A small implementation can be built and even a medium size one, but not a whole Internet-like scenario. Larger values should be guessed and, hence, might be erroneous.
- As the scenarios are set by hand the complexity increases with the size. Moreover, many of them would be required to ensure that the results are not just a matter of a partial view. Some others could not be easily held.
- The results would be valuable for small size cases, but not related to scaling. Guessing values for a more complex and realistic Internet can be an option. Nevertheless, errors increase as the guesses go further.
- Limitations due to the operating system used and the existing functionalities and facilities that it supports can make the code not as efficient as expected when moving the architecture from the theory into the practice. In addition to this, the final implementation would also depend on the programmer skills.
- Other elements apart from the kernel would introduce interferences into the values: processors, memory, time to switch memory space, links, routing policies… And they would become unpredictable as the size of the scenario increases.

At last, the conclusion is that even though it has its benefits, it could not help in getting to the high numbers involved in scaling.

b.) Network simulators

Some already existing network simulators are commonly used to probe new protocol functionalities. This includes Network Simulator 2, OPNET, the recently appeared NCTU Network Simulator 2… Additionally, several extra packages allow creating large scenarios. Some are in-built and others are extra components that have been developed to handle this particular demand. These topology generators are usually based upon many research works showing the existence of power laws in the behavior of the Internet. By filtering traffic and observing different parameters, it has been shown that as networks grow the resulting graphs of their structures evolve and turn to respond to this sort of laws. Some values can be guessed to model these events.

Some of these topology generators take into account the connectivity degree between different elements of the Internet. A randomly generated network can be built by setting some related parameters like the final amount of nodes and the degree itself. Some others take into consideration other effects that have been observed in the Internet such as having levels.

The most well known ones are T-ITM, Inet, Tiers and BRITE.

After this brief on their basis, these are the main reasons for which to discard them:

- These simulators do not save having to program all the features from the architecture to have it running, while the field of possible results concerning the specific study is limited.
- Furthermore, you might have to describe the behavior or other layers of the protocol which even though being required are not finally related.
- Some already existing modules must be readapted. The new protocol will not work isolated but in relation with the existing ones. For example, it might be interesting to

validate several aspects of a new location system able to substitute DNS. This means that for any node trying to send traffic to any other, all the other protocols that are implied in the communication scheme would have to be rearranged, *i.e.*, getting rid of DNS itself and readapting HTTP, FTP,… to get a response from the new protocol prior to the emission of the user data traffic.

- There is a dependence on the chosen parameters for generating the random scenario.
- Having the real Internet topology might be much more interesting than a fake that could be far away from the real one. This would introduce distortion into the results.
- There are several topology generators and the all of them have plenty of proofs showing their benefits. Therefore, there is no unique solution for this item. This lack of agreement turns into divergences in the outcomes. The decision taken in a first stage concerning the features to point out and the correspondent generator to use would mask the study, as they would be carried all the way long.
- The maximum amount of nodes that can be handled is not enough, as what is being tested is the whole Internet.

c.) Implementing a full scenario

The implementation of the full characteristics in the scenario would not be an approach itself on how to build it. In fact, it refers to the reasons for which not to perform a scenario including all the features supported by the architecture. Somehow, several hints on this question have been already provided. The concept is as follows: only those elements of the protocol that were relevant in terms of the goal have to be implemented. Therefore, any other proceeding included in the protocol has to be avoided (even though it is a key point of the architecture functionality), as not the all of them have final effects on the quantization of the networks when using the proper metrics.

All these considerations drove into discarding the all of them and motivated the creation of a self-made solution able to overcome the existing hindrances by fulfilling the requirements of this thesis.

This ad hoc solution would fill in the gap between the limits of other possible ways in which to develop the project and the target of scaling. It would mainly allow the processing of any network up to the size of the Internet overriding the effects of large networks over the processing requirements.

4.3- OVERVIEW OF METHODOLOGY

Once the necessity of building an ad hoc solution instead of using an already existing approach has been discussed, this section shows which steps were taken to develop it and to solve out the problem.

First, it implied an in detail research of the Internet structure. The most recent contributions to the study of its shape and inner characteristics were analyzed. As a result, the proper way to model large networks through Autonomous Systems was found. These ASes share several global proceedings to interrelate. However, they are able to keep a certain degree of independency as the inbounds communications could differ among the all of them. On the other hand, they provide enough flexibility to the overall system as any network type can be drawn by interconnecting them in the desired way. Moreover, and as ASes mean a both already existent and operative concept, they provide us with real data. Companies can register an AS to operate in the Internet at different levels. They use the Border Gateway Protocol to interchange the information that they require (mainly for routing). Several developed algorithms can process this traffic to extract not only the existence of the ASes (or their identifiers) but the relations between any pair as well. Afterwards, this relations can be sorted out depending on what do the ASes transmit and in which way. Then, heuristics are run to classify the resulting ASes and their relations into the real Internet topology.

After obtaining this knowledge on the real Internet basis, this can go even further. Another interesting scenario can be set. From a relatively small network to any other approach which could be considered relevant due to its morphology. For instance, it would allow:

- Checking the performance of a proposal for a new Internet model.
- Guessing the future of the Internet, *i.e.*, the new shape it could have in some years time
- Testing the new Internet architecture and extract, as a result of feedback, several key points for its development:
    - values for different parameters which can now be adjusted accordingly to the model, as the improvements that are obtained can be validated
    - rearrangement of basic concepts of the architecture which now become clearly unsuitable as the network grows larger, the structure gets more hierarchical,…
    - applying new ideas to improve the performance of the system by reducing the traffic, allowing faster tracking down,…

As a conclusion, the Internet structure is not any more a fixed element inside the study as it becomes a parameter of the system. And as any other, it allows the introduction of changes to check the final performance.

In addition to this, and once there is a representation of the desired system in terms of ASes, a communication pattern has to be applied. All the hosts belonging to the Internet try to contact each other by first finding the path for the communication, then establishing it and, afterwards, interchanging data traffic. The protocols at the different layers involved in this communication process typically take care of all the necessary handshake steps. However, not any host might try to communicate with any other, and therefore to first locate it, equally. For many reasons, some of them might be accessed more often than the others. As a conclusion, a pattern should be introduced into the model. This would allow to have a realistic communication behavior in the system. Actually, many studies point to real values to model this effect.

Nevertheless, it might also be interesting to get results for the worst case by applying the most unfavorable communication pattern possible.

The next parameter to consider is the population of the system. This means setting a value for the amount of users all over the system (*i.e.*, the Internet or its replacement in terms of a new architecture proposal). Furthermore, it has to be decided how to distribute it among the ASes.

Afterwards, it has to be set how much traffic related to the architecture do they send to the network. As a rule, the architecture offers support to the traffic created by the different applications. However, this traffic does not belong to the protocol itself. Therefore, it does not have to be taken into account. The only relevant packets crossing the Internet are those that are explicitly considered in the definitions of the protocol.

Finally, the values for those parameters that are intrinsic to the next generation Internet architecture being considered for a scalability analysis have to be specified.

While the first commented parameters have almost no dependence on the architecture itself, the last ones partially do. At least, they have to be readapted in order to fit the protocol. The same would apply to the chosen metrics. Nevertheless, it can be already told in advance that they have to measure that traffic, which is an inner characteristic of the architecture. Additionally, it can also be interesting a measure on the required memory at any point to keep the state of the communication. Therefore, this provides the basic elements of any computer like device:

- processing power, through the number of lookups per second
- memory, through the number of entries that need to be stored

The programming environment under which the application was coded (as a result of several considerations; mainly the statistical/mathematical behavior of this study and the matrix facilities that it provides) is MatLab. Further details can be found in Chapter 5.

The application considers all the explained parameters. The host population is distributed among the ASes as required. Each of them sends traffic for connection through the network. This traffic will propagate across the Internet according to the traffic pattern to reach the destinations.

This propagation will follow the paths provided by the different ASes and their interconnections. As a result, the calculus provides the processing power and memory that any element of the Internet would require to support the new architecture.

4.4- RESEARCH ON THE INTERNET STRUCTURE

4.4.1- INTERNET CHARACTERISTICS

It is well known that nowadays Internet is composed of Autonomous Systems (ASes). Each of them has a range of IP's under its domain. ASes typically belong to Internet Service Providers and major companies at different scales, from local to global ones. Additionally, some large organizations can hold their own AS and provide services and connectivity to their subsidiaries. ASes are told to be autonomic, even though this is not absolutely certain. It is true that each AS can manage its own inbounds communications, routing, addressing… in different degrees, but up to a point:

- They cannot operate independently from others; hence, they are not truly autonomic, even though this could seem a paradox
- There are many global agreements and services.
- Interoperability between heterogeneous networks is not easy. The archetypical example of this situation is IPv6, which is in fact introduced as a parallel network, instead of being integrated.

Internet Assigned Numbers Authority (IANA) assigns a unique identifier to each AS. Additionally, this organization assigns blocks of IP addresses to regional Internet registries. Therefore, a second definition of AS could consider them as blocks of IPs assigned by IANA and uniquely identified through their AS number.

The AS numbers are 16 bits length. This implies 65536 possible Ids. However, not the all of them can be assigned. Some are kept as private numbers. Among several reasons, they are used for research purposes. In this way, they avoid any interference with the normal Internet operation.

ASes are not a homogeneous group. They can have several functionalities in the Internet. Their missions and connections will depend on their final aim, and vice versa. ASes can:

- Be the source and sink of data traffic. This would be the case of final organizations and Internet users. While there are other sources of traffic, mainly due to network control and monitoring functions, this would only refer to payload containing useful data. They commonly have more than one connection to the Internet through transit ASes. This allows them to ensure that in case of link failure the connectivity to the Internet is not lost. Additionally, they can route traffic to one or another ISP depending on the sort of traffic, the geographical or logical destiny it has, the available bandwidth… However, some others possess a single connection to the Internet. Actually, these ones would not require holding an AS number unless they want to keep a certain degree of independency from their ISP.
- Be transit ASes. ISP's hold this type of ASes. As explained, they do not add final traffic to the system. What they do is provide connectivity to the Internet, as they route traffic (through their inbounds network) for those hosts that belong to organizational ASes. In this way, they allow hosts belonging to different networks to communicate through a path of interconnected ISPs.

In addition to this, the existing connections between ASes can also be classified (depending on the type of AS and the routing policies that they apply) into:

- Provider to customer. This is the case of user ASes connected to transit ASes. Moreover, there can be transit ASes which provide connectivity to other transit ones as well. This is because not all ISP's work at the same scale. Some are large-scale

service providers that shape the core of the Internet structure. Final ASes containing hosts are not usually connected straight to one of them. The business for the larger ASes is connecting smaller ones. These are in charge of providing service to the final users. The resulting structure is a chain of providers sorted as a hierarchical topology.

- Peering relations. Two ASes can share a connection between them. However, none of them is a provider to the other, as they are, somehow, at the same operative level. This means specific routing policies and agreements between them.
- Sibling relations. Even though all relations that are not provider to customer could be classified as peering relations, some studies point that there are existing relations that should be sorted out as something slightly different due to their nature.

The interconnection between ASes is based upon the Border Gateway Protocol (BGP). Although BGP can be used inside ASes as well, its main use (and, actually, the one considered herein) is to exchange information for inter-AS routing. It is then namely called External BGP (EBGP).

BGP is the key protocol of the Internet in order to allow autonomy (up to a point, as already explained) by supporting fully decentralized operation of the network.

BGP is a path vector protocol and works by keeping IPs which identify networks. These IPs build paths that interconnect the whole set of existing ASes. Theoretically, each BGP table should contain the whole path to another AS, as the protocol supports full reachability. However, and due to the large amount of ASes that can be possible destinies, this would turn the tables into non-scalable. As a solution, tables are built in such a way that routes can be aggregated.

Instead of using metrics to take the routing decisions, it applies sets of policies that are configured by the system administrator. Thus, the final routing routines are not as a rule the optimal ones. This means that there is no existing algorithm adjusting the paths according to several measures on certain parameters all over the network, which would apply in an adaptive

way. In spite of this, the decisions are taken almost by hand. As a conclusion, the final routing tables imperfect. There are several reasons for this to happen, including:

- Mainly, the complexity of the Internet. While a smaller network could be managed with another protocol, the size of the Internet is so high that it cannot be properly handled. Even though it does not proof it, [SUBRAMANIAN2002] already points at the fact that the problem of running an optimal routing algorithm is NP-hard and, therefore, cannot be efficiently performed.
- A too in detail algorithm would imply too many updates. Even though these adjustments would mean improvements in the routing operation, it would also imply continuous disturbances in the system. Even more, it would become a problem for the later coming traffic. Packets of user traffic would require being resent too often as the paths change.
- The shape of the Internet is static. Its inner structure changes smoothly. Larger ISPs holding the most important ASes in the Internet stand almost immovable. Changes mainly occur to smaller ones as a result of temporarily loosing a connection, getting a new peering agreement, switching provider due to a better offer, delivering traffic through a less charged link… On one hand, it is known that changes in the policies could be observed if monitoring a reduced set of ASes. However, on the other hand, if having a broader view of the Internet there would be no appreciation at all. The statistical characteristics and behavior of the network remain still.
- There are some parameters that cannot be translated into quantifiable metrics. This is a result of having a heterogeneous network with different interests and requirements. Economical, political, social, organizational, geographical… concepts have intrinsic consequences that can hardly ever be properly expressed into metrics, as they would loose their significance.

The architectural elements in charge of exchanging information with the neighbors of the

AS (the providers and the peers, if existing) are the border routers. They can be considered as gateways, as they connect ASes to one another.

As a conclusion, BGP is an important network protocol. Although it could seem not of interest for the final user, it allows connectivity in the Internet through the ASes. Therefore, all the important information required to perform the connections of the network must be stored in BGP tables to carry out this task.

These BGP tables are the result of applying policies and rules for routing and keeping them up to date when required. Somehow, they are an output of having applied the rules. However, and as they include all that knowledge on the Internet routing performance, they can be used as an input and this valuable information can be extracted for the modeling purposes.

Actually, there are several already existing studies that have applied this commented methodology, as [SUBRAMANIAN2002] and [GAO2001].

After having performed an in detail study of the Internet, the ASes and the BGP tables, they know the relevance of the information that is hidden in there. They have developed algorithms that are able to extract this useful payload, analyze it and present it in a user friendly way. Furthermore, they have applied them successfully and the scientific community agrees with the results. Later coming studies have made comparisons to check the suitability of these algorithms and to validate their results. Even though they explain that there are other existing methods to deduce the structure of the Internet and check its statistical parameters, they all claim this BGP-tables-method as a proper one.

As a result of applying these algorithms they obtain, as a rule, a set of identifiers that correspond to the AS numbers. Additionally, they get all the existing connections between the all of them. Hence, the whole Internet topology can be stored in some files ready to be processed.

Its main usage until now had been the analysis itself of the Internet and its inner characteristics. They could check growth, variability, statistics, degree of connection… and extract some conclusions concerning the health of the system. Some organizations publish and offer resources and tools to get graphs out of them, as [CAIDA] and [NLANR].

The next step is putting order into this huge amount of data. One of the studies of the Internet topology [SUBRAMANIAN2002], after getting the AS numbers and the existing links, applies a set heuristics to the results. In this way, they manage to classify the ASes depending on whether they are the origin, the destination or a middle step regarding traffic paths, the degree of connectivity that they have, the number of customers, if they have a provider or not…

As a result, they classify all the existing ASes into 5 hierarchical levels. The fifth level, at the bottom of the hierarchy, contains the hosts. These are the only ones that act as sources and sinks of data traffic. Although there are some other sorts of traffic due to network control, operability and monitoring that might find its origins in other levels, that is the only one containing real users. These fifth level ASes are almost 14000 out of 17000. Note that these are most of the ASes (over 80%) and, thus, the amount of providers is lower than expected.

The top of the hierarchy are 22 single ASes. These are almost fully meshed, even though not absolutely (15.8 peers per AS out of a maximum of 21). However, the level ensures that any AS can be reached from any other in, at the most, two AS hops (diameter). Most likely, a single hop should be enough to reach most of the other first level ASes.

The second level does not have so many peering connections per AS (5.7). However, there is a single AS cloud that allows connectivity among the all of them in less than 10 hops. The average distance between two ASes is less than 4 hops.

In the third level of the AS hierarchy, there is in average a single neighbor per AS. The diameter is 11 and the average distance between ASes is 2. However, this is only true if not considering ASes that are absolutely disconnected. Otherwise, the distance and the diameter should be considered as infinite. In addition to this, most of ASes with peers only have a single and reciprocal one, *i.e.*, two ASes are connected between them but they are disconnected from the rest. These are considered as divisions of a single organization.

Levels 4 and 5 do not have any peering connection inside them at all.

All these features are summarized in Table 4.1. Note that these values might slightly change depending on the analyzed data sets.

	Number of ASes	Avg. Distance	Diameter	Avg. peers per AS
1 (highest)	22	1.25	2	15.8
2	215	3.90	10	5.7
3	1391	1.98(*)	11(*)	1.0
4	1421	-	-	0
5 (lowest)	13872	-	-	0

Table 4.1. Properties of the different AS levels.

The morphology of the Internet can be considered in terms of a hierarchical pyramidal structure starting with a large base and growing up into a highest reduced level. However, and as shown in Figure 4.1, the shape of the Internet topology would be rather a sphere. The hosts are located in ASes in its boundaries, in the most external layer (the fifth one). These ASes are the largest subset among all levels.

This hierarchical behavior of the Internet is also supported by [GE2001] and [MAO2004].

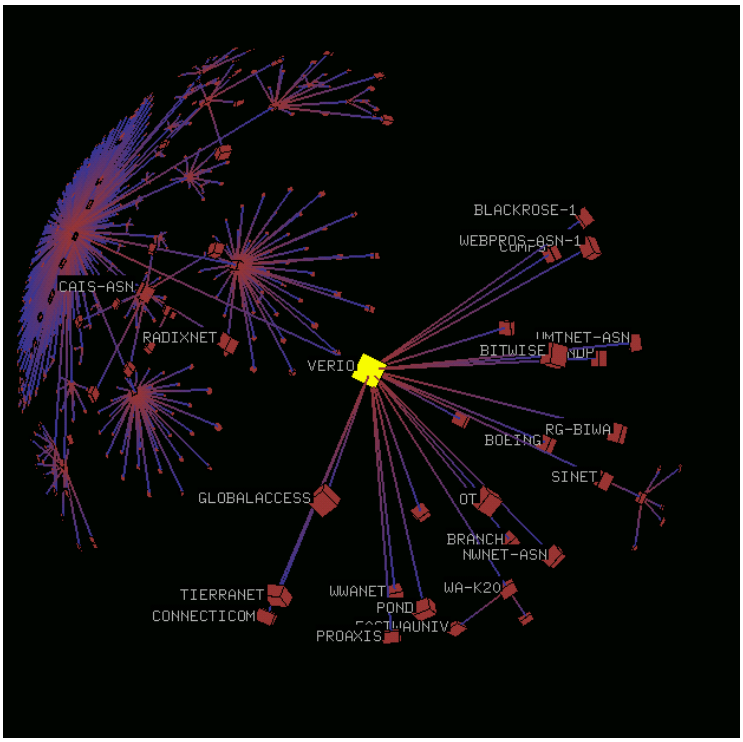

Figure 4.1. Shape of the Internet from [CAIDA].

However, these algorithms and heuristics do not go further as their aim is to provide a view of the Internet. The data sets that they obtain and the analysis that they make are a result by themselves. Another step can be taken. As a principle, this dissertation uses the Internet structure extracted from the BGP tables. Then, the research analyses are applied over the obtained topology. Thus, this proceeding manages to turn an output into a valuable input for these scalability analysis purposes.

4.4.2- COMMUNICATION PATTERN

A communication pattern must model, in terms of AS hops, the amount of requests that cross the local AS boundaries (thus, not including the internal requests for communication) to find a resolution. This distance distribution models the requests that are submitted for a new connection with hosts inside ASes at a given distance from the origin.

The model studied in this thesis is hierarchical, as it fits the description of the TurfNet architecture for a naming and addressing scheme. The corresponding distance distribution for lookup requests is in this case the same as for the later coming data flows once the communication has started (as shown in Chapter 3). Therefore, this dissertation used already existent studies on communication patterns that apply to nowadays Internet user traffic. In case that another architecture was studied, a different distance distribution might be required (depending on the proceedings for resolution of the new architecture).

This communication pattern has as its most remarkable feature a low average amount of AS hops from host to host. The main reasons are:

- The most important websites and Internet services distribute their contents for an improved usage of the Internet resources (avoid denials of service, provide fast access to the users, allow geographically based contents…)

- Corporations possess ASes at different levels for the inbounds purposes. This would be the case of a main organization that has several subsidiary locations and departments, which, even though keeping peer-to-peer relations, are ruled and managed through different policies each one.
- Due to its hierarchical shape, the higher that an AS is settled in the topology, the more other ASes that are under its domain. Once a maximum is reached, further steps do not add many more possible paths.
- Some other reasons discovered in Chapter 6 regarding the provider to customer links between ASes (an AS at a certain level might not have why to be connected to another at the immediately above one, which should be the rule).

One relevant aspect of the pattern is its definition for AS hops. As inter-AS metrics are considered for distance, the pattern must not count the initial AS as part of the communication. The resulting traffic path establishments that are inbounds regarding a single AS do not affect the global system behavior, and, thus, do not have to be considered in the model. These tables or graphs should be purged or rearranged, so that all information fits the same definitions.

[CAIDA] uses data from the RouteViews project corresponding to 1997. Even though it estimates AS path lengths using a quite different methodology from the one used by [NLANR], with data from 1997 as well, they already claim that the results are strikingly similar. These data from [NLANR] are shown in Figure 4.2. Therefore, a dynamic system based on Internet traffic flows confirms others based on relatively static routing tables information. It also illustrates the required rearrangement (*i.e.*, discounting an AS from the path).

Other possible works that have studied this statistics are [BROIDO2002] and [KRIOUKOV2004]. [YANG2003], another proposal for a new Internet architecture, also confirms the considered values.

However, as these data sources were rather old, this dissertation applied another one [UHLIG2002] for the analysis in Chapter 6.

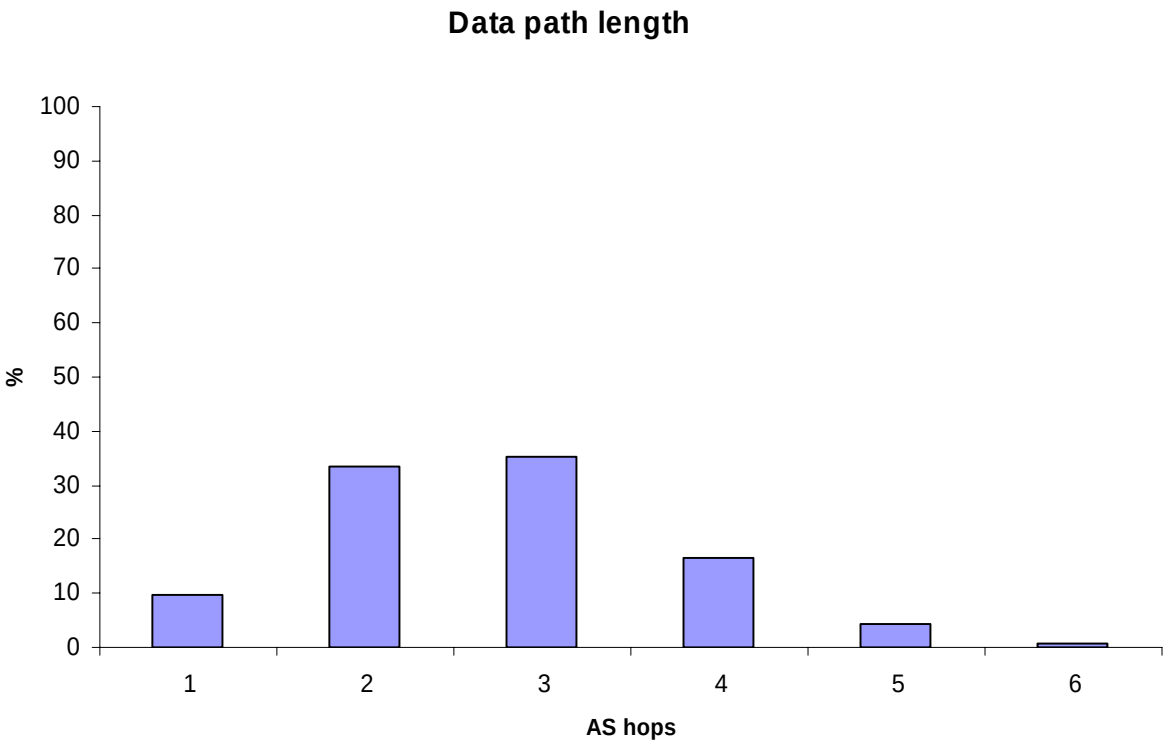

Figure 4.2. Communication distance distribution in the Internet adopted from [NLANR].

4.5- OTHER SYSTEM PARAMETERS AND METRICS

4.5.1- HOST POPULATION

This is the number of hosts in the system performing lookup requests of identities. As explained, they are located at the bottom level of the hierarchy.

In addition to this, the sort of distribution among the different ASes at level 5 is a choice of the model as well; *i.e.*, the hosts can be distributed among them in different ways (uniformly, exponentially,…) and according to different policies (randomly or depending on some parameter extracted from the Internet topology).

Furthermore, there should be a parameter modeling the percentage of time that each host is actually connected (accessible and communicating).

$\upsilon_{host} \in [0,1]$

As seen later, and to perform an analysis on the worst possible case, all hosts are considered concurrently connected (*i.e.*, $\forall host, \upsilon_{host} = 1$). This means, both registered and submitting requests for new resolutions all the time. Hence, the results provide the maximum memory space and processing power in an edge scenario.

Therefore, the final chosen number for the total population would support an even much larger amount of real end terminals, mainly due to two key aspects:

- Hosts can decide up to which point they want to be accessible through the whole topology. This can happen, for instance, in order to keep privacy. Therefore, they can choose to be only locally reachable. As this would mean not requiring a global address, these hosts are not counted.
- Hosts are connected intermittently. Internet architectures as TurfNet already take this into consideration in their proceedings. TurfControls (*i.e.*, network operators) and/or hosts themselves set the parameter that applies to how often they have to renew their online status for the connections to be able to be established. In all crossed ASes through the possible communication paths a life time is set. Unless periodically updated, the assigned state would become free to be used by other hosts.

Hence, the chosen population would translate into a several times larger real equivalent one as υ_{user} tends to realistic values.

4.5.2- COMMUNICATION RATE

These are the connection requests per user and second. When trying to communicate with another host for which there is no known locator, a host typically tries first to resolve its identity. Once resolved, the data flow between both hosts can start.

This communication rate parameter only takes into account brand new connections. This means that:

- Once the communication has started, the host does not have to perform any other lookup.
- Furthermore, other hosts in the same AS would skip having to resolve the identity and their communication would start immediately.
- Hosts which requests traverse ASes in which state was already created might beneficiate from it.
- The state would be typically kept during a period defined by the architectural policies. This would replace any cache system. Moreover, the state of websites and services considered static could be almost permanently stored to save requests.

Here again, as in the previous item, the concept of υ_{user} should apply. Users are neither 100% of the time connected nor trying to communicate. This must be taken in conjunction with the previously described characteristics of this parameter. Hence, the communication rate herein used would translate into a *de facto* much larger one.

Concerning the heterogeneity of the users, Figure 4.3 shows an equivalent model of the hosts and their communication rate.

Inside an AS, several types of hosts coexist (mobile devices, PCs,…) and in different number. Each of these subsets has its own particularities regarding traffic statistics. Nevertheless:

- Due to the in-built privacy of ASes, which increases when applying a next generation Internet architecture on top of them, the amount of users and their individual traffic rates cannot be externally distinguished.
- Actually, the only traffic amount that is relevant is the outgoing one, $\lambda_{Forwarded}$, as it will have to be handled by the inter-AS gateways and the provider ASes.

Therefore, the final parameter considers plain users and common values for the lookup rates, λ_{user}.

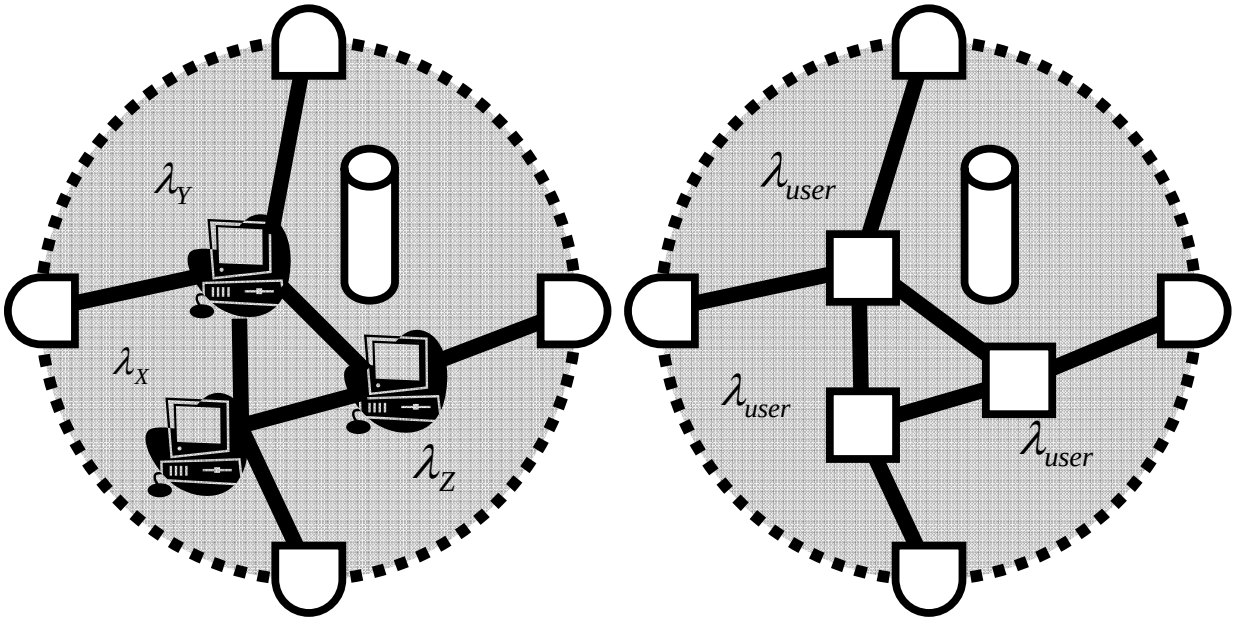

Figure 4.3. Equivalent model of the hosts and their communication rate.

4.5.3 METRICS

The information that this work collected was measured in terms of the following two metrics. Even though others might have been used, as overheads, the used ones fulfilled the scope of this study as they included all the relevant possible outputs. Furthermore, both of them have practical significance as hardware features by applying a conversion factor, which is a parameter of the current technological state of the art.

- Lookup requests per second. It allows the validation of the frequency of address lookups for remote globally reachable hosts prior to communication. The model measures the requests being attended, resolved and forwarded by each AS, as well as the averages and the aggregate values per level. Each request will require some CPU time to be processed; *i.e.*, to check out the existence of the host in the local tables and to resolve the identity if possible. Hence, each machine must be able to accomplish with all the involved operations by having a capacity μ larger than the minimum.

$$\mu \geq \lambda \ , \ \lambda\left[\frac{req}{\text{sec}} \cdot \frac{operations}{req} \cdot \frac{cycle}{operation}\right] \Rightarrow \lambda\left[\frac{1}{\text{sec}}\right] \Rightarrow \ frequency$$

The conversion factors depend on the efficiency of the architectural proceedings and the hardware. For instance, a dedicated on purpose device would achieve much higher rates than a commercial one using a software solution.

- Number of entries to be stored for registered hosts. They imply the memory requirements in each AS. It depends on the size of the identifiers defined by the architecture, the additional information per entry to be added concerning the routing, and the sort of database storage to be used. Thus, they could be measured as *bytes*. As most new Internet architectures are based upon the concept of soft state, and to provide faster access, volatile memory is recommended.

4.6- APPLYING TO THE TURFNET NEW ARCHITECTURE

It has been already explained that the Internet can be expressed as a graph, which is indeed a vast collection of ASes and their interconnections. Previously, this dissertation discussed how the actual Internet architecture lacks in autonomy and it cannot support the current and future requirements of the network and its users, as communication is turning into both autonomous and mobile. The main reason for this is that the ASes cannot achieve the desired performance due to their implicit definition.

On the other hand, it is feasible to claim that both given the definition of what a Turf is and the way it runs its protocol for internal and external hierarchical naming and addressing proceedings, the TurfNet architecture fits the AS Internet graph and is then able to overcome the restrictions of the ASes. Indeed, many similarities between ASes and Turfs can be pointed out to assure this:

- Both are administrative independent subsets of a global architecture.
- Nowadays inter-AS communications are held by gateways.
- Turfs have an internal address space and ASes can also rule their own (even though under several restrictions). Turfs can go even further and have their own protocol for internal communications, which can differ from those from their neighbors. In this case, the Turf gateways take care of performing the proper protocol translation apart from the inherent address translation.
- The provider to customer and the peer-to-peer relations match the definitions of vertical and horizontal composition respectively.

As observed, this classification fits the concepts of TurfNet from Chapter 3. TurfNet required having a hierarchical structure with, at least, vertical composition among its Turfs. Furthermore, the current Internet has peer-to-peer relations that can be supported by the horizontal composition of the TurfNet design.

As a conclusion, this study models the whole Internet as a set of interconnected Turfs or, in general terms, domains running a roughly similar hierarchical scheme as the one from TurfNet.

4.6.1 MAPPING TURFNET AND THE INTERNET

Due to some constraints of the Internet that have been already commented, the setting of the TurfNet architecture into the Internet would require an additional feature, which was not considered in the descriptions of TurfNet.

In the Internet, there is no single top provider, but there are 22 of them. Therefore, any hierarchical architecture would require some additional mechanisms. TurfNet can use the horizontal composition. In addition to this, a conceptual upper virtual level can be considered as running on top of the 22 major network providers. This 0-level could be a common repository or database that is shared, implemented, run,… by them as a collaboration. Each one would be a

customer for this final provider that would be able to resolve all the remaining lookup requests for resolution of identities.

As a second consideration, the sort of horizontal composition that has been implicitly used implies that the type of information that is shared between ASes is that of the registry tables. Chapter 3 and, mainly, [SCHMID2005] suggest alternatives for more flexible scenarios.

Finally, and as a third consideration, the approach that has been adopted in this thesis regarding the registrations is that the situation is rather static. This means that the registration traffic is not considered, as it is a small fluctuating factor on top of the large amount of lookup requests. In case that it was considered, it would be counted as an extra factor to be added to the requests for resolution of identities (as the path and the forwarding proceedings are both hierarchical and roughly similar). More details on this can be found later on in Chapter 7.

4.7- MATHEMATICAL DESCRIPTION

The main contribution of this thesis is an investigation into whether novel naming and addressing architectures for large-scale internetwork hierarchies are feasible, *i.e.*, whether they can scale to networks that are at least the size of the current Internet.

This subsection describes the definitions, operations and metrics of the analytical model used for this analysis.

It follows the same pattern as the one applied in the extended explanations on the proceedings that have been provided in the previous sections. Therefore, and first of all, it presents these definitions, operations and metrics in a generic shape. This would allow, as already explained, to run a parallel study of a different new Internet architecture proposal by just instantiating them in some other way that would apply to these other architectural schemes.

Later on, it places a redefinition of some of them to fit the hierarchically based naming and addressing architecture that we are studying (*i.e.*, to our application example, that is the TurfNet architecture).

4.7.1- GENERAL DESCRIPTION

This item is divided into two parts. While the first part sets the definitions for the model, the second one places the operations that apply to it.

Definitions

A network domain

D

consists of a set of nodes with unique identifiers

$id_D(n)$

taken out of its local namespace

N_D: $D = \{n_1, n_2, ... \mid id_D(n) \in N_D\}$

The population of a network domain D is

$pop(D) = |D|$

Nodes can be part of multiple network domains at the same time (supporting multihoming and multisourcing) and have a local identifier in each one,

$id_A(n) \in N_A$ and $id_B(n) \in N_B$

Note that different network domains can use overlapping of local namespaces; global uniqueness is only required for identities.

An internetwork is a topology graph

$I = \langle V_I, E_I \rangle$

with network domains as vertices, *i.e.*,

$V_I = \{D_1, D_2, ...\}$

and links between network domains as arcs. The population of I is

$$pop(I) = \sum_{\forall D \in V_I} |D|$$

The distance

$d_I(D_A, D_B)$ between network domains D_A and D_B in I is the length of the shortest path between the two of them. The diameter of the internetwork is

$$dia_I = \max_{\forall A,B \in V_I} d_I(D_A, D_B)$$

Each network domain D in I has a level

$$l_I(D) \in \Im$$

where 1 is the highest level of the hierarchy, and

$$h_I = \max_{\forall D \in V_I} l_I(D)$$

is its overall height. The instantiation offers more details on the assignment of these levels into a hierarchy.

A simple, statistical function models the global traffic in an internetwork. Communication between all pairs of network domains that are at a distance d_I apart causes a fraction $p(d_I)$ of the overall traffic in the network, *i.e.*,

$$\sum_{1 \le d_I \le dia_I} p(d_I) = 1$$

The function $F(d_I)$ is built out of its Cumulative Distribution Function

$$CDF(d_I) = \sum_{i \le d_I} p(i)$$

$$F(d_I) = P(X > d_I) = 1 - CDF(d_I) = \sum_{i > d_I} p(i)$$

For the purposes of this analysis, all nodes communicate at a uniform rate

$$\lambda_{user}$$

and they are equally likely to communicate with any other node of equal distance, but not with those that are located at different distances, according to the traffic pattern itself. Consequently, the aggregate communication frequency of a domain *D* is

$$\lambda_I(D) = \lambda_{user} \cdot pop_I(D) = \sum_{\forall i \in D} \lambda_i = \lambda_{Forwarded}$$

and the overall communication frequency in *I* is

$$\Lambda = \sum_{\forall D \in V_I} \lambda_I(D)$$

Note again that communication between the nodes within the same domain does not factor in.

Operations

Without a fixed addressing scheme like the one in the current Internet, which implicitly defines a communication path between two entities that follows the aggregation structure, nodes must explicitly discover communication paths. For this purpose, nodes announce their presence through registration to allow others to contact them. To resolve the location of an identity for the purpose of initiating communication, nodes perform lookup operations.

The individual network domains maintain node registrations. They manage at least the registrations of those of their local nodes that choose to be reachable outside the domain and the more transient registrations of other nodes that were registered in there through the inter-domain relations.

$$S_I(D)$$

On the other hand, the domains in the internetwork are able to resolve a fraction of the incoming requests. The others will be forwarded

$$\lambda_{Incoming}(D) = \lambda_{Forwarded}(D) + \lambda_{Resolved}(D)$$

The resolutions depend on $F(d_I)$ and a factor η, which models the additional effect of the inter-domain relations.

$$\lambda_{Forwarded}(D) = f\left(\lambda_{Incoming}(D), F(d_I), \eta(D)\right)$$

$$\lambda_{Resolved}(D) = g\left(\lambda_{Incoming}(D), F(d_I), \eta(D)\right) = \lambda_{Incoming}(D) - f\left(\lambda_{Incoming}(D), F(d_I), \eta(D)\right)$$

4.7.2- INSTANTIATION

Definitions

Each domain is an AS or Turf

$$D \rightarrow AS$$

Therefore, the Internet can be expressed as the following internetwork

$$I = \langle AS_I, R_I \rangle$$

where the relations (peer-to-peer or provider to customer) are the arcs

The Internet has 5 levels. Each AS belongs to one of them

$$h_I = 5$$

The population of each AS is

$$if\ l_I(AS) = 5 \rightarrow pop(AS) = |AS| \geq 0$$
$$else\ pop(AS) = 0$$

The distance $d_I(AS_A, AS_B)$ between ASes are AS hops, or simply hops

$d_I = hops$

Operations

The size of the tables that an AS in *I* needs to maintain is

$$S_I(AS) = \alpha(AS) + \beta(AS)$$

where α are the registrations through vertical composition. It depends on the position of the domain in the overall topology

The second factor, β, are the registrations through horizontal composition and depends on the used *scope*. *H* is the subset of *I* that has only the peering connections for horizontal composition

$$H = \langle AS_H, R_H \rangle \subset I$$

Given an AS, β will include the registrations belonging to all ASes such that

$$d_I(AS_A, AS_x) \leq scope$$

On the other hand, the resolved and forwarded requests are

$$\lambda_{Forwarded}(AS) = \lambda_{Incoming}(AS) \cdot F(hops) \cdot \eta(AS)$$
$$\lambda_{Resolved}(AS) = \lambda_{Incoming}(AS) \cdot (1 - F(hops) \cdot \eta(AS))$$

Nevertheless, the total amount of lookup requests leaving an AS might be larger than $\lambda_{Forwarded}(AS)$ as ASes could have more than a single provider. Therefore, the requests get multiplied as they are forwarded

$$Total\ \ forwarded(AS) = k \cdot \lambda_{Forwarded}(AS)$$
$$1 \le k \le N$$

k depends on the set of policies that have been locally or globally defined to handle these events for each AS. To avoid system overload, it should tend to 1.

η is calculated as the gain by means of horizontal composition, *i.e.*, number of entries that are acquired from the registers of other ASes (belonging to the scope radius) compared to all the entries that the AS does not have (*i.e.*, those that are not under its hierarchical subtree). Note that $\eta = 1$ if no peering scope is used.

$$\eta(AS) = (pop(I) - \alpha(AS) - \beta(AS))/(pop(I) - \alpha(AS))$$

Even though the same rules could be applied to level 1, it is more proper using a deterministic approach, as there are only 22 ASes and they shape the top of the hierarchy. These ASes are able to resolve a proportional part of their input requests corresponding to the amount of entries in their registers compared to the total amount of possible identity destinations.

$$\lambda_{Forwarded}(AS) = \lambda_{Incoming}(AS) \cdot \zeta(AS);\ \lambda_{Resolved}(AS) = \lambda_{Incoming}(AS) \cdot (1 - \zeta(AS))$$
$$\zeta(x) = (pop(I) - S_I(AS))/\ pop(I)$$

In case that the architecture requires a unique top provider in a namely called level 0, it would be able to resolve all the remaining lookup requests that the level 1 ASes were unable to.

$$\lambda_{Resolved}(0) = \lambda_{Incoming}(0) = \sum_{\forall i: l_I(i)=1} \lambda_{Forwarded}(i)$$

5. IMPLEMENTATION

5.1- INTRODUCTION

This chapter offers the basic guidelines of the implementation for the model. First, it shows the facilities of MatLab and its adequacy to the problem requirements, as a means of justification for its use. Then, it makes a pseudo-algorithm proposal describing the functionalities of the mathematical model; *i.e.*, this description fits the definitions and operations for the required measurements that were discussed in Chapter 4. A working version of this pseudo-algorithm in terms of MatLab code is shown and commented in the Appendix.

5.2- ENVIRONMENT

This section provides some hints on why to code the application under the MatLab environment and the advantages that it provided for this dissertation.

The application was coded under MatLab7 R14, which is the main commercial product of The MathWorks Inc. [MATHWORKS]. MatLab is a high-level technical language and interactive environment for algorithm development, data visualization, data analysis and numeric computation. It allows solving technical computing problems faster than with traditional programming languages as C or C++.

MatLab has a wide range of applications including signal and image processing, communications, test and measurement… In addition to this, the MatLab generated code can be easily integrated with other functions and applications coded with different programming languages.

The key features of MatLab include:

- High-level language for technical computing.
- Development environment for managing code, files, and data.

- Interactive tools for iterative exploration, design, and problem solving.
- Mathematical functions for linear algebra, statistics, Fourier analysis, filtering, optimization, and numerical integration.
- 2-D and 3-D graphics functions for visualizing data.
- Tools for building custom graphical user interfaces.
- Functions for integrating MATLAB based algorithms with external applications and languages, such as C, C++, Fortran, Java, COM, and Microsoft Excel.
- Avoids the use of pointers in functions. All parameters are passed as references as a default. If they are modified in the function, they get a duplicate.

The reference documentation of MatLab can be found at their official site [MathWorks] for further reference.

5.2.1- SPECIAL FUNCTIONS AND UTILITIES

This introduces the main functions and tools that were used when coding the application. In addition to this, it provides a brief on their syntaxes and improvements that derive from them.

Matrix functions

size(X,dim) returns the size of the dimension of X matrix specified by dim.

find(X,k) returns at most the first k indices corresponding to the nonzero entries of X. k must be a positive integer, but it can be of any numeric data type

X can be defined from another matrix Y. This is interesting when locating the k elements belonging to Y that have a certain property.

Note that those elements from X that are zero do not count.

The use of the *find* function was a key point when coding the application in order to save running time. This function has been seriously optimized in MatLab and is far more efficient than any other searching function or comparing strategy under any other programming language like C.

Random functions

MatLab provides several in-built functions to generate random numbers. In addition to this, we can obtain sequences that follow a certain probability distribution.

unidrnd(N) generates discrete uniform random numbers with maximum N. The parameters in N must be positive integers. N can be a vector, a matrix, or a multidimensional array.

This function allows randomizing the election of a provider among several, in case that this was required by the forwarding policies. As it is in-built, the time consumption that it had, turned to be negligible. Therefore, the abusive use that the application made out of it in the code did not affect the performance of the operations.

Input/output functions

While in other programming languages the data has to be stored almost character by character, MatLab allows saving and loading data in the same matrix like format that you can use in the environment, for an ease of access.

dlmwrite('filename', M, 'D') writes matrix M into an ASCII format file, using the delimiter D to separate matrix elements. The data is written starting at the first column of the first row in the destination file, filename. The resulting file is readable by spreadsheet programs.

load('filename') loads all the variables from filename. If filename has no extension, load treats the file as ASCII data.

Sparse matrices

The data sets that were used to run the model were huge. This makes sense, as they included all the existing ASes in the Internet and their interconnections.

Just creating the matrices that would have been required already meant an amount of RAM memory that made the system both unfeasible and unstable. This fact was validated and the solution was discarded.

After studying the shape of the matrices that were built (see Chapter 6), it was noticeable that most of the space for each of them was empty. As already explained in Chapter 4, an AS number is provided to those domains being accessible in the Internet. This means from 1 to 65535. However, some of them are reserved as public and others are not used yet.

Additionally, and as the matrices are not full, it was not only a waste of memory but of execution time as well, due to the intensive use that was made out of the function *find*.

MatLab can store this data in memory as sparse matrices, *i.e.*, matrices containg a high number of zero elements. This allows:

- Storage of only the existent elements and their indices.
- Reduction of computation time by applying operations only to non-zero elements.

This is made by using a three matrices strategy to keep the proper indices and data. Nevertheless, even though what is really there in memory are these three matrices, MatLab allows the use of the normal matrix functions on them in a transparent way. Therefore, the system can virtually keep on working with the original matrix.

sparse(A) creates a sparse matrix or converts an already existent full one into sparse

Profiler

The Profiler is a MatLab tool that helps improving the performance of the M-files. The Profiler produces a report for these MatLab files showing where the time is spent; code line by code line both in absolute times and in percentages. Additionally, it points out where the time is wasted (for instance, when presenting useless values in screen) and suggests the possible actions to be taken in order to reduce the computing time.

MatLab Component Runtime (MCR)

The MatLab Component Runtime (MCR) is a stand-alone set of shared libraries that enables the execution of M-files. It allows deploying components in a machine in which MatLab has not been installed.

However, it has several limitations, mainly including no support for applications with graphical interface. Hence, all applications have to run through the command line.

mcc

mcc is the MATLAB command that invokes the MatLab compiler. It generates the C wrapper files and is able to build the final stand-alone binary files from the initial M-files. Additionally, other C or object files can be added.

mcc –W main –T link:exe <MAIN.m> <funct1.m> … <functX.m>

This invocation gets the .m files as input, produces a POSIX shell main() function, generates a C/C++ code function that operates as a container for the MatLab code, and compiles

it to get the object files. Finally, it links them into a stand-alone application with the same name as the MAIN function and .exe extension (or more generically a binary file, as we could be using a Linux operating system).

The files that are obtained include a *ctf* (Component Technology File). This sort of file contains executable content that is automatically extracted during the first execution.

5.2.2- OTHER MATLAB FEATURES

In addition to those, other MatLab features were used. However, these are quite generic functions that could be found in other packages and were used not as part of the main code but to rearrange the final data outputs. It includes graphical and statistical support.

bar, xlabel, ylabel, title, set, saveas… draw bar graphs and arrange their properties.

max, min, median, std… obtain statistics of a given collection of numerical inputs.

5.3- SYSTEM SETTINGS

The MatLab files were coded and compiled in a laptop. However, and mainly due to the high amount of operations involved in the calculus, there had to be a scheme to have them full time running. The reason for this is, obviously, the size of the data sets that were managed. These sets contain the information regarding all the AS identities shaping the Internet and their interconnections.

Actually, it has already been suggested (even though not proved yet) by [SUBRAMANIAN2002] that managing information from BGP tables to obtain a topology graph as defined in Chapter 4 (*i.e.*, with network domains as vertices and links between them as arcs) is an NP-hard (Non-deterministic Polynomial time) problem. More technically, NP-complete problems

are defined as those that cannot be solved by a nondeterministic Turing machine in polynomial time. When a decision version of a combinatorial optimization problem is proved to belong to the class of NP-complete problems, then the optimization version is NP-hard.

Although the BGP tables themselves to be analyzed are not handled, the calculus operate over the whole set of inter-AS links. This suggests that any strategy for 100% optimal end-to-end routing is indeed infeasible, and reinforces the usage of a statistical approach for the system.

Hence, the conclusion is that, as explained in Chapter 4, the number of operations and parameters in the system had to be kept as under control as possible.

Nevertheless, despite accurately controlling the operations by usage of the MatLab Profiler utility (see Section 5.2), the running time of most of the experiences turned into quite high.

As a conclusion, it was decided to deploy the components to other machines, which could be running in a Laboratory without any scheduling restriction. The simulations were held in there with 2 PCs 2’40GHz with 256Mbytes of RAM that could be running 24-7 without any sort of restriction.

This required the use of the MatLab Component Runtime (Section 5.2) for the same version of MatLab as the one used to compile the code. Once it is installed in the new machines, they allow running the code without the necessity of having MatLab fully installed in there; even though we already explained that it has several limitations. These include being limited to command line applications, as the MCR does not have support for graphical interfaces (that, otherwise, would increase the computation time).

The proceeding is as shown in the following steps.

1. Create the MatLab scripts or modify an already existent one by adding some new features
2. Debugging and profiling processes in MatLab (optional)
3. Moving the files to MatLab files .m
4. Compiling and packaging from the command line
5. Transferring the stand-alone executable and required files to the remote machine

6. Controlling through remote desktop application
7. Collecting the results and performing analysis to obtain graphs and statistics

5.4- IMPLEMENTATION

This section shows the mechanisms that rule the application modeling the system. It first provides a global view on how the algorithm works by means of a high-level description of the different code blocks.

Afterwards, a pseudo algorithm models the overall system including the particularities of TurfNet.

5.4.1- OPERATIVE MODEL

Provided the definitions of the different elements of the model and the mathematical theoretical operations that apply to these elements, from Chapter 3 and Chapter 4, these are the basics of the algorithm.

The application can be divided into 4 blocks:

1. Initialization
2. Registration of nodes
3. Lookup and resolution
4. Collecting data, performing statistics and drawing graphs

Initialization

The first task that is held in this part of the code is that of loading the files that shape the AS hierarchy and its inner connections and setting all this information as a graph with vertices and arcs. In addition to this, the code requests the amount of hosts in the system (as defined in

Chapter 4) and the frequency of requests for resolution that each one is submitting to the Turf hierarchy.

The second objective is preprocessing the information. It has to be remarked that this is the key point of the whole algorithm. The main issue when handling all the information loaded into memory is that the set of ASes that shape the Internet is huge. Hence, this could render the study into infeasible, as there has to be a high number of searches.

As explained in Chapter 6, the data sets are divided into two files. The first one contains the hierarchical level of each AS. The second one contains the links between ASes. Thus, a lookup would mean identifying in the first list an AS to locate its level. Afterwards, it would require checking the second list for all those ASes that share a hierarchical connection with it. Then, the level for each of them would have to be checked prior to applying the defined policies to decide to which one to submit the requests. Additionally, a similar approach would apply to all the peer connections of the initial AS. This proceeding is clearly inefficient.

As a conclusion, if the data was treated without being previously preformatted, it would mean a waste of resources due to the amount of searches to be performed. Therefore, the data was placed into memory in a smart way by running a previous preprocessing stage that is handled before any other operation. This data placement follows a matrix structure that is defined as follows:

Given $I = \langle AS_I, R_I \rangle$ and according to the definition of level $l_I(AS)$ in Chapter 4:

$$if\ PROVIDER(x,x) = k, k \neq 0 \Leftrightarrow l_I(x) = k$$

$$if\ PROVIDER(x,y) = -1 \Leftrightarrow Peer_{xy}$$

$$if\ PROVIDER(x,y) = k, k \neq 0, k \neq -1 \Leftrightarrow Provider_{xy} \wedge l_I(y) = k$$

where *Provider* and *Peer* denote provider to customer and peer-to-peer relations respectively between ASes *x* and *y*.

In any other case, either the AS or the relation do not exist. It is represented by a blank in the sparse matrix.

The following properties can be extracted from the matrix characteristics:

- If an AS number is used by a certain AS (*i.e.*, the AS exists), the corresponding element in the diagonal of the matrix is different from zero.
- Given an AS *x*, its meaningful vector can be extracted by multiplying all the elements in the matrix by a second vector that has all of its elements equal to zero except the one corresponding to *x*, which is equal to 1.

$$if\left(i = x\right) \rightarrow h^{x}\left(i\right) = 1$$
$$else \rightarrow h^{x}\left(i\right) = 0$$

$$v^{x}_{1\times MAX} = h^{x}_{1\times MAX} \cdot PROVIDER_{MAX\times MAX}$$

- All the elements of a representative vector that are equal to -1 show a peer connection between *x* and *y*.

$$v^{x}\left(y\right) = -1 \Leftrightarrow Peer_{xy}$$

- All the elements of a representative vector that are equal to N show that *y* is a level N provider of *x*.

$$v^{x}\left(y\right) = N \Leftrightarrow Provider_{xy} \wedge l_{I}\left(y\right) = N$$

Actually, the utilization of a matrix evidently fits the behavior of MatLab, which has in its basics being able to deal with matrices in a highly efficient way. Furthermore, as there are elements that are not defined (in fact, the most of them), it allows the usage of sparse matrices, as previously explained in this chapter.

The shape and density of the resulting matrix are shown in Chapter 6.

In this part of the code we can also get some statistics from the original data sets regarding the number of ASes per level, the number and sort of links, the size of the clouds, the per level diameter, the average number of peer hops… that are shown in Chapter 4. Most of them were removed when the complexity of the system started to grow up, as they were not contributing with new information.

Registration of nodes

```
REGISTRATION PROCESS{
        ∀i : l_I(i) = 5 { #from all level 5 ASes
                Register(i);
                Peering_registration(i, original_scope);
                Hierarchical_registration(i);
        }
}

Hierarchical registration(i){
        ∀j : v^i(j) > 0 { #to all providers
                Register(j);
                Peering_registration(j);
                Hierarchical_registration(j);
        }
}

Peering_registration(i, scope){
        scope - -;
        if scope ≥ 1
                ∀j : v^i(j) = −1 { #to all peers
                        Register(j);
                        Peering_registration (j, scope)
                }
}

Register(i){
        #add entries for ASi users in local tables
}
```

Figure 5.1. Registration process.

Lookup and resolution process

```
LOOKUP PROCESS{
        ∀i : l_I(i) = 5 { #from all level 5 ASes
                Select_providers_to_send( v^i_1xMAX );
                for j=1 to N_Selected _Providers
                        Resolution_request(lookup_requests, 0);
}

Resolution_request(incoming, hops){
        if collision_detection=true
                if Check_if_already_arrived()=true
                        return();

        #resolve and forward the non resolvable
        forwarded = (pop(I) − (α(j) + β(j))) / (pop(I) − α(j)) × incoming × F(hops)
        hops + +; #count the hops for resolution
        Resolution_request(forwarded, hops);
}

Select_providers_to_send( v_1xMAX ){
        #according to defined policies; the all of them as a default
}

Check_if_already_arrived{
        #perform fast check of recent queries
}
```

Figure 5.2. Lookup and resolution processes.

Collecting data, performing statistics and drawing graphs

The aim of this code block was an automatic means of support in order to collect and analyze the obtained data. Nevertheless, and as it does not imply any special feature, no further explanation on it is provided.

5.4.2- ALGORITHM

For more details, a version of the code can be found in the Appendix.

6. APPLICATION AND RESULTS

6.1- INTRODUCTION

This chapter contains the results for different cases and scenarios of application of a hierarchical naming and addressing architecture for a next generation Internet proposal. Thus, it refers to the instantiation in Chapter 4 of a generic case and its implementation in Chapter 5. First of all, it shows the values that were used for the different parameters in the system to fit the previously described model. Somehow, this would allow a third party to reproduce these experiments under the same conditions.

Afterwards, the chapter introduces the obtained results and discusses them. As a conclusion, it points the necessity of refining the architecture to increase the efficiency. Therefore, the most valuable tests are repeated to validate the achieved improvements by using different strategies.

In addition to this and apart from other complementary results, the chapter shows the latency for the beginning of the communication data flow in AS hops. This allows a conceptual comparison in terms of efficiency between this hierarchical data flow like scheme for lookup and resolution, and a more conventional one like DNS.

6.2- VALUES FOR THE PARAMETERS

6.2.1- INPUT FILES

The original input files used for this dissertation belong to the Computer Science Division, University of California, Berkeley. The date is 10th of February 2004. They were obtained according to the algorithms and heuristics shown in [SUBRAMANIAN2002]. The original BGP tables came from the RouteViews project in the University of Oregon.

There are two main files. The first one contains a list of ASes and their interconnections. It also specifies whether the link is considered a provider to customer relation or a peer-to-peer one. The second file contains the hierarchical level of each of the ASes in the Internet topology.

They allow the extraction of the topology graph $I = \langle AS_I, R_I \rangle$.

6.2.2- POPULATION AND DISTRIBUTION

According to the definition of host in Chapter 4, there must be a number of concurrent users in the fifth level of the hierarchy. Those are the ones that try to establish communications with each other. This dissertation considers $pop(I) = 1$ billion hosts in the system at the bottom of the hierarchy.

This value was estimated from the Internet Domain Survey of [ISC] and the World Fact Book of [CIA]. Actually, the estimation is much above the real values, mainly due to the concurrent behavior. This was made on purpose in order to provide a wide enough margin for the values concerning a raise in them in the future.

Regarding its distribution, two functions were applied. The first one provides an evenly distribution. This is the fairest way in which to set the hosts in the different ASes at level 5. Any other method pretending to resemble the real situation would require having many more details of each of those ASes (corporative identity, business sort…) and a by hand distribution.

Nevertheless, this chapter also checks the effect of an exponentially decreasing distribution that has the same amount of final users.

6.2.3- TRAFFIC AMOUNT

Each of the hosts tries to establish a communication with a new destiny host not belonging to the same AS every 100 seconds (*i.e.*, $\lambda_{user} = 0.01 [req/sec]$). In the same way as with the population, this value was estimated further above any reasonable expectancy.

For more details on its significance and repercussions, a broader explanation is provided in Chapter 4.

6.2.4- TRAFFIC PATTERN

The traffic pattern $p(hops)$ mainly proofs that, due to several reasons already pointed in Chapter 4, hosts are more likely to communicate with other hosts that are not too far away from their location, usually about 3 AS hops.

As already mentioned, there are multiple possible sources of traffic patterns. However, the all of them are quite similar and share the same mean value. The finally used one, [UHLIG2002], was chosen as a result of the following considerations:

- It was the one that presented the most recent data.
- It had not only a graph but concise values for each number of AS hops. This avoided having to guess them.

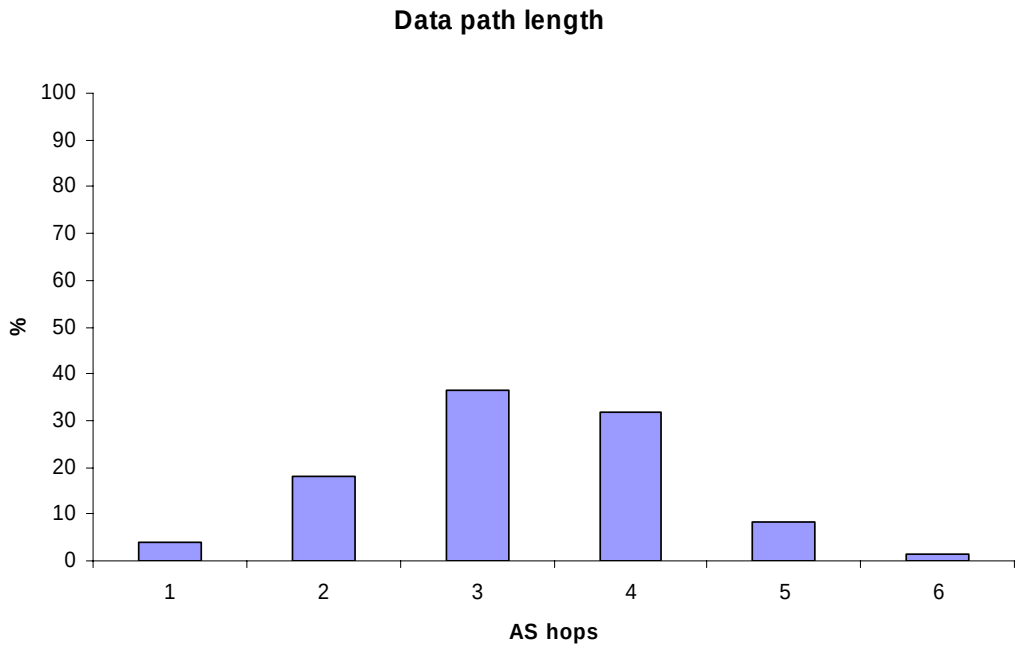

Figure 6.1. Communication distance distribution in the Internet adopted from [UHLIG2002].

AS Path Length	1	2	3	4	5	6+
Percentage of Communication Instances [%]	3.9	18.1	36.4	31.8	8.3	1.5

Table 6.1. Communication distance distribution in the Internet adopted from [UHLIG2002].

If instead of using a pattern the study applied absolutely none, it would mean relying just on the hierarchy itself to provide resolutions (*i.e.*, each AS would be able to resolve a part of their input requests proportional to the amount of population they know from the whole amount of hosts in the Internet). This would mean that each host in the Internet is likely to communicate with any other host at any distance, while it has been shown not to be true.

6.3- LINEARITY OF THE RESULTS

This set of tests checked the behavior of the system related to the input parameters: the population in the system and the traffic frequency in lookup requests per second that they are issuing to the Internet.

The aim of it was a quantization on the effect of these parameters on the final obtained results. Actually, they are the only assumptions in the system. All the other parameters have a solid background.

It was confirmed that the system behaves in a linear way both in reference to the size of the tables and the number of lookup requests per second in the system.

As a conclusion, it proofs that the choice of values for them is not a critical point in this work. Thus, the results for any new pair of resolution requests per second and number of users can be automatically derived by applying a mere conversion factor.

6.4- EFFECT OF THE POPULATION DISTRIBUTION

This part of the chapter pretended to approach quantification for the effect of a not evenly distributed population. This would model a both more accurate and heterogeneous Internet where the different network domains have different sizes in terms of amount of users.

$$f[n] = f_O \cdot e^{-\frac{n}{\tau}} = 10^6 \cdot e^{-\frac{n}{999.5}}$$

As a simplification, it assumed having all the ASes in level 5 straightly connected to a single Internet-like level. The same 1 billion of hosts were distributed according to an exponential distribution with 1 million as its starting top value. Furthermore, it assigned a single differentiate rate for communication to the users of each AS (*i.e.*, it considered that the amount of lookup requests being forwarded would increase or decrease as the number of inbounds hosts decreases or increases respectively).

$$\lambda'_{user} = \lambda_{user} \cdot \left(1 + \frac{\Delta population}{Population}\right) = \lambda_{user} \cdot \left(1 + \frac{\Delta population}{1\,billion}\right)$$

The assumption was that the more that the population is concentrated in some ASes, the less amount of lookup requests that are issued to the core network for a resolution.

Surprisingly, this new distribution had almost no effect. The reduction was only 0.04%. To achieve a 5.25%, the starting value for the exponential should be one tenth of the whole population, which makes absolutely no sense. The reason for this is that, even though there is a high contrast between the most populated AS and the less populated ones, the population is spread into too many ASes (around 14000) for any effect to be noticeable. This spreading has

much more relevance than the way in which the population is distributed. Therefore, it is just a pointless ripple.

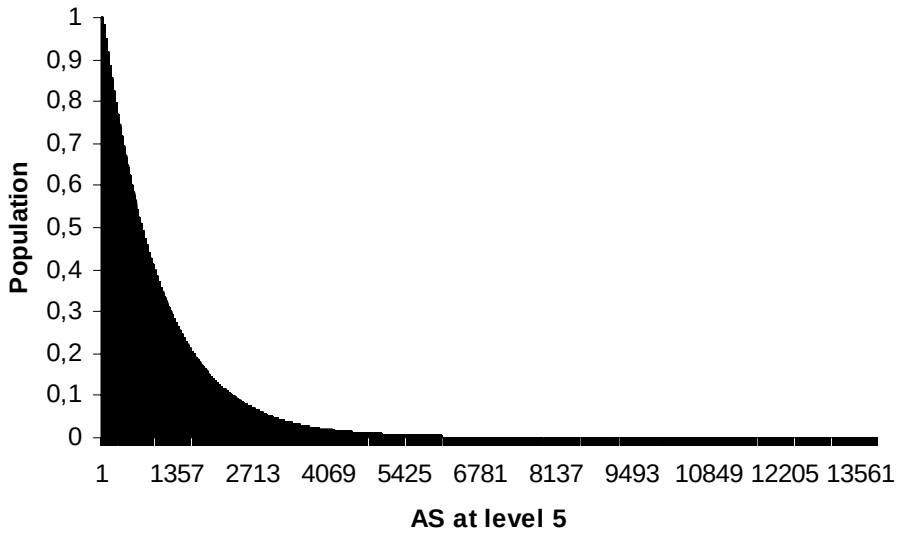

Figure 6.2. Exponential distribution of hosts through the ASes at level 5.

If applying to the global data set of the Internet, the aggregate results per level would remain. In addition to this, the individual values per AS at high levels would tend to the same ones, as the different paths would start mixing and converging. Hence, the only effect would be at individual ASes at lower levels. Of course, these low-level ISP providers would have to supply more or less resources (storage for the entries and processing power for the lookup requests) depending on the real population that they are attending.

The conclusion is that the way in which the population is distributed among the ASes in the fifth level has no effect on the overall performance of the model. Hence, the only distribution used in the remaining was the even one. This allowed a reduction in the code running time and in the complexity of the implementation, which would not turn, otherwise, into any higher quality achievement of the results.

6.5- AS DENSITY IN THE AS SPACE

The usage of sparse matrices contributed to the feasibility of this study by using common PCs. Otherwise, this work might have required non-conventional processing power (for example, from one of the servers in the Laboratory). It reduced the total amount of memory to run from 1.5GB to 100MB (approximately, and including the Virtual Memory).

Figure 6.3 is a representation of the main matrix in memory. For each AS in the *y* axis, it shows the ASes that are connected to it. Therefore, it provides a means of visualizing the density of ASes out of the maximum possible amount and the degree of connectivity that they have.

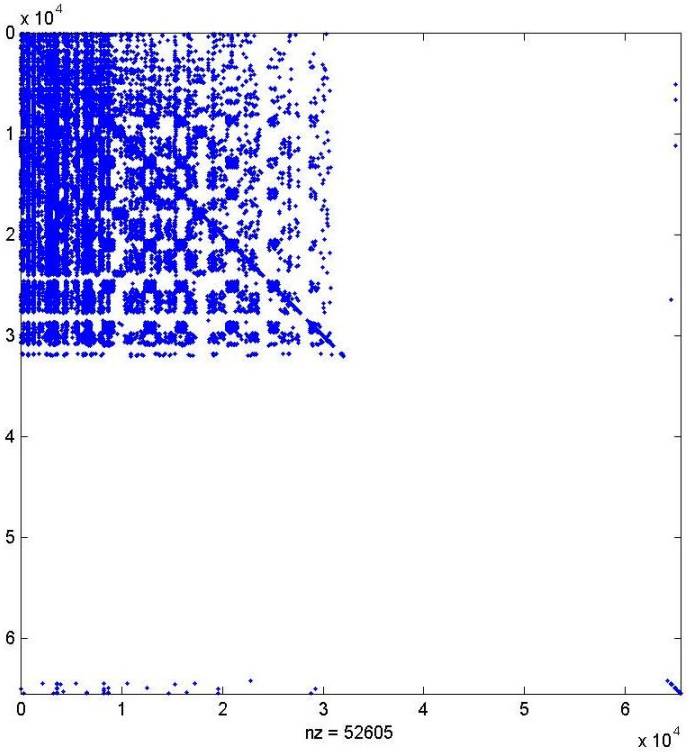

Figure 6.3. Provider matrix.

6.6- PLAIN RESULTS. DETECTING DUPLICATES

According to the description of the naming and addressing architecture in Chapter 4 and

to the details on the TurfNet in Chapter 3, a host that wants to establish the communication with another host, which does no belong to the same local domain, has first to locate it in the network prior to starting the communication.

This location is provided through hierarchical paths, which means that each lookup request is forwarded from the original AS at the bottom of the hierarchy and until it is resolved, to the provider ASes. This section focuses on the inefficiency of this basic method and the necessity of a refinement.

The problem that this method has is that ASes have as a rule not only one but many providers. Each AS tries to locally resolve all the incoming requests. This means checking whether it knows the location of the peer host or not. It turns into an extensive usage of resources (*i.e.*, performing lookups in the local registers). Those lookups that cannot be locally resolved are resent to the providers once again.

After analyzing this situation, it turned to be an inconvenient. As shown in Figure 6.4, those unique (*i.e.*, single origin and single destiny) requests for resolution that cannot be resolved in intermediate levels reach the top of the hierarchy (*i.e.*, levels 1 and 0, if existing) through several paths during a certain delay window. Hence, even though they are, indeed, the same request, they are treated as different requests and they are either resolved or re-forwarded. However, only one communication data flow can beneficiate from it (usually, the fastest). The others will be discarded. Thus, they are just a waste of network resources.

This situation becomes evidently noticeable when it occurs at the first levels. ASes at level 2 typically have a high degree of connectivity with those in level 1, which are just 22. Therefore, the likelihood of having many duplicates is already high in there. Furthermore, all those requests that have to be forwarded to the 0-level will merge into a single point from 22 ASes at almost the same time. This could even make the system collapse.

As a proposal, a new improvement could check whether there is a collision in the system. This means that for each new request arriving, the system first tries to verify if in not much time this same request was already received. Thus, it should be considered as a duplicate and

discarded. For instance, it might keep a list of the most recently attended lookup requests (or a fingerprint out of them) for resolution during a certain configurable window time.

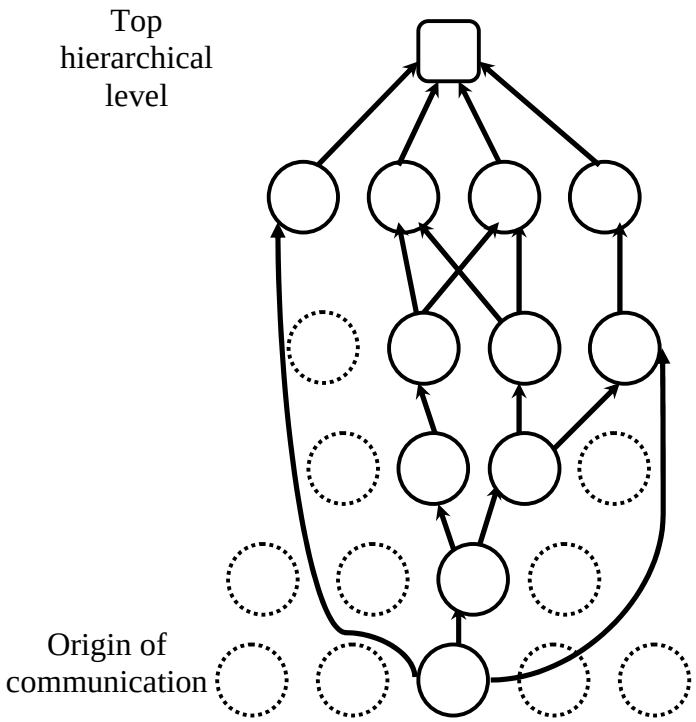

Figure 6.4. Multiple paths from an origin to the top of the hierarchy.

Apart from saving requests being solved through the parent links, which the providers would strongly appreciate, it could even have a second potential advantage. It is clear that the possible bottlenecks of the system are in levels 1 and 0 (due to the hierarchical behavior of the naming and addressing architecture). Thus, the tables in those ASes are large and performing lookups of identities in them consumes resources in terms of processing power. However, a previous check up on the request would be faster. Those lookup requests that manage to pass this filter would then continue the usual resolution process (*i.e.*, the local domain would look up the peering identity of the request in the local tables, and then it would either resolve or forward them) as they are considered as unique.

The whole procedure is shown in Figure 6.5, and Table 6.2 shows the improvement obtained if performing this detection of replicas and with the notation from the previous figure. In case that checking duplicates really saves time, the comparison should be made with the attended requests. Then, the detection of a replica has a repercussion in the same AS in which it is performed. Otherwise, if that operation was also time consuming, the comparison should be made with all the arriving requests. Then, it is the provider AS the one that beneficiates from this strategy. This is why both sorts of comparison have quite similar values, despite a 1 level delay. In addition to this, the detection of replicas shows a performance of 1152% in the best case and 88.68% in the worst case for a level 0 to which all the non resolved lookup requests are forwarded from level 1.

On one hand, it shows that this improvement only requires being implemented at high levels. Levels 3 and 4 skip it, as there are not that many different paths to higher levels crossing them. On the other hand, there is a remarkable snowball effect. If not detected, the duplicates start multiplying as they pass through different levels and they become added later on when they merge.

This section proofs that detecting these duplicates provides a means of having the bottlenecks under control. Otherwise, the amount of redundant requests being received at the upper levels would render the scheme infeasible.

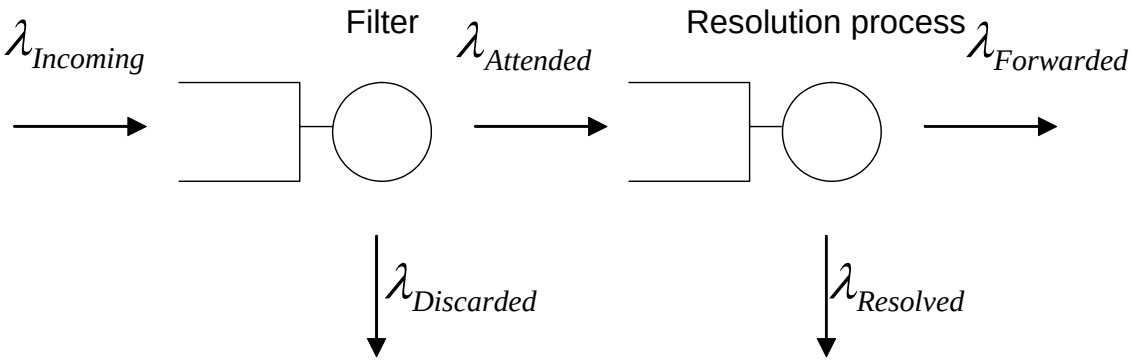

Figure 6.5. Filter for detection of duplicates.

From here on, this chapter analysis the results obtained by applying this detection of duplicates (*i.e.*, the *arriving* requests per second are not the *incoming* as they should be, but the *attended*). Later on, a second strategy that would overcome its usage is compared to this one.

AS Level	0	1	2	3	4
Compared to the arriving requests [%]	88.68	9.78	1.57	0.00	0.00
Compared to the attended requests [%]	1152	88.69	9.63	2.69	0.00

Table 6.2. Performance of the detection of duplicates as a relative mean number of duplicates.

6.7- LOOKUP REQUESTS

The frequency of the lookup requests (λ_x) are one of the metrics that were chosen to characterize the naming and addressing architecture. They show the amount of requests per second that arrive at a certain AS. Thus, they can find a translation in terms of operations per second or final processing power that is required to perform these operations. Hence, they measure the feasibility and the required resources that each domain would require to support the features of the new architecture, or, on the other hand, become a proof of the necessity to improve the methodology due to some bottlenecks.

After running, the tool provides the exact amount of lookup requests for resolution that arrive, are attended, are resolved, or are forwarded per AS (according with Figure 6.5) and second. At low levels, there is some mobility regarding the AS numbers. However, the highest ones are much more stable. Thus, the identity of the ASes becomes significant. For instance, Table 6.3 shows a list of the 22 major service providers that are considered as the level 1.

AS Number	Entity
174	PSINET
209	ASN-QWEST

701	ALTERNET-AS
1239	SprintLink
1299	TCN-AS
2828	XO-AS15
2914	VERIO
3320	DTAG Deutsche Telekom AG
3356	LEVEL3
3549	GBLX
3561	CWUSA
4200	AGIS-NET
4323	TW-COMM
4565	EPOCH-INTERNET
4637	IAMERICA-NET
5400	BT European Backbone
5511	France Telecom
5650	ELIX
6453	TELEGLOBE-AS
7018	ATT-INTERNET4
7911	WCG
8220	RIPE-ASNBLOCK7

Table 6.3. Relation between AS number and entity for the original data set.

These providers in the first level are the ones getting the most lookup requests for resolution. They are the most affected for having a purely hierarchical method for resolution.

As an application scenario, the herein presented per AS graphs (Figures 6.6 to 6.9) show the amount of requests that arrive at each AS in the hierarchy. The scope parameter was set to 2 in all the ASes (the effect of the scope is validated later on in this section and it was described in Chapter 3 regarding the horizontal composition of networks).

Figure 6.6 allows an identification of what load should each of the main ASes (identified on the *x* axis) be able to handle. This value is typically around 1 million requests per second and reaches 3.5 millions in some cases. Hence, the variance is relatively small compared to what happens inside the other levels.

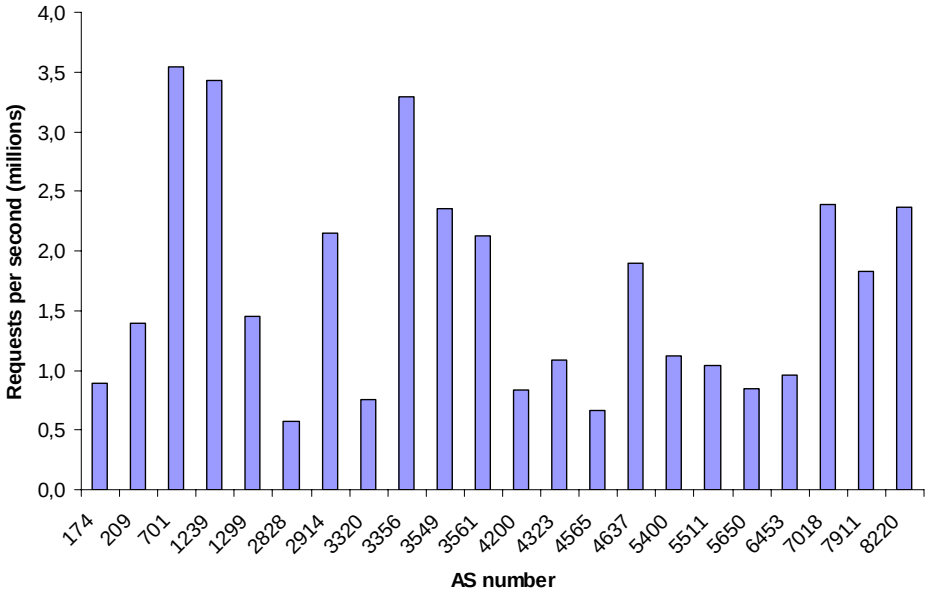

Figure 6.6. Lookup requests arriving at ASes in the first level.

On the other hand, the values for level 2 present the largest heterogeneity. Up to four ASes in this level are even higher than some from level 1.

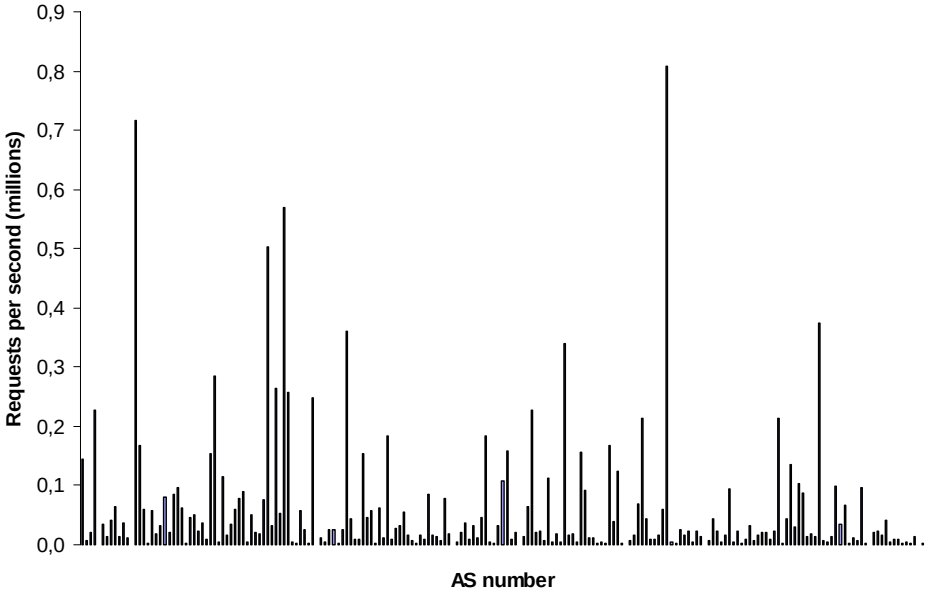

Figure 6.7. Lookup requests arriving at ASes in the second level.

There are two plausible explanations for this. First of all, the classification into levels from [SUBRAMANIAN2002] is arbitrary. Actually, it is not an algorithm but a heuristic. For instance, level 1 providers are classified according to the in-level connectivity. It means that those in the border are at one or another AS level according to a single missing or existing link. Having a look at the AS assignments is sufficient to realize that some of the service providers holding a second level AS are well-known large-enough companies to be considered first class ones. Thus, it can be taken for granted that the sorting out that it performs is not perfect. Levels are a conventionalism. When a domain registers as an AS it is provided an identifier, not a level.

Second, the real amount of lookup requests would depend on the real population in the final ASes from level 5, and the type of traffic that they produce. Some of those could be much larger than others are and originate a bias effect in the results. However, as the possible hierarchical paths cross and multiply many times, the higher that an AS is located in the hierarchy, the softer that this effect should become.

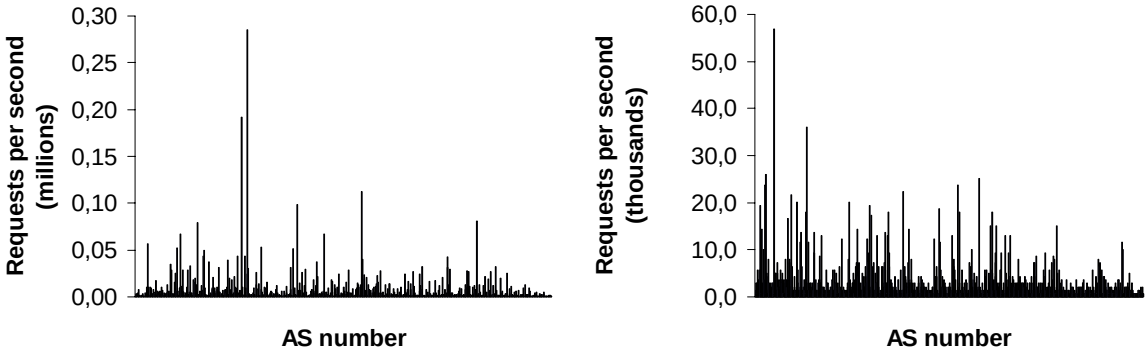

Figure 6.8 and Figure 6.9. Lookup requests arriving at ASes in the third and fourth level respectively.

In levels 3 and 4, there are also some peaks that distort the individual graphs. Once again, these single values could be considered as wrongly placed ASes.

A possible solution for this effect could be a by hand reassignment of some ASes into another level. Nevertheless, it would require a previous validation of the real identity of the AS to

avoid any undesired effect.

For a better appreciation of all the involved details, Figure 6.10 shows the same scenario of application as a Cumulative Distribution Function. Level 1 is the most homogeneous one (less difference between its maximum and minimum). Levels 3 and 4 present a rather similar shape. Level 2 shows a wide range of ASes, from those with a small amount of requests for resolution and up to the point of the first providers. At 90% of the ASes there are less than 5, 18, and 158 thousand requests per second to be resolved at levels 4, 3 and 2 respectively.

In addition to this, level 4 shows a stepped behavior. More than 45% of these ASes get traffic from a single AS at the fifth level. A 20% has two customers. This situation remains in level 3 and almost vanishes in level 2.

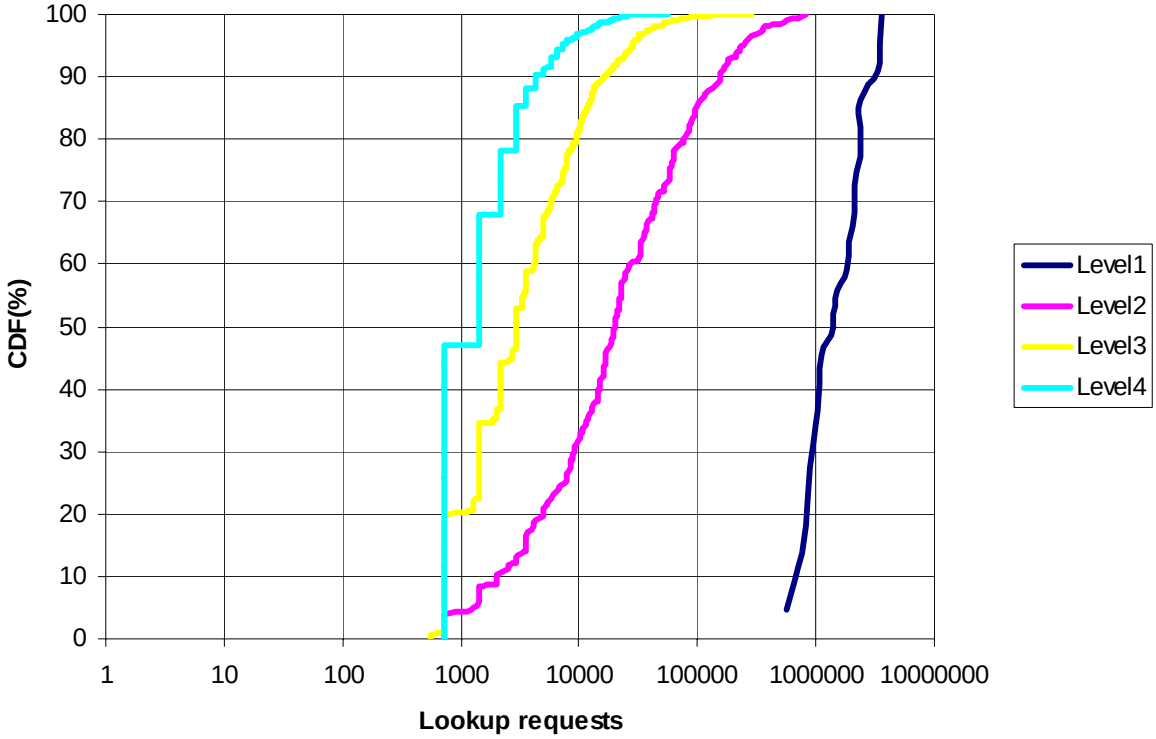

Figure 6.10. Cumulative Distribution Function of number of requests arriving at a certain level.

Furthermore, there is a common pattern applying to levels 4 to 2. An AS from level 5 can

have as its providers any AS from any other level. The common overlapped part of the graph proofs that the three of those levels have ASes that only provide service to a single user AS. Moreover, the values for level 3 that are under those in level 4 are the clear result of a stub, as explained in Chapter 4. This is a chain connection from level 5 to 3.

As a final comment regarding the variance inside levels, some extra levels could be added to provide specific support and considerations for some ASes that are the odd one out (*i.e.*, those that might belong to another level as well, because their values are in the boundary). This is the case of level 2, which might be split into sublevels. Some of the ASes in this level could be considered 1.5 or 2.5 ASes.

Figure 6.11 shows the average per level lookup requests per second arriving at the different levels of the AS hierarchy. There is a comparison between different values for the scope parameter used to achieve horizontal composition. As a simplification, only three cases are presented, as they are the most representative. Even though a different scope can be set for each AS level (*i.e.*, X-Y-Z), a unique one is introduced for all the levels (*i.e.*, X-X-X).

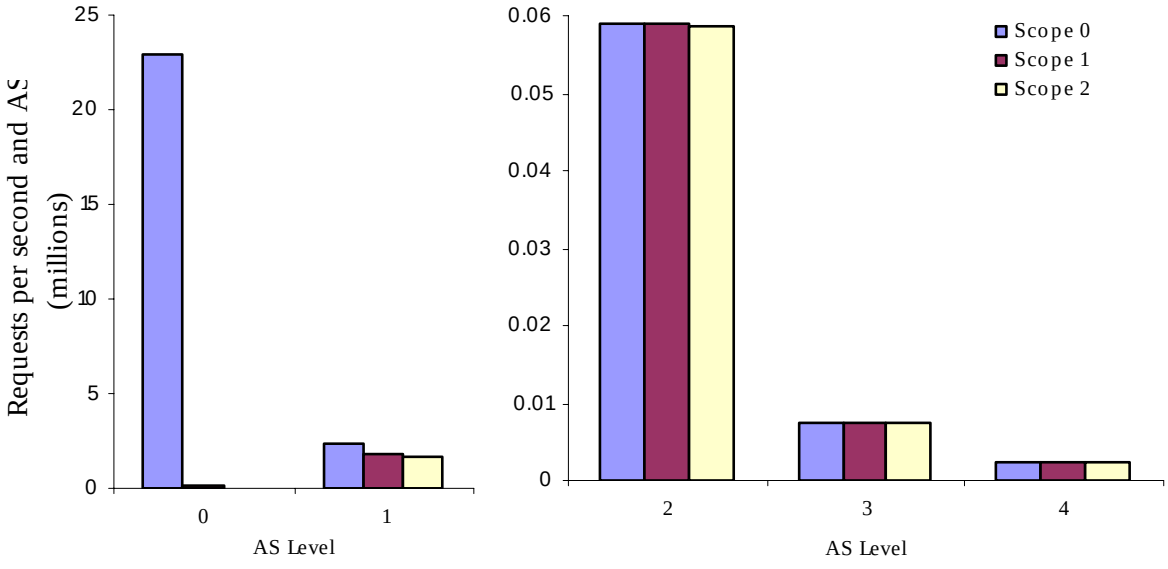

Figure 6.11. Mean lookup requests arriving at different levels of the AS hierarchy for different scopes.

The 0-level is a common database or repository that contains all the information for all hosts. This virtual level has to attend all the requests that ASes in level 1 cannot resolve. As this scheme for resolution is topology based, it always relies on a single highest-level domain that plays the role of root provider.

Around 2 million requests per second arrive at each of the 22 level 1 ASes. While some can be locally resolved, others require being forwarded to that 0-level. Up to 23 million requests per second reach that level.

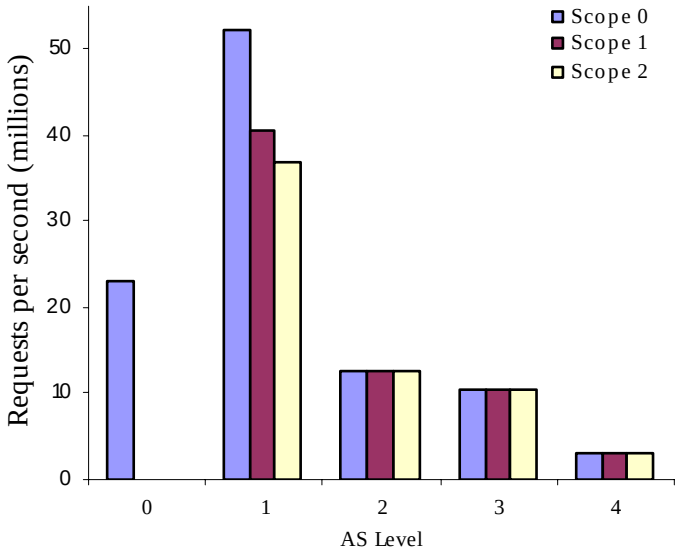

Figure 6.12. Mean aggregate lookup requests arriving at different levels of the AS hierarchy for different scopes.

However, if using just a simple scope of 1, these requests decrease to almost zero. This has two meanings. First of all, it shows that most of the queries submitted by the different ASes at level 1 are repeated. This happens as a result of the multitude of possible paths: each lookup coming from an AS at level 5 that could not be resolved through the hierarchy arrives at most of the ASes at level 1. Then, the replicas of the original request that cannot be resolved in the first level are forwarded and merge into the 0-level, potentially collapsing it. Once the system is using

the horizontal composition and, thus, the scope, the lookups that reach the common database almost disappear.

Second, even though the first level of the hierarchy is not fully meshed (it requires two hops to ensure getting to any AS from any other in that level), one hop already reduces significantly the lookups.

Figure 6.12 contains the per level aggregate average arriving lookup requests at each level of the AS hierarchy. In an interconnected hierarchical system where each AS is linked just to those in the immediately upper level, the lower levels would have the maximum requests per second arriving. Then, a part of it would be locally resolved and those remaining would be forwarded. This would happen successively until the 0-level, where those non-resolved ones would find a resolution. Hence, the expected shape would be decreasing.

However, it was already pointed out that any AS can have as its providers ASes from any higher level. This one of the reasons for which the aggregate lookups increase, instead of being reduced. The other one is that ASes have several providers. The level at which more lookups arrive is the first one. There globally reach even more requests than at level 0. It fits the behavior of these ASes as the major providers, as they have many final customers to attend.

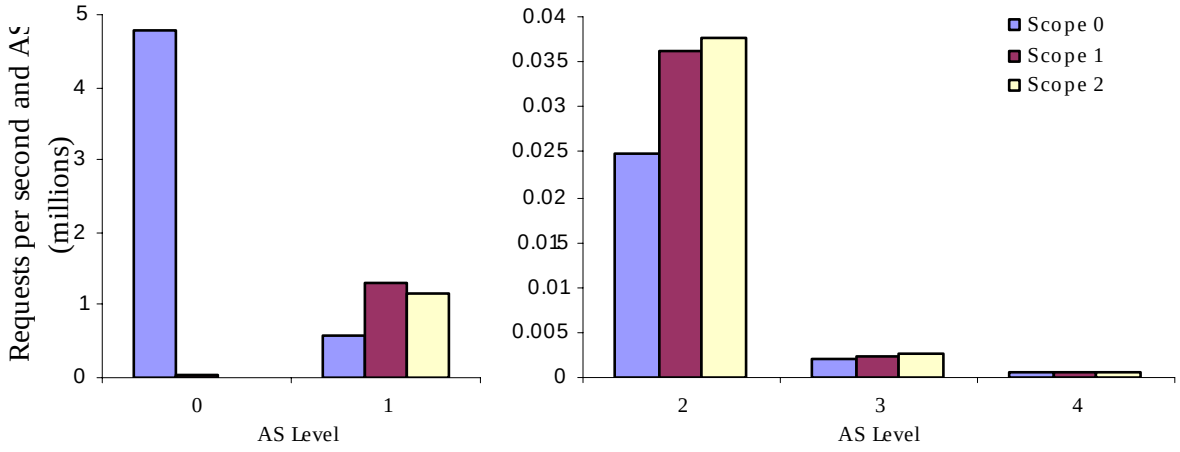

Figure 6.13. Mean lookup requests resolved at different levels of the AS hierarchy for different scopes.

Figure 6.12 also proofs the performance of peering. In the 0-level, no request is necessary any more. In level 1, it reduces more than 10 million requests per second out of 50 million. It happens thanks to the effect of level 2 showed in Figure 6.13. The usage of horizontal composition allows many more lookup requests to be resolved at that level. Therefore, it reduces those arriving at first level.

There is a second order behavior in this. As a rule, it makes sense to say that the more scope, the more lookups that are locally resolved. However, while it applies to levels 2 and 3, level 1 does not accomplish with it at all if comparing scope 1 with scope 2. The explanation is that the effect of having a slight reduction in level 2 amplifies due to the several existing paths to level 1. This has more impact than a change in the scope in the first level from 1 to 2. Most of the ASes in this level can already be reached in a single hop. A second hop does not improve much this performance.

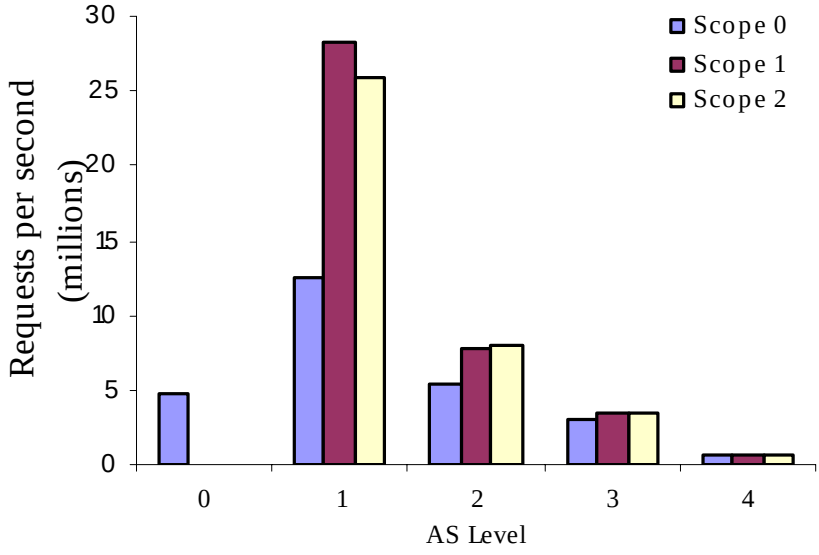

Figure 6.14. Mean aggregate lookup requests resolved at different levels of the AS hierarchy for different scopes.

Figure 6.14 reaffirms these statements. It also shows that levels 2, 3 and 0 resolve the same aggregate order of lookup requests, around 5 millions. If using composition, the first level doubles its contribution to the system.

A comparison in the lookup requests per second arriving and those being resolved at levels 0 and 1 shows a high degree of redundancy, and thus inefficiency, due to the amount of duplicates that are created. Level 0 receives 20 million requests and resolves 5 of them. For level 1, the proportion is 50 to 12. In both cases, it is around a 25%.

This shows that, even though detecting the duplicates moderates the usage of the network resources (according to Table 6.2), the architecture requires another even more proper solution to handle the hierarchical resolution scheme and increase its efficiency. Another proposal is presented later on in this chapter.

6.8- SIZE OF TABLES

Using the horizontal composition to improve the performance of the lookup and resolution scheme of this hierarchical naming and addressing architecture has, on the other hand, an implicit side effect.

Even though it decreases the number of lookups per second to be resolved, it increases the size of the registry tables that any domain has to keep ($S_I(AS)$). The more scope, the more that ASes inside a level share their knowledge of the network by spreading it. Hence, the more resolutions that are locally made.

Figure 6.15 shows the average size of the registration tables in each level in terms of entries. As expected, the possible bottlenecks are in levels 0 and 1.

The 0-level requires a 100% of the entries. It must be able to resolve any lookup request that could not be resolved in the first level. Hence, all the hosts from the fifth level of the hierarchy have to register there to achieve global reachability.

If the architecture applies a scope for horizontal composition as a means of reducing the amount of lookup requests per second in the highest levels, which is the other possible bottleneck, then the first level ASes also require one of those tables. With just a single hop in level 1, the 22 tables almost reach the same size of the one at the 0-level. They complete the small missing fraction when using a value of 2 for the scope, as then any AS in the first level can reach any other AS inside that level.

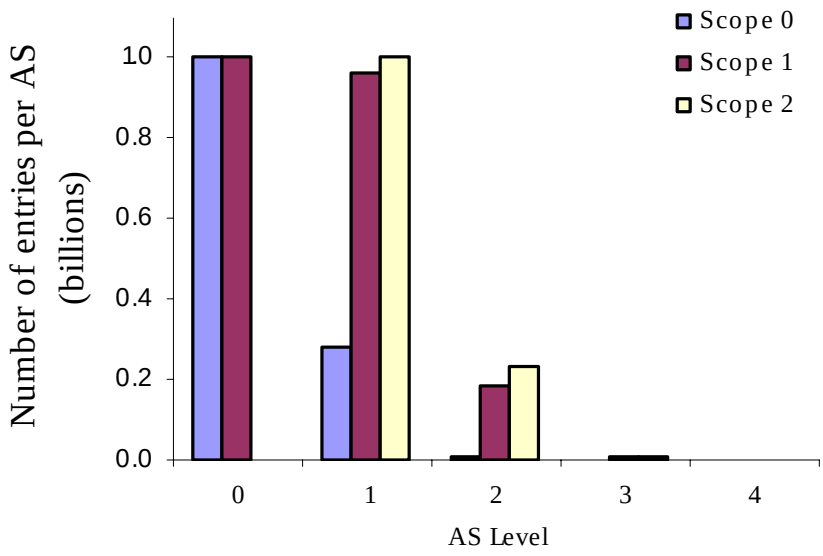

Figure 6.15. Mean registration table size per AS at different levels of the AS hierarchy for different scopes.

The most noticeable and positive effect of the horizontal composition regarding the size of the tables (it was previously described when concerning the amount of lookup requests) appears in the 0-level when applying a scope of 2 in the first level. Then, any of the domains in this level can resolve all the incoming lookup requests without having to forward any of them to the 0-level.

Hence, the 0-level is not required any more and the architecture can avoid its usage. In this way, using peering gets rid of the most inconvenient bottleneck element of the AS topology.

In Figure 6.15, the size of the 0-level is 0 for a scope of 2. This saves the more than 20 million requests per second arriving at a single AS. Thus, it reduces the bottleneck into less than 2 million, which are the requests at ASes in the first level.

Somehow, this works as some sort of distributed system. Instead of having a single top domain, the tables are shared so that each AS is able to resolve its lookup requests on its own without having to overload an upper level. Then, the system has 22 similar tables that resolve one part each one.

Concerning the feasibility of these tables, a quantization would require more details on the size of each identity (depending on the concrete address space) and the extra information to be kept associated to each one for its location (directly related to the next generation Internet proposal). However, an approximation using identities the same size as addresses in IPv6 would mean some Giga bytes. This could fit in any conventional HDD and even in volatile memory (for faster access), as the top providers can easily afford this sort of system resources.

In addition to this, and as a hint, [GOOGLE] is nowadays already indexing over 2 billions of images, apart from other data regarding websites themselves, and attending user searches over them in a feasible environment.

6.9- EFFECT OF THE SCOPE

In order to measure the effect of the scope, this section analyzes how the tables at the different levels increase their size as the peer relations are more used. Thus, and on one hand, this justifies the utility of using the horizontal composition, as the increase in capacity to resolve lookup requests shows up. On the other hand, it provides a means of quantization to choose a proper value, in general terms, when spreading the registration tables, as it outcomes a scheme to validate the performance for different values of the scope.

Therefore, the following figures show the size of the registration tables in terms of hosts that have been registered in there. Thus, this is the population over which the local AS is able to resolve the incoming requests.

As an obvious result, there is a clear compromise between the amount of requests that can be locally resolved, and the desired size that turns out to be. However, the storage of these tables into memory is much easier than handling the lookup requests.

Figure 6.16, shows the performance of level 1. Without peering, each AS is able to resolve 30% of the incoming requests. The remaining ones have to be submitted to the 0-level. Once a value of 1 for the scope is used, almost the 100% of the requests find a local resolution. This is due to the highly meshed shape of the level. Nevertheless, there is still a residual amount of resolutions that is not attended, which forces to keep the virtual level (even though, at least, without so many lookup requests per second to be handled as before). Once a value of 2 is used, the knowledge of the local AS on the whole topology reaches the 100% and the 0-level can be omitted.

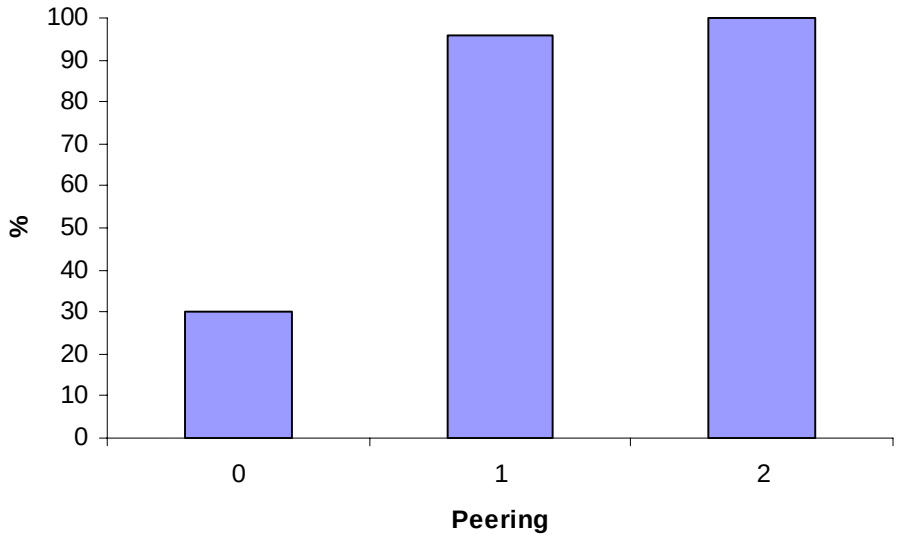

Figure 6.16. Percentage of known population by using a peering scope (at level 1).

The same sort of study can be applied to the second level, as shown in Figure 6.17. Here, as there are less peering connections, the increase is both more regular (there is no big step like in the first level) and the values that are reached are lower. The behavior of the slope is roughly linear.

The value of 2 that is used in the analysis as a reference scenario shows not to have the required performance. It provides a 15% of hosts. To achieve a 50%, a value of 5 is required. Higher percentages would mean too much spreading of the registration tables. Keeping these tables up to date could turn into an overload into the system that is not worth. Hence, a value around 4 and 5 would be the perfect solution, in generic conditions, for this level.

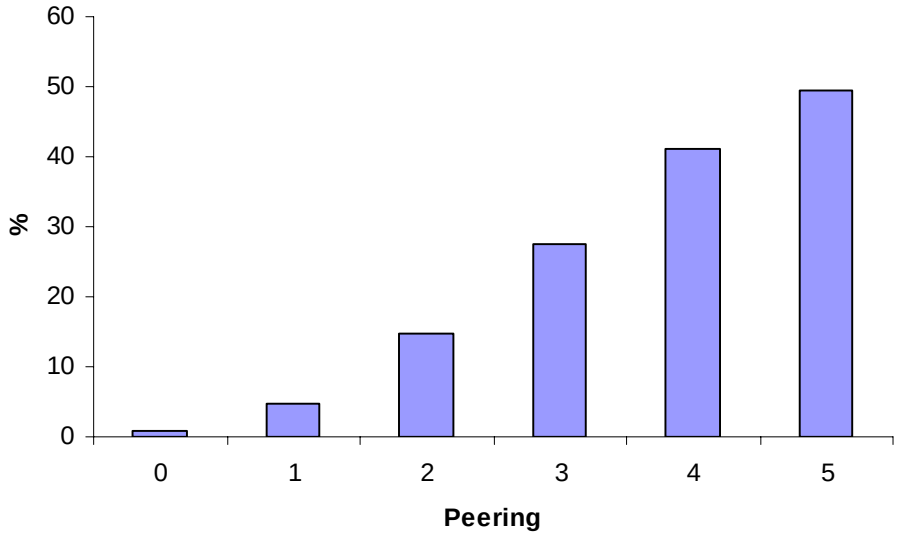

Figure 6.17. Percentage of known population by using a peering scope (at level 2).

The impact of the horizontal composition in the third level is almost unperceivable. It does not reach even 1%.

Most ASes in that level do not share any connection with the others at all. Among those which do, most of them share a single connection with another AS. This situation might correspond to ASes belonging to a same company and sharing an inter-level connection to save

their inbounds traffic from going to higher levels (either not to waste resources or to provide protection to their communications by keeping them under control).

Nevertheless, as the links are already there, they should be used. A parameter around 2 would be enough. Those ASes without peering connections would not beneficiate from the scope, but would not have any con for it neither; they would just try to propagate the registers without any positive result. Those with a single neighbor would just spread their tables a single hop; this would mean an effective scope of 1. Those remaining (belonging to clouds) could share their information with the others.

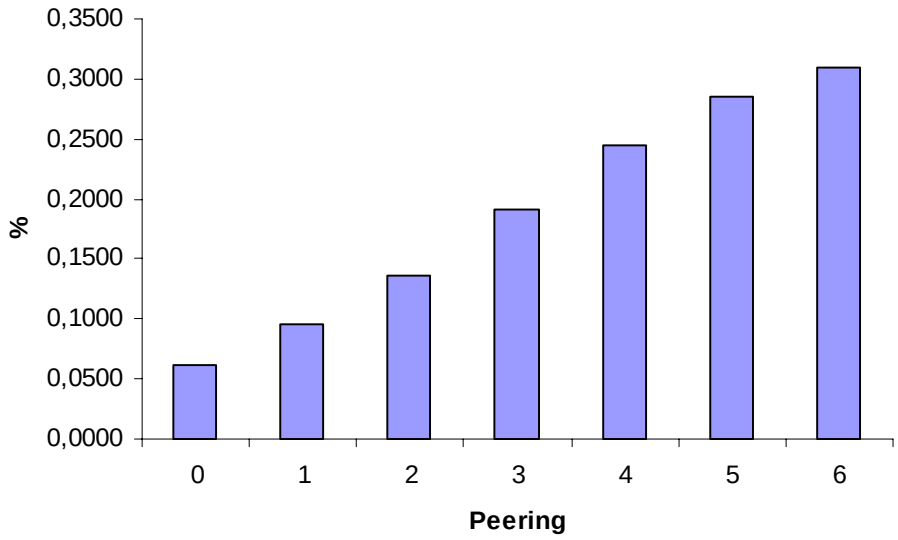

Figure 6.18. Percentage of known population by using a peering scope (at level 3).

Finally, and regarding levels 2 and 3, note that these percentages refer to the amount of hosts that are known at a certain AS. These are not directly the percentage of requests that can be locally resolved, as the communication pattern runs on top of it. The likelihood of connecting to some nodes is higher than to some others, depending on the distance between the end points of the communication. Therefore, the percentage of resolutions would differ from these values, as they would be larger.

6.10- HOPS FOR RESOLUTION

This section evaluates the performance of the hierarchical naming and addressing architecture for the resolution of identities into locators in the lookup process.

When a host tries to locate another one, it sends a lookup request. If the host does not belong to the same AS, the lookup propagates through the hierarchy as it is submitted to all providers of the starting AS. At some point, the request will be resolved and the location for the host identity will be returned to the origin AS to complete the communication establishment.

Figure 6.19 shows the average number of AS hops until the resolution is achieved. This considers the fastest resolution among all those that will appear due to the spreading fact and, thus, the first one that will reach its destination.

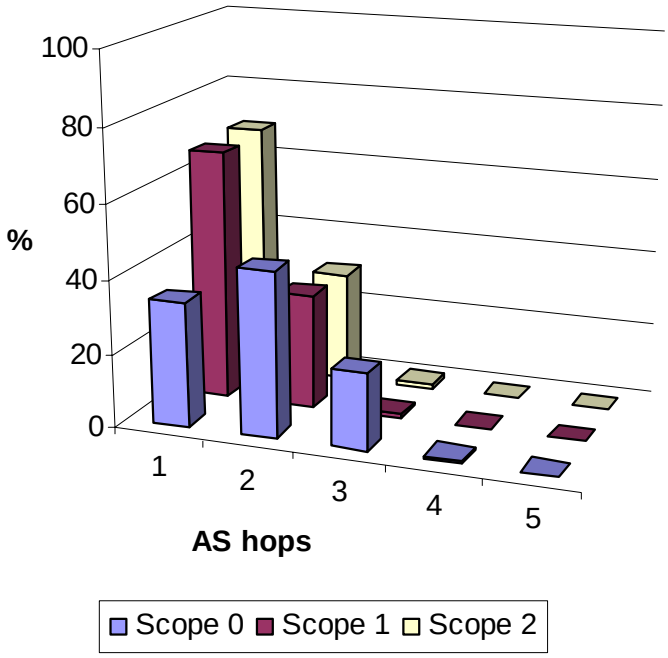

Figure 6.19. Number of AS hops until resolution depending on the scope.

The average values until resolution are 1.89, 1.33 and 1.32 for scopes 0, 1 and 2 respectively. As achieving the communication requires a full round trip, the final mean values for the resolution of an identity and beginning of the communication are 3.78, 2.66 and 2.64.

In addition to this, the values for 3 and more hops are almost zero. Hence, it is unlikely that any request needs more than 2 hops to be resolved.

It is noticeable that without any horizontal composition, the average value is around two. Once some scope is used for the peering relations, most requests are resolved in a single hop. On the other hand, scopes 1 and 2 are quite similar. The main reason for this effect is that using scope in the first level cuts down the amount of requests going the 0-level all of a sudden. Therefore, this turns into almost a one hop bias.

This behavior can be reviewed in Figure 6.20. It shows the Cumulative Distribution Function of the number of lookup requests that are resolved at a certain amount of hops from the origin AS. Both curves for scopes 1 and 2 do not differ in excess. Over 90% of the requests find a resolution in less than 2 hops in both cases. Without any peering at all in the system, the same percentage requires 3 hops.

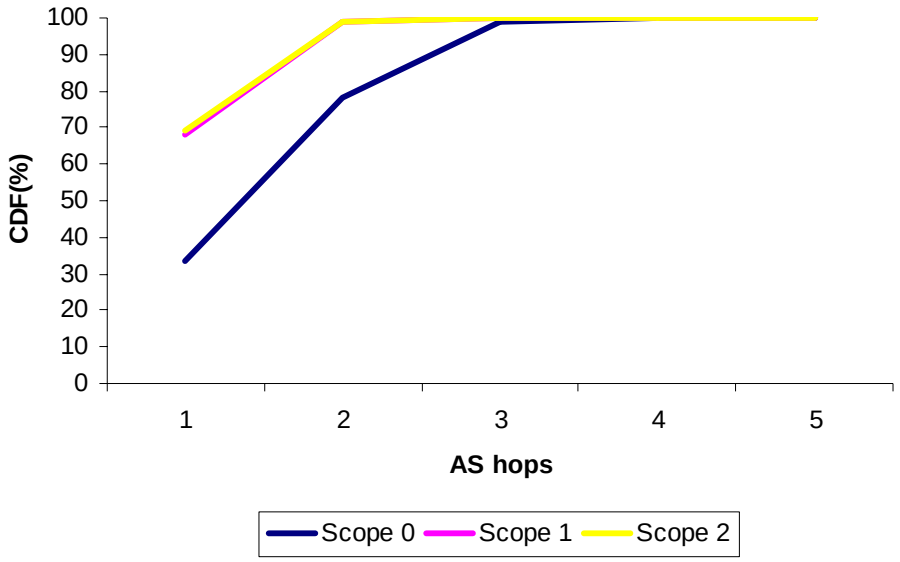

Figure 6.20. Cumulative Distribution Function of the AS hops until resolution depending on the scope.

As a main conclusion, the hierarchical scheme shows up as a highly efficient methodology to achieve fast resolution of nodes.

In addition to this, and in comparison to the system employed nowadays in the Internet (DNS) or related ones (HIP [MOSKOWITZ2005]), the resolutions perform in a single round trip. DNS and similar methods typically require several iterations to identify an identity, as all the domains and subdomains will require their own identification until the final and really useful information regarding the host is obtained. Furthermore, and according to [ZEITOUN2004], it has been proofed that the path delays mainly correspond to intra-AS delays. Hence, an efficient method to reduce the number of traversed ASes would have a significant impact in terms of network performance and latency reduction. To improve its performance, DNS uses cache systems. Nevertheless, the hierarchical scheme (at least in TurfNet) also considers the usage of an implicit cache in all ASes that are traversed by the communication flow.

Hence, this is another key issue to be considered. Although the hierarchical system implies overloading the highest levels of the hierarchy, on the other hand it provides fast resolutions.

6.11- IMPROVED SCHEMES

As seen in the previous section, the naming and addressing architecture considered in this dissertation does not only provide a platform for a next generation Internet and all the corresponding benefits, but it quickly resolves the lookup requests.

However, Section 6.7 showed that, on the other hand and as its main con, the architecture has a possible bottleneck at the highest level when handling all the requests that merge in that level due to the topology of the Internet and its interconnections.

Hence, once this feedback was analyzed, the aim of this thesis turned to be providing some means of reduction of the values on the top of the hierarchy by modifying the architectural

methodology. On the other hand, the fast resolution behavior had to be kept, as the main condition for any improvement.

The solution was an arbitrary enhancement when forwarding the lookup requests. Any AS, instead of submitting to all of its providers (according to the hierarchical rules of TurfNet) decides, according to some predefined policies, to which reduced subset of them to send the lookup requests for resolution. This is modeled by the *k* parameter from Chapter 4. Some considered possibilities were:

- Submitting to a single provider, if many are available. Choosing the one with the highest position in the hierarchy among all possible.
- Submitting to a single provider per level, if there are several parents that have the same level. This would be an intermediate approach between the previous one and the normal one.
- Submitting to a single provider like in the first case. However, it gives preference to the second level before than the first. Even though this method would increase the average path lengths, on the other hand it would save frequency of incoming requests in the first level.

The first one was validated as it managed to perform with almost the same efficiency as the optimal case regarding the average number of hops for resolution, as seen in the previous section.

Table 6.4 shows the obtained results (using a scope parameter of 2). Although there are great improvements at all levels, the most significant one appears at level 1, as those ASes are the odd one out of the model: the possible bottlenecks.

(thousands)	Generic approach	Redefined method	Improvement
Level 1	1680	327	80%
Level 2	59	14	75%

Level 3	7.5	4	45%

Table 6.4. Improved method for lookup requests.

The reduction of 80% in frequency of requests provides a clear chance of feasibility to the top providers if choosing to run such a hierarchical naming and addressing architecture. This is analyzed in the following section of this chapter.

Moreover, this strategy would override the requirement of detecting duplicate requests to save collisions in the system, as they are a waste of network resources. Once the problem of the duplications is cut down from its origins, there is no need any more for filtering.

In addition to this, the application of one of these schemes should integrate the required mechanisms to ensure that the finally built point-to-point path for the user data flow is the shortest possible. Otherwise, both the communication delay could be larger than in the optimal case and the amount of traffic in some random links could increase.

This could be implemented using and adaptive system based on [MAO2004]. Once the communication has started (with the optimal latency provided by the hierarchical naming and addressing architecture), it would check for (while transmitting data and in a transparent way for the user) known ASes in the path that can be reached through less AS hops, if possible.

6.12- FINAL CONSIDERATION ON THE TABLES

The weak point of any hierarchical architecture is, evidently, its top level. Overcoming this situation first requires a review of the state of the art regarding the databases that should be able to handle the storage and attend the high frequency of address lookups.

As a reference, [MYSQL] has confirmed *friendster.com* as one of its top solutions of 2005. It claims to handle over 1.5 billion requests per second, which translates into more than 17,000 requests per second. Other router statistical tools for Internet control show average values around 30,000 requests per second being processed.

Nevertheless, whatever that it is the real maximum amount of frequency of requests to be handled that the most powerful database and server can offer nowadays, it might not be enough.

The conclusion is that the order of magnitude of levels from 5 to 2 could be managed by normal equipments. On the other hand, the first level (which is still at an order of magnitude above, around 300,000 requests per second) is a challenge as it could require specific on purpose hardware or even the usage of a method for distribution of the registration tables.

Distributed Hash Tables (CHORD [STOICA2003], KOORDE [KAASHOECK2002], FISSIONE [LI2005]) are one of these technologies for distribution of contents. They currently apply to peer-to-peer networks in the Internet and their efficiency has already been proofed.

7. DISCUSSIONS

7.1- INTRODUCTION

Chapter 6 evaluated the feasibility of novel naming and addressing schemes based on a hierarchical behavior for a new Internet architecture and its application to the TurfNet. However, there were some assumptions that were taken when building the model in Chapter 4. This chapter analyzes these possible restrictions and their effects over the results.

Additionally, this chapter proposes possible solutions for technical requirements that appeared while verifying the functionality of the system, mainly when trying to reduce the negative effects of this hierarchical approach.

Finally, this chapter makes several proposals for future research lines that should be made in the field of new Internet architectures. It particularly remarks those that are related to the usage of a hierarchical naming and addressing scheme.

7.2- RESTRICTIONS OF THE SYSTEM

Several assumptions were taken in Chapter 4 when modeling the system. Although the all of them were justified, some might require specific attention to realize what would happen if running a real scenario.

Non-static approach

Chapter 4 assumed that hosts register once to get a global identity. From then on, they keep the identity, which means that:

- There is an infinite cache time in the provider ASes.

- The host does not explicitly ask for disconnection.

However, some hosts might be registering often enough so that the registering requests should be considered. Such is the case of mobile devices performing a high number of inter-AS handovers.

Nevertheless, the AS topology has as its inputs commercial, political, social… relations, and geographical ones, as well. Hence, geographically close ASes are also at a small AS hop distance form one another.

The conclusion is that re-registering would only affect low levels. Therefore, it would have no effect on the bottlenecks, which are located at high levels.

On the peering

Following the last item and as a first peering consideration, there are hosts for which it can be presupposed that they are going to be accessible for a reduced amount of time. The host itself and the network control might take this decision. Therefore, they could decide not to spread their identities when registering through horizontal composition. Otherwise, it would become a waste. A feasible solution would be using the peering links not when registering but when trying to locate a host.

The features of this approach are:

- It does not increase the size of the tables.
- The final communication path is the same and, thus, has the same length.
- The registering requests decrease.

However, it also has some cons:

- The lookup requests in the system rise.

- The latency increases until the lookup request reaches an AS in which the host was registered.

As a conclusion, if the architecture is able to identify a host as a perfect candidate for a high load of registrations, the system should not horizontally spread its location; only the mandatory hierarchical forwarding should apply. On the other hand, the requests from other hosts trying to locate it should be forwarded through the peers, as a means of not turning into a worse location performance than in the original system.

As a second consideration in this item, the possibility of artificially increasing the number of peering connections in lower levels should be considered.

On the other hand, adding peer links could attempt against the integrity of the system. The reason for this is that it might be breaking the basic principles shown in Chapter 2 regarding the independency of the networks. While high-level ASes are global providers, smaller ones have commercial implications and duties. Therefore, no general rule can apply; it should be specifically validated what is the real situation in each case.

Furthermore, the non-economical but topological aspects have to be considered:

- Network growth required to achieve a certain level of population knowledge (as described in Chapter 4 and 6). Each AS at a certain level has an average intrinsic knowledge through the hierarchy. The additional one comes as a result of peering.

$$\beta \propto f\left(links\ per\ AS, actual\ scope\right)$$

Therefore, to increase the efficiency of the horizontal composition, the network would require a certain amount of new connections. Having a high degree of connectivity is relatively easy at high levels (*i.e.*, an almost full mesh in level 1 and almost a single AS cloud in level 2) as there are few ASes (22 and 215). However, in low levels there are thousands of ASes. The formula for number of connections in a fully meshed

network is meaningful in this case, as it shows a quadratic growth with the number of nodes.

$$\#links = \frac{N \cdot (N-1)}{2} \approx \frac{N^2}{2}$$

Additionally, the changes in the topology are much more likely to happen in low levels. This points the weakness of low-level agreements.

- Traffic increase. In vertical composition, mechanisms to distinguish the different providers for an AS exist. They allow taking smart decisions to decide to which parent a query has to be submitted to. This enables a traffic control to reduce the amount of lookup requests. Actually, this was the main aim of Chapter 6. However, no immediate parallelism can be applied to horizontal composition, as all neighbor ASes have *a priori* the same chance of being the optimal one for a fast resolution. Thus, the more peering connections that we have, the more registering requests or lookup requests per second to be attended.

Other possible metrics

Even though Chapter 6 also had latency concerns, the main metrics for the model were agreed to be the frequency of the aggregate lookup traffic in the ASes and the size of the registrations tables in each AS in terms of identity entries.

However, there are other metrics that despite not applying to the naming an addressing architecture are related to the final communication. Regarding the later coming user traffic (actually, the real payload) that would follow the location of a host in the system, it could be additionally considered the communication throughput and the required state in the inter-domain gateways. The throughput is the amount of traffic that reaches destination per time unit. The state

in the gateways is kept according to a cache time regarding the type of traffic traversing them (at least in the TurfNet). Thus, both depend on the user traffic characteristics.

To model the real traffic, we should have to switch to another solution from those three explained in Chapter 4: a simulator, an implementation or an extension of the features included in this on purpose solution (although the complexity would render this possibility infeasible).

In addition to this, it would require not the statistical plain approach that was explained in Chapter 4 to define a user and its traffic, but real user traffic patterns for different types of application. Peer to peer, best effort, video on demand, VoIP… Only a both fair and reasonable combination of statistical behaviors would not mask the results.

7.3- TECHNICAL IMPROVEMENTS AND SOLUTIONS

In Chapter 6, some items appeared regarding technological aspects not directly related to this scalability study. The herein discussed items are technical solutions that could apply to the architecture in order to provide both functionality and feasibility to the system, or at least to increase the ones it has right now.

Somehow, several requirements showed up during the analysis due to key aspects that had to be handled in order to, basically, decrease the amount of lookup requests in the higher levels. They find in this chapter a solution or at least an explanation on how they could be deployed. Even though these might not be the optimal solutions, they could suggest the reader other possible operative proposals.

The limitation of this study is the status of the architecture itself and the broad view and application field of this study. These are some hints for a better understanding:

- Even though the example over which the study was held was TurfNet, the aim of this thesis was driving a generic study on scalability of new Internet architectures that require a different naming and addressing scheme from those solutions that exist nowadays. In a first stage, all definitions were kept as generic as possible, while in a

second stage they were implemented following a hierarchical scheme that fits, for instance, the TurfNet architecture. Thus, to save the generic *status quo* of this work, no solution can use specific architecture dependent details, as far as it would require being out of sight and losing the broad view of the situation. The appliance of too in detail solutions would definitely turn the conclusions from the reader into apparently misleading.

- Technical decisions that are taken only concerning scalability issues might have side effects over other future work in the field of the new Internet architecture being considered. This means that the considered best solution regarding scalability could achieve its worst effects when security, management, interoperability or any other concept were considered.

Therefore, there is an implicit compromise between the decisions that can be taken when drawing up a technical proposal and the consequences that it would have in terms of reducing the wide range of this work. For instance, a proposal might require the usage of a central node or device in each domain for control aims. While this is true and it already exists under TurfNet, as it has a namely called TurfControl, it would render a limitation to other possible architectural scenarios.

As a conclusion, only some hints will be provided regarding the items herein presented. In case that some assumptions are taken as a means of setting an example, this sense will be explicitly reinforced.

On the other hand, several improvements can be easily deduced from the nowadays use of NATs, BGP routing… The know-how of years of real life experience are one of the most valuable and helpful information sources for configuration of an Internet related system.

Maintenance of the scope parameter

In Chapter 6 it already became obvious the necessity of using a scope parameter to avoid

an excess of sideways propagation of tables that would not be realistic. However, this is not an inner parameter of the topology, but the result of an arbitrary decision when building up the model.

Furthermore, it is of common sense realizing that the links for those peer connections are already established nowadays in the Internet. Thus:

- The connections exist; they do not have to be created. Therefore, they are not an added cost in economical terms. The Internet domains or ASes already possess these business agreements.
- Moreover, there is no setting up and no infrastructure change required for the peering to work.
- Not using them is a waste. Even though the resolution requests could be sent only through the hierarchical connections (thus, forgetting the peering links), this would be inefficient. Such a solution would provide a fast resolution, *i.e.*, a low latency until the communication flow can start, as pointed in Chapter 6. However, the throughput might become under some circumstances higher, as the traffic would be following a potentially longer or more overloaded path.

Hence, it is clear that it has to be taken profit of the peering links. However, in which way should the architecture handle it?

The usage of provider to customer connections has been clearly specified in Chapter 3 for the system description. In the worst case, a lookup request would have to go from level 5 to the first one in the topology until there is a resolution. Nevertheless, it has an implicit limit.

On the other hand, there is no limit for horizontal composition. Alternatively, the limit is so high that it makes no sense: the whole diameter for a level. Chapter 6 analyzed the use of different scope parameters. However, it would be interesting having some mechanism to provide the optimal peering in each case.

The optimal solution would be an adaptive algorithm mainly considering the following inputs:

- Host identity preferences.
- Traffic flow characteristics, if possible to guess them *a priori*.
- Individual AS policies.
- AS level general policies.
- Tables size increase factor.

The algorithm would care of a proper weight of each of the parameters from a default starting value. Having as its comparison threshold the hierarchical resolution, the horizontal one should try to provide a faster performance without excessively increasing the size of the tables. There must be a balance.

In addition to this, the adaptive algorithm could decide to give some priority to some peering links. In this way, the peers are not equally treated any more. This would provide a fairer resource usage.

Caching systems

Somehow, the TurfNet architecture already includes having some sort of cache, which avoids repeating the whole connection proceedings. Therefore, we can save unnecessary lookup requests as far as the necessary communication state is saved in any traversed AS. Then, all lookup requests that go through that AS can beneficiate from this stored information. According to Chapter 3, this information is kept during a lifetime according to a parameter, which value is based upon several issues (e.g., type of connection to be later held, mobility of the host…). This can be a key point related to common web services of massive usage.

Other hierarchical like naming and addressing architectures could beneficiate from similar strategies to save part of the lookup requests in the high levels of the topology.

In this sense, some services of common usage could push their identities down through the hierarchy providing not only an even faster location but a less overloaded first level of the topology. Somehow, the Internet pattern for communication already points that communications are held much more locally rather than globally. Some big companies distribute their contents geographically for a better usage (e.g., Microsoft updates, Google, Yahoo, CNN…). This implies that all the requests for communication with those major service and data providers are local. Additionally, most of the communications are held inside companies; a domain or a sub-tree of domains inside the AS topology.

7.4- FUTURE WORK

This section differentiates between work that specifically applies to this study and other lines that have a broaden aim related to the system architecture.

7.4.1- RELATED TO THIS SCALABILITY STUDY

Security: privacy and DoS

Security is an issue of public knowledge and concern. The global community is aware of security related issues and any new proposal has to be first security proved before being deployed.

First of all, the architecture could provide a means of protection for the lookup requests. There could be some enhancements like in-built digital signature to avoid identity supplanting (quite a common hacking strategy to obtain online banking user keys). As both the naming and addressing architecture considered herein and the common certification scheme have a hierarchical base, a complementary usage of both might be feasible.

As a second point, it should be reviewed up to which point other users can access the global per-domain control center (*e.g.*, the TurfControl) or the inter-domain gateways, as they contain the communication state of the current connections. It is a privacy concern that might require strong security measures.

Nevertheless, and as a general rule for all these issues and any other, the first step to be taken is always checking what happens nowadays in the ASes. As we consider that the hierarchical naming and addressing architecture would make use of the current topological infrastructure, some of the questions might already have a solution in terms of ASes. Thus, an extension might be applied to the new architecture.

For instance, DoS attacks are usually held against BGP routers. Against them, BGP routers are usually organized in confederations providing, in this way, redundancy to the system in case of failure. A second functionality is a better performance by reducing the times in cue for packets that are changing domain. A large enough number of routers can logically reduce the system time concerning the inter-AS delays, which would tend to be the service time.

Other techniques that BGP routers use to assure their operability include route flapping and dampening.

Another attack might be related to the growth of BGP routing tables. Older routers in the Internet cannot cope with it, as the tables do not stabilize after a major change. This might mean an unreliable or even unavailable service for a while. The growth of the tables was exponential until 2001. After many global efforts including route aggregation, it acquired a linear behavior that allows replacement of old BGP routers (the gateways in a generic scenario).

Other measurements

The feasibility of the architecture has been already proved in this dissertation. However, there are other items that could be checked for the system performance: the overheads in the system capacity and the delays. Despite already having some hints on the delays in terms of

communication latency, it would be interesting, as well, having it not only as AS hops delay but as a time measurement.

This would require a real implementation in a laboratory with enough machines to set a wide enough range of scenarios. A simulator might turn to be a cheaper solution with quite similar results.

Nevertheless, the final results would be always scenario dependant and, therefore, tricky for a proper unbiased and non-tendentious interpretation.

The case of the overheads is similar, as it depends on the concrete settings for the algorithm characteristics, the bandwidths of the system, the frequency of the lookup requests to be attended…

The future Internet

Apart from the already discussed more mobility, a new Internet might also have more levels. The reason for this is that each user would not only have a host but a PAN (Personal Area Network) including all the devices that belong to it as a new level. Nevertheless, this new level does not mean an extra cost for the infrastructure. The devices will typically register in a small scope and, thus, their lookup requests will be almost locally resolved. Furthermore, and as a general answer to this item, having more levels is not an inconvenience. As the resolutions are usually held at few AS hops (according to the traffic patterns), having a more hierarchical topology might even reduce the lookup requests in the higher levels. On the other hand, the temporal delays might increase as more ASes are crossed.

GNU version of the environment

The application was coded under MatLab, which has proved to be the best commercial solution for high-level technical problems and has found applications in several fields and industries.

However, on the other hand, MatLab is a proprietary software and, thus, requires the acquisition of a license to run it and deploy applications.

The state of the art of other similar programming tools with a mathematical oriented behavior was checked to study the feasibility of moving the code into them. The main candidates were Octave and SciLab. These are brief descriptions of each of them:

- Octave is the GNU clone of MatLab. It is a global community effort to develop all the language features of MatLab from zero. Even though the number of functions in its libraries is quite high and relentlessly increasing, the trickiest and most elaborated ones (which are actually, those most useful) are still missing. This includes most of the sparse functionalities. Moreover, the efficiency of any function in terms of time consumption compared to the respective one under MatLab is low. On the other hand, the main pro of Octave is the code migration from MatLab, as it does not require major changes. The tested version 2.1.50 runs under Linux, and under Windows through Red Hat's Cygwin or already compiled binaries (no AMD support with these). Additionaly, it makes use of GNU Plot.
- SciLab. On one hand, SciLab provides a user friendly interface that resembles the one from MatLab. However, on the other hand, the migration from MatLab does not perform as desired. The code is quite different and requires using conversion tools that do not achieve the expectancies. Additionally, SciLab is an INRIA&ENPC consortium and it has its own license and usage agreement, which differs from the GNU General Public License. The tested version was 3.0.

Therefore, even though deploying the application under a free environment could be feasible, the price paid in terms of wasted computing time is not worth nowadays. However, the increasing number of features and the improvement of the already existing ones are noticeable and drive into optimism due to the global community effort.

7.4.2- RELATED TO OTHER FIELDS OF HIERARCHICAL ARCHITECTURES

Resilience

Inter-AS communication relies on dedicated gateway nodes that are able to relay traffic between ASes. The system depends on their correct performance. This does not only apply to DoS attacks as explained before but it could also find a weak point in case of failovers. Nevertheless, this situation is not a result of a new Internet architecture. The nowadays one already has these limitations as a result of an abusive usage of NAT middleboxes. Many home users, corporate business entities and mobile terminals with Internet capabilities are located behind NATs. Somehow, the fact that these scenarios already exist demonstrates that the deployment of the gateways is feasible.

The first solution would be the same as the one commented for DoS attacks: gateway redundancy. The system should be dimensioned to allow operability even in the case of some failures.

The second solution is deduced from the AS topology. Most of ASes have not only one provider but several of them. Furthermore, these providers are usually at different levels. One of the main reasons for this is ensuring that the communications are always available. Hence, in case that one of the links is broken, the lookup requests and the later coming communication would be driven in a natural way through the other links. This would mean, at the most, a non-optimal path for a short period of time and until the connection to the optimal provider is reestablished. In an extreme case, peering links might also be used if necessary as the main connections for communication, even though this is not their main purpose.

In addition to this, the next generation Internet architecture allows, as a rule, support for multihoming and multisourcing. Thus, it is quite likely that some other physical addresses for the same identity are available.

Mobility

Mobility support is one of the new features that any next generation Internet architecture includes. While the AS inbounds mobility can be locally held, the outbounds one requires new procedures that the architecture must be able to supply as a key point of the new network.

It must provide a means of transparent capabilities for this aim, in a way that the final user cannot notice in their communications when changing AS.

This situation could be for instance, that of inter-operator roaming, handovers, a PAN moving location…

Address space

When building the model in Chapter 4 and, later, when implementing it in Chapter 5, it was already mentioned that the system requires a global address space, able to uniquely identify the elements of the architecture (*i.e.*, the hosts). However, and apart from the requirements for a common purpose address space that came from Chapter 2, no assumption was taken. Even though the choices regarding the address space condition any proceeding in the architectural approach (registration, location, resolution, creation of the communication state…), the aim of this dissertation was that of offering a broad view of the situation without taking any unnecessary assumption that could render this work into loosing its generic applicability. Therefore, despite knowing that the address scheme has a central role in the overall architecture, the flexibility that it preserved still remains.

Nevertheless, some added conditions could increase the efficiency of the system by providing a means of better resource usage. For instance, some implicit information could be inserted into the global identities, when assigned, to allow a forecast of the identity location. This would turn into a faster and less network resource consuming host lookup resolution.

In addition to this, the local address spaces could also apply some rules to improve the usage of the architectural resources. For instance, the upper levels of the hierarchy (the first and

second ones) could decide to share a non-overlapped address mapping and a common protocol. In this way, the gateways between ASes would become mere routers. On one hand, this situation would require less communication state to be kept. On the other hand, each inter AS router would be much more efficient and, therefore, less router redundancy would be required. At lower levels, the system would go on benefiting from the flexibility granted by address and protocol translations.

8. RELATED WORK

This Chapter summarizes the current state of the art of different related work. It distinguishes between three main blocks of studies. First of all, those ones that deal with scalability analysis. Then, it continues with some additional works on the Internet characteristics that, although not mentioned in Chapter 4, can suggest some new working lines. Finally, it shows the present situation of the some of the most known next generation Internet architectures.

8.1- SCALABILITY ANALYSIS

This work presented a scalability analysis that uses real AS information extracted and deduced from BGP tables to build an Internet-like model an analyze a novel architecture in the most realistic scenario.

This has been the first real attempt to provide a means of quantification to this aim in the scientific community. Other architectural proposals as [YANG2003] already considered using the AS structure as a model. However, this model is only used to extract hints on how to build inter-domain relations and in which way to relay registrations, lookup requests, traffic… through the topology. No high level study is held, just some low scale considerations.

On the other hand, the other new Internet architectures usually run small laboratory implementations. Their main aim is checking the performance of their protocols in terms of overheads and delays. However, these implementations are, as explained in Chapter 4, insufficient to extract the behavior of a network the size of the Internet. Additionally, they are too much hardware dependent and become easily influenced by the concrete arbitrary scenario that is set to run (there is no 100% justification on how they choose the traffic characteristics, the links, the number of users…).

Hence, this dissertation is a novel approach for this sort of analysis and a brand new scientific contribution that takes profit of previous Internet research sources.

8.2- INTERNET STUDIES

Apart from those research studies that were the basis for this thesis and that were, therefore, already commented in Chapter 4 for the model, there are others interesting contributions that deserve, at least, a few lines. They can complement some aspects of this thesis for a better understanding of some complex issues.

As a starting point for a better knowledge of the Internet structure, [MEDINA2000] includes a brief introduction to the different topology generations algorithms and associates many concepts regarding the existence of power laws in most of the parameters that can be considered.

As a second recommended work, [LI2004] offers an at first glance description and quite complete visualization of the Internet topology.

OPCA [AGARWAL2003], as an ongoing effort of the BGP related studies, claims that the BGP routers are becoming insufficient to handle the inter-AS communication. They suggest an overlay-like parallel infrastructure that cooperates with the present one providing resiliency to the system.

[UHLIG2004] provides a brief on the different types of research related to model the traffic in the Internet and the obtained results. However, there are still some missing links. It includes some references to flow arrival processes that could be considered for a refinement of the statistical model.

Related to the DNS system, [LEE2003] pointed that it is a critical component of the architecture. It studies the interconnectivity of the different DNS root servers and tries to provide a solution on how to place them for a maximum efficiency. Somehow, it is remarkable the stress that it puts on the hierarchical problems of the structure. This shows that bottlenecks do not only apply to the naming and addressing architecture presented in this thesis.

In the DNS field as well, [BROWNLEE2001] proposes a measurement infrastructure to control the growth and operation of the DNS. As its main conclusion, it claims that DNS is able to scale thanks to caching, as it reduces the lookup requests in the top of the DNS hierarchy.

Keeping the communication state in the different ASes through the communication path is another sort of natural implicit caching that was commented in this thesis for a novel naming and addressing architecture.

Finally, [WESSELS2003] shows procedures on the operation of DNS root servers and their best usage for an optimal performance.

8.3- OTHER PROPOSALS FOR AN NGI ARCHITECTURE

This section includes information regarding next generation Internet architectures. The TurfNet was already presented in Chapter 3, due to its exemplarity for this thesis. As the TurfNet already found its in detail explanation, it is not included herein. Therefore, this section includes only other proposals that also aim to solve the problem of the Internet.

Hence, and even though no other candidate should be excluded, these are the main candidates to take profit from the scalability analysis presented in this dissertation.

HLP

[SUBRAMANIAN2004] suggests the necessity of a next generation inter-domain routing protocol able to make use of the AS structure knowledge. After claiming, as this dissertation does, that the current research is not taking enough profit of the research on BGP tables and the Internet topology, it suggests a hybrid link-state path-vector model for routing.

NIRA

[YANG2003] attempts to provide the user with the ability of domain-level route selection. This added flexibility claims a user control of its own traffic as a key point for a distributed Internet scenario. As a result, it would increment the economical competence of the providers. As

the previously commented new architecture, it also claims the necessity of assuring the scalability of the system. However, it does not quantify the feasibility of their system. One of the main contributions of this paper is a clear AS approach of the situation as the starting point for a next generation Internet.

FARA

Forwarding directive, Association and Rendezvous Architecture [CLARK2003], is based upon the decoupling of end-system names from network addresses that can be instantiated to add new features like mobility. Hence, as it is quite abstract, it could fit a hierarchical model. Its aim is to separate the identity space from the address space while providing mobility and adjusting to other architectural proposals.

TRIAD

[CHERITON2000] is another proposal for a new Internet architecture. The method that it suggests for resolving nowadays lack of connectivity in the Internet due to the large amount of middleboxes that break the architectural rules is the use of an explicit content layer. It follows an identification scheme that decouples from the addressing one, as pointed in Chapter 2. Therefore, it requires lookup mechanisms for resolution that could be evaluated for scalability following the procedures shown in this thesis.

Plutarch

[CROWCROFT2003] claims the necessity of allowing different sorts of networks cooperating and coexisting inside the Internet. The translation between domains, or *contexts*, is granted through what they call, *interstitial functions*, which allow the flow to traverse the *contexts*.

IPNL

[FRANCIS2001] proposes the NAT-extended IP architecture IPNL as the solution for the problematic of Internet. It also recommends separate domains that could fit the AS model herein presented. Even though independent from one another, these networks can reach interoperability through the use of NATs. However, while this thesis presented a multilayer non-limited model, IPNL can only provide two levels: namely a private and a public ones. Hence, it has strong limitations regarding its flexibility.

4+4

While [TURANYI2003] also has the same limitation of a public and a private domain, its improvement relies on the possibility of having middle layers. However, its naming architecture is still too topology dependent. The ideal one should perfectly dissociate naming from addressing.

9. CONCLUSION

The key contribution of this work is a study of the scalability of new naming and addressing schemes that fit the requirements and are the main achievement of any next generation Internet architecture.

The current Internet architecture is obsolete. It was created as a military resource. Nevertheless, it soon turned into a scientific purpose network. It allowed international cooperation and concurrent utilization of resources from multiple geographical locations.

However, the Internet expanded into a huge global network for business and private use as companies and users found in it the ideal platform for commercial use, communication, entertainment... in any imaginable sort of. While the purposes became radically different from the original ones, the architectural rules that applied to the network remained. These include: fixed networks (and, thus, lack of support for mobility both of networks and users), no decoupling of address and identity (addresses are used both to identify and to topologically locate), no intrinsic support for neither multihoming nor for multisourcing (as an identity can only possess an address), no support for dynamical composition of networks, inter-domain relations are difficult to break, obstacles when deploying new protocols and services (like the recent IMS and SIP), limited number of addresses (which is a barrier for the increasing number of users and networks)…

This situation and any other imaginable impediment have a common pattern. They all find their inner roots in naming and addressing intrinsic limitations that have been carried since the Internet was created.

Since some years, there is a worldwide agreement from the scientific community claiming that the Internet has to evolve into a new flexible network able to handle the new user demands. Furthermore, international organizations and governments support this initiative. One of these projects is [AMBIENT].

The all of them have the addressing related issues as their structural main requirement, even though each of them deals with it in its own way. Among the existing approaches the TurfNet architecture, partially funded by [AMBIENT], proposes a clear and absolute independency inbounds for each domain, for most of the architectural points. The global structure would take care of the inter-domain communications by providing the required upper-level support. In its basis, TurfNet has a hierarchical approach for the registration, resolution and location procedures. Other solutions, like [YANG2003], also show a predisposition to apply their methods in a hierarchical way.

Although the TurfNet was used as the exampling reference, the generic approach that was applied allows the extension of this work to any other architecture. Despite not being the TurfNet itself, all those Internet proposals that have in their basis a hierarchical scheme, find herein a scalability analysis for their purposes. Only some particularizations might be necessary. On the other hand, all those next generation architectures that apply significantly different strategies could also take profit of this work, as this possibility was already taken into consideration and is actually feasible. It would imply following the definitions and mathematical descriptions of the model up to the point in which they are instantiated to the TurfNet example. From there on, a parallel implementation would follow.

The second remarkable aspect of this thesis is the usage it made of the existing work. In order to perform the analysis, it was required an in detail study of the nowadays Internet to guess its evolution into a brand new network. These existing Internet studies like [SUBRAMANIAN2002] just tried to show the structure of the Internet to stimulate later discussions related to this field in the scientific community. This study overcame this limitation and gave a full sense to all this previous research. It went further by having the obtained studies not as mere results but as inputs for the project and opening a new research line. This enabled this dissertation to check the performance of naming and addressing architectures.

Chapter 6 presented a detailed experimental investigation of the hierarchical naming and addressing scheme and its possible bottlenecks. This chapter verified that the deployment of this hierarchical approach for the registration, resolution and location functionalities is feasible.

Hence, the TurfNet and any other hierarchical architecture could operate a new Internet while providing the brand new functionalities that the users require.

As no other next generation Internet proposal has checked the viability of their systems regarding scalability yet, the herein presented work could be a template for them. Then, this work would achieve its maximum level of contribution to the research community.

The next steps related to the scalability that should be carried out would include a small-scale implementation including most of the architectural features.

In the case of TurfNet, and now that the scalability is assured, this working proof has motivated an ongoing effort that will improve the mobility aspects of the architecture.

Additionally, by having validated the feasibility of the TurfNet in networks up to the size of the Internet, it becomes automatically demonstrated that this architecture could also run in smaller scenarios, as an on purpose application. In this sense, the TurfNet could find its immediate application (once the remaining items regarding mobility, security,… manage to perform resiliently) in lower scale cases: medium scale (e.g., a city; providing differentiated services for users, neighborhoods, buildings…) or small scale (e.g., a company; providing interoperation for networks, groups and employees inside it).

As a final statement in this conclusion, the changes in the Internet advance relentlessly. This thesis has proudly contributed to this aim with results, proposals and a new research line regarding scalability.

LIST OF FIGURES

REFERENCES

[ABLEY2003] J. Abley, B. Black, and V. Gill, "Goals for IPv6 site-multihoming architectures," RFC 3582, August 2003

[AGARWAL2003] OPCA: S. Agarwal, C. Chuah, and R. Katz, "Robust interdomain policy routing and traffic control", Proc. *IEEE Openarch*, New York, April 2003

[BALAKRISHNAN2004] H. Balakrishnan, K. Lakshminarayanan, S. Ratnasamy, S. Shenker, I. Stoica, and M. Walfish, "A Layered Naming Architecture for the Internet," Proc. *ACM SIGCOMM*, Portland, OR, USA, Aug 2004

[BRADEN2000] R. Braden, D. Clark, S. Shenker, and J. Wroclawski, "Developing a next-generation Internet architecture," Whitepaper, available at http://www.isi.edu/newarch/ DOCUMENTS/WhitePaper.ps, July 2000

[BROIDO2002] A. Broido, E. Nemeth, and KC Claffy, "Internet expansion, refinement and churn," *European Transactions on Telecommunications*, Vol. 13, No. 1, January-February 2002, pp. 33-51

[BROWNLEE2001] N. Brownlee, "DNS root/gTLD performance measurements," Proc. *15th USENIX Systems Administration Conference*, December 2001

[CHANG2002] H. Chang, R. Govindan, S. Jamin, S. Shenker, and W. Willinger, "On inferring AS-level connectivity from BGP routing tables," Proc. *IEEE INFOCOM*, 2002, pp. 618-627

[CLARK2002] D. Clark, J. Wroclawski, K. R. Sollins, and R. Braden, "Tussle in Cyberspace: defining tomorrow's Internet," Proc. *ACM SIGCOMM*, Pittsburgh, PA, USA, August 19-23, 2002, pp. 347-356

[CLARK2003] D. Clark, R. Braden, A. Falk, and V. Pingali, "FARA: reorganizing the addressing architecture," Proc. *ACM SIGCOMM Workshop on Future Directions in Network Architecture (FDNA)*, Karlsruhe, Germany, August 2003, pp. 313-321

[CROWCROFT2003] J. Crowcroft, S. Hand, R. Mortier, T. Roscoe, and A. Warfield, "Plutarch: an argument for network pluralism," Proc. *ACM SIGCOMM Workshop on Future Directions in Network Architecture (FDNA)*, Karlsruhe, Germany, August 2003, pp. 258-266

[CHERITON2000] D. R. Cheriton, and M. Gritter, "TRIAD: a scalable deployable NAT-based Internet architecture," Stanford Computer Science Technical Report, January 2000

[FRANCIS2001] P. Francis, and R. Gummadi, "IPNL: a NAT-extended Internet architecture," Proc. *ACM SIGCOMM*, San Diego, CA, USA, August

2001, pp. 69-80

[GAO2001] L. Gao, "On inferring Autonomous System Relationships in the Internet," *IEEE/ACM Transactions on Networking*, Vol. 9, No. 6, December 2001, pp. 733-745

[GE2001] Z. Ge, D.R. Figueiredo, S. Jaiswal, and L. Gao, "On the hierarchical structure of the logical Internet graph," Proc. *SPIE ITCOM*, Denver, Colorado, USA, August 2001

[JONSSON2003] Andreas Jonsson, Mats Folke, and Bengt Ahlgren, "The split naming/forwarding network architecture," Proc. *First Swedish National Computer Networking Workshop (SNCNW)*, Arlandastad, Sweden, September 8-10, 2003

[KAASHOECK2002] M. F. Kaashoek, and D. R. Karger, "Koorde: a simple degree-optimal distributed hash table". Proc. *2nd International Workshop on Peer-to-Peer Systems*, Berkeley, CA, USA, February 2003, pp. 98-107

[KRIOUKOV2004] D. Krioukov, K. Fall, and X. Yang, "Compact routing on Internet-like graphs," Proc. *IEEE INFOCOM*, March 2004

[LEE2003] T. Lee, B. Huffaker, M. Fomenkov, and KC. Claffy, "On the problem of optimization of DNS root servers' placement,", *Passive Measurement and Analysis Workshop*, 2003

[LI2004] L. Li, D. Alderson, W. Willinger, and J. Doyle, "A first-principles approach to understanding the Internet's router-level topology," *ACM SIGCOMM*, Portland, Oregon, USA, August 2004

[LI2005] D. Li, X. Lu, and J. Wu, "FISSIONE: a scalable constant degree and low congestion DHT scheme based on Kautz graphs," *IEEE INFOCOM*, 2005

[MAO2004] Z. Mao, D. Johnson, J. Rexford, J. Wang, and R. Katz, "Scalable and accurate identification of AS-level forwarding paths," *IEEE INFOCOM*, Vol. 23, No. 1, Hong Kong, China, March 2004, pp. 1606-1616

[MEDINA2000] A. Medina, I. Matta, and J. Byers, "On the origin of power laws in Internet topologies," *ACM SIGCOMM Computer Communication Review (CCR)*, April 2000, pp. 18-28

[MOSKOWITZ2005] R. Moskowitz, and P. Nikander, "Host Identity Protocol architecture," Work in Progress (draft-ietf-hip-arch-02.txt), January 2005

[PUJOL2005] J. Pujol, S. Schmid, L. Eggert, M. Brunner, and J. Quittek, "Scalability Analysis of the TurfNet Naming and Routing Architecture," Proc. *ACM MOBICOM, Workshop on Dynamic Interconnection of Networks*

(DIN 2005), Cologne, Germany, September 2, 2005.

[SCHMID2004] S. Schmid, L. Eggert, M. Brunner, and J. Quittek, "TurfNet: an architecture for dynamically composable networks," Proc. *First IFIP TC6 WG6.6 International Workshop on Autonomic Communication (WAC 2004)*, Berlin, Germany, October 18-19, 2004

[SCHMID2005] S. Schmid, L. Eggert, M. Brunner, and J. Quittek, "Towards Autonomous Network Domains," Proc. *8th IEEE Global Internet Symposium*, Miami, FL, USA, March 17-18, 2005

[SUBRAMANIAN2002] L. Subramanian, S. Agarwal, J. Rexford, and R.H. Katz, "Characterizing the Internet hierarchy from multiple vantage points," Proc. *IEEE INFOCOM*, NY, USA, June 2002, pp. 618-627

[SUBRAMANIAN2004] L. Subramanian, et al., "Towards a next generation inter-domain routing protocol," *Third Workshop on Hot Topics in Networks*, November 2004

[STOICA2003] I. Stoica, R. Morris, D. Karger, M. Frans Kaashoek, and H. Balakrishnan, "Chord: a scalable peer-to-peer lookup service for Internet applications," *IEEE/ACM Transactions on Networking*, Vol. 11, No. 1, February 2003, pp. 17-32

[TURANYI2003] Z. Turanyi, A. Valko, and A. Campbell, "4+4: an architecture for evolving the Internet address space back towards transparency," *ACM SIGCOMM Computer Communication Review*, Vol. 33, No. 5, October 2003, pp. 43-54

[UHLIG2002] S. Uhlig and O. Bonaventure, "Implications of interdomain traffic on traffic engineering," *European Transactions on Telecommunications*, Special issue on traffic engineering, 2002.

[UHLIG2004] S. Uhlig, "A review of scaling behaviors in Internet traffic," *CIRM Power-laws Workshop*, Luminy, March 2004

[WALFISH2004] M. Walfish, J. Stribling, M. Krohn, H. Balakrishnan, R. Morris, and S. Shenker, "Middleboxes no longer considered harmful," Proc. *USENIX Symposium on Operating Systems Design & Implementation (OSDI)*, San Francisco, CA, USA, December 2004, pp. 215-230

[WESSELS2003] D. Wessels, M. Fomenkov, "Wow, that's a lot of packets," *4th Annual Passive and Active Measurement* Workshop, 2003

[YANG2003] X. Yang, "NIRA: a new Internet routing architecture," *ACM SIGCOMM Workshop on Future Directions in Network Architecture (FDNA)*, Karlsruhe, Germany, August 2003, pp. 301-312

[ZEITOUN2004] A. Zeitoun, C.N. Chuah, S. Bhattacharyya, and C. Diot, "An AS-Level

Study of Internet Path Delay Characteristics," Proc. *IEEE GLOBECOM*, November 2004, Dallas, USA.

[AMBIENT] "Ambient Networks," http://www.ambient-networks.org

[CAIDA] "Cooperative Association for Internet Data Analysis," http://www.caida.org

[CIA] "CIA World Factbook," http://www.cia.gov/cia/publications/factbook

[GOOGLE] "Google," http://www.google.com

[ISC] "Internet Systems Consortium," http://www.isc.org

[MATHWORKS] "The MathWorks, Inc.," http://www.mathworks.com

[MYSQL] "MySQL," http://www.myswl.com

[NLANR] "National Laboratory for Applied Network Research" http://www.nlanr.net

APPENDIX

This appendix shows, as an example, one of the releases for the code in MatLab language that was implemented to model the system and obtain the final outputs. It follows the structure pointed in Chapter 5. This version is quite significant as the main programmed features are included herein.

From the original code, some parts have been rearranged to provide a user-friendly view at first glance. Others have been removed due to its lack of specificity.

Other releases include:

- A generic solution that uses all the existing links.
- A version to count the minimum number of hops for resolution.
- An improvement to traverse all the peers. This one is specially processing time consuming.
- A version that sends to one provider per level.
- Sending to one provider with preference to the second level (to save operations in the first one, as explained in the improvements of Chapter 6).
- Older versions than did not distinguish among providers.

It is quite clear that there is a wide range of combinations and possibilities. The programming style provides a means of flexibility to the scheme that is validated by just some slight changes in the code.

CODE

This code was used to distribute the lookup requests for resolution to just one provider if many are available. The choice is made randomly so that no provider is arbitrarily overloaded while others get a low rate of communication establishments. This approach statistically

distributes the requests in the fairest way. Otherwise, those ASes with lower AS numbers would appear as the most used ones.

In addition to this, this code enables detection of duplicates for lookup requests that reach a certain AS from the same original AS level 5 one through different paths (as seen in Chapter 6).

MAIN.m

```
function MAIN

% INITIALIZATION

% Peering at level 1

fprintf(1,'%s\n','Enter peering mode at Level 1:');
fprintf(1,'%s\n','(1)Access to a centralized database (or upper AS)');
fprintf(1,'%s\n','(2)Use of peering');
global peer_1_mode;
peer_1_mode=input('(1=Default)->');
if peer_1_mode~=1 & peer_1_mode~=2 & peer_1_mode~=3
   peer_1_mode=1;
end

% Sort of peering used for other levels

fprintf(1,'%s\n','Enter peering to be used for the rest of levels:');
fprintf(1,'%s\n','(1)Peering when Registering');
fprintf(1,'%s\n','(2)Peering when Looking Up');
fprintf(1,'%s\n','(3)No peering');
global peer_x_mode;
peer_x_mode=input('(1=Default)->');
if peer_x_mode~=1 & peer_x_mode~=2 & peer_x_mode~=3
   peer_x_mode=1;
end

% Ask for total population

population=input('Enter population in the system:')

% Ask for requests/(second*user)

req_user=input('Enter requests per second and user:')


% Reading the ashierarchy file. It contains a list of ASes and their
```

```
% hierarchical level form 1 to 5, where those in the 5th level are the
% customers

global Hierarchy;
Hierarchy=load('ashierarchy.txt');

% Obtain the number of ASes per level

num_AS1=0;num_AS2=0;num_AS3=0;num_AS4=0;num_AS5=0;
for num_AS=1:size(Hierarchy,1)
   switch Hierarchy(num_AS,2)
      case 1
         num_AS1=num_AS1+1;
      case 2
         num_AS2=num_AS2+1;
      case 3
         num_AS3=num_AS3+1;
      case 4
         num_AS4=num_AS4+1;
      case 5
         num_AS5=num_AS5+1;
   end
end
num_AS
num_AS1
num_AS2
num_AS3
num_AS4
num_AS5

% Get peer scope per level. Scope[X]=scope for level X.
% -1 means no limit

Scope=load('scope.txt');

% Assign requests/second and population to each level 5 AS
% Applying an evenly distribution

pop_turf=population/num_AS5;
req_turf=req_user*pop_turf;

% Load the inter AS relations

Relations=load('as.relation.txt');

% Create Provider sparse matrix
% Provider(i,j)->AS level for i's providers (=-1 if P2P)
```

```
% Provider(i,i)->AS level for i

Provider=sparse(0,0);
for i=1:size(Relations,1)
   AS1=Relations(i,1);
   h_AS1=Hierarchy(find(Hierarchy==AS1,1),2);
   AS2=Relations(i,2);
   h_AS2=Hierarchy(find(Hierarchy==AS2,1),2);
   if(Relations(i,3)==1)
      Provider(AS2,AS1)=h_AS1;
   else
      Provider(AS2,AS1)=-1;
      Provider(AS1,AS2)=-1;
   end
end
clear AS2 h_AS1 h_AS2
for i=1:size(Hierarchy,1)
   AS1=Hierarchy(i,1);
   h=Hierarchy(i,2);
   Provider(AS1,AS1)=h;
end
clear Relations h AS1

% To compute the population under each AS in levels 4 to 1

global Pop_under
Pop_under=sparse(0,0);
global Peer_pop_gain
Peer_pop_gain=sparse(0,0);

% Load probabilities depending on AS hops and arrange them into a vector

prob=load('prob.hops.txt');
global REQ
REQ=prob*req_turf;

% Initialization of working variables

dim=size(Provider,1)
global turflookup0;
global turflookup1;
global turflookup2;
global turflookup3;
global turflookup4;
global BLACK_LIST;
global BLACK_PEER_LIST;
turflookup0=zeros(1,2);
```

```
turflookup1=zeros(dim,5);
turflookup2=zeros(dim,5);
turflookup3=zeros(dim,5);
turflookup4=zeros(dim,5);
global entries0;
global entries1;
global entries2;
global entries3;
global entries4;
entries0=zeros(1,1);
entries1=zeros(dim,2);
entries2=zeros(dim,2);
entries3=zeros(dim,2);
entries4=zeros(dim,2);
ROW=0;
global flag_level0;
```

% REGISTRATION OF NODES

% This first pass allows registration and getting probabilities for forwarding of lookups

```
for i=1:dim
   if Provider(i,i)==5
      flag_level0=0;
      BLACK_LIST=0;
      for k=1:4
         clear ROW
         ROW=find(Provider(i,:)==k); % Find all level k providers
         dim2=size(ROW,2);
         if dim2~=0
            for j=1:dim2
               % Call registration function
               scoping(k, ROW(j), Provider, i);
            end
         end
      end
      if peer_x_mode~=3
         % The BLACK LIST contains all those ASes that have been traversed
         % Make use of the BLACK LIST to access peers for registration
         BLACK_PEER_LIST=0;
         dim2=size(BLACK_LIST,2);
         for j=1:dim2
            % Those ASes contained in the BLACK PEER LIST have been already used
            % Check if in the BLACK PEER LIST; if not, go on
            if size(find(BLACK_PEER_LIST==BLACK_LIST(j)),2)==0
               own=Hierarchy(find(Hierarchy(:,1)==BLACK_LIST(j),1),2);
               if Scope(own)>0
                  % Find all the peers for the element j of the BLACK LIST
```

```
                ROW=find(Provider(BLACK_LIST(j),:)==-1);
                dim3=size(ROW,2);
                if dim3~=0
                   for l=1:dim3
                      % Reset the scope for each pass in the loop and discount one hop
                      Remaining_scope=Scope;
                      Remaining_scope(own)=Remaining_scope(own)-1;
                      % Call peering registration function
                      peer_scoping(Hierarchy(find(Hierarchy(:,1)==ROW(l),1),2), ROW(l),
Provider, Remaining_scope, i);
                   end
                end
             end
          end
       end
    end
    i % Show progress
end

Reachable_pop=or(Pop_under,Peer_pop_gain);
global population_under
population_under=sum(Pop_under,2);
population_under=full(population_under);
dlmwrite('population_under.txt', population_under, 'delimiter', ' ', 'newline','pc');
global gain_by_peering
gain_by_peering=sum(Peer_pop_gain,2);
gain_by_peering=full(gain_by_peering);
dlmwrite('gain_by_peering.txt', gain_by_peering, 'delimiter', ' ', 'newline','pc');
```

% LOOKUP AND RESOLUTION

% In this second pass the lookup requests are distributed and statistically resolved

```
for i=1:dim
   if Provider(i,i)==5
      flag_level0=0;
      BLACK_LIST=0;
      flag_out=0;
      for k=1:4
         clear ROW
         if flag_out==0
            % Find all level k providers of i
            ROW=find(Provider(i,:)==k);
            dim2=size(ROW,2);
            if dim2~=0
                  % Call the lookup resolution process
                  TurfLookUp(k, ROW(unidrnd(dim2)), Provider, req_turf, pop_turf, i, 1);
```

```
                flag_out=1;
            end
        end
    end
    if peer_x_mode~=3
        % Make use of the BLACK LIST to add peer queries
        BLACK_PEER_LIST=0;
        dim2=size(BLACK_LIST,2);
        for j=1:dim2
            % Those ASes contained in the BLACK PEER LIST have been already used
            % Check if in the BLACK PEER LIST; if not, go on
            if size(find(BLACK_PEER_LIST==BLACK_LIST(j)),2)==0
                own=Hierarchy(find(Hierarchy(:,1)==BLACK_LIST(j),1),2);
                if Scope(own)>0
                    % Find all the peers for the element j of the BLACK LIST
                    ROW=find(Provider(BLACK_LIST(j),:)==-1);
                    dim3=size(ROW,2);
                    if dim3~=0
                        for l=1:dim3
                            % Reset the scope for each pass in the loop and discount one hop
                            Remaining_scope=Scope;
                            Remaining_scope(own)=Remaining_scope(own)-1;
                            % Call the peer lookup resolution process, in case that it is required
                            peer_TurfLookUp(Hierarchy(find(Hierarchy(:,1)==ROW(l),1),2), ROW(l),
Provider, req_turf, pop_turf, Remaining_scope, i);
                        end
                    end
                end
            end
        end
    end
    i % Show progress
end
clear Hierarchy
```

% COLLECTING DATA, PERFORMING STATISTICS AND DRAWING GRAPHS

% This section has been removed as it had no AS specific functionalities
% It uses the statistical and drawing tools from MatLab to format the output data

TurfLookUp.m (lookup process)

```
function TurfLookUp(own, i, Provider, req_turf, pop_turf, original, jumps)

global population_under
global gain_by_peering
global REQ

% Add load in requests per second (some of them will be discarded)

switch own
   case 1
      global turflookup1;
      turflookup1(i,1)=turflookup1(i,1)+ req_turf;
   case 2
      global turflookup2;
      turflookup2(i,1)=turflookup2(i,1)+ req_turf;
   case 3
      global turflookup3;
      turflookup3(i,1)=turflookup3(i,1)+ req_turf;
   case 4
      global turflookup4;
      turflookup4(i,1)=turflookup4(i,1)+ req_turf;
end

% Check if in the BLACK LIST (requests already arrived, so that they do not have to be resent)

global BLACK_LIST;
if size(find(BLACK_LIST==i),2)~=0
   return;
end

% Add itself to the BLACK LIST

BLACK_LIST(size(BLACK_LIST,2)+1)=i;

% Requests which are attended (locally resolved or forwarded)
% Statistical calculus of the proportion of requests that is forwarded
forwarded_req=REQ(jumps)*((population_under(MAX)+gain_by_peering(MAX))-
population_under(i,1)-gain_by_peering(i,1))/(population_under(MAX)+gain_by_peering(MAX)-
population_under(i,1));
switch own
   case 1
      % For level 1, use of a deterministic approach
```

```
      forwarded_req=req_turf*((population_under(MAX)+gain_by_peering(MAX))-
population_under(i,1)-
gain_by_peering(i,1))/(population_under(MAX)+gain_by_peering(MAX));
      turflookup1(i,2)=turflookup1(i,2)+ req_turf;
      turflookup1(i,5)=turflookup1(i,5)+ forwarded_req;
   case 2
      turflookup2(i,2)=turflookup2(i,2)+ req_turf;
      turflookup2(i,5)=turflookup2(i,5)+ forwarded_req;
   case 3
      turflookup3(i,2)=turflookup3(i,2)+ req_turf;
      turflookup3(i,5)=turflookup3(i,5)+ forwarded_req;
   case 4
      turflookup4(i,2)=turflookup4(i,2)+ req_turf;
      turflookup4(i,5)=turflookup4(i,5)+ forwarded_req;
end

% Forward requests
if own==1
    global peer_1_mode;
    if peer_1_mode==1
       % This flag allows detection of duplicates for the 0 level
       global flag_level0;
       % Special function in case of 0 level
       Turflookup0(forwarded_req, pop_turf, jumps+1);
       if flag_level0==0
          flag_level0=1;
       end
    end
   return;
end
ROW=0;
if size(population_under,1)<i
   population_under(i)=0;
end
if size(gain_by_peering,1)<i
   gain_by_peering(i)=0;
end
flag_out=0;
for k=1:(own-1)
   clear ROW
   if flag_out==0
      % Finf the level k providers of i
      ROW=find(Provider(i,:)==k);
      dim2=size(ROW,2);
      if dim2~=0
            % Recursive usage of function
```

```
            TurfLookUp(k, ROW(unidrnd(dim2)), Provider, forwarded_req, pop_turf, original,
jumps+1);
            flag_out=1;
        end
    end
end
```

TurfLookUp0.m (lookup process)

```
function Turflookup0(req, pop, jumps)

% This function handles the special case of a 0 level

global turflookup0;
global entries0;
global flag_level0;

% Add requests arriving
turflookup0(1,1)=turflookup0(1,1)+req;
% Only once per request; to avoid duplicates
if flag_level0==0
    % Add requests to be solved
    turflookup0(1,2)=turflookup0(1,2)+req;
end
```

peer_TurfLookUp.m (lookup process)

```
function peer_TurfLookUp(own, i, Provider, req_turf, pop_turf, Scope, original)

global population_under
global gain_by_peering

% Check the settings for peering
global peer_1_mode;
if own==1
    if peer_1_mode==1
        return;
    else
        global turflookup1;
        turflookup1(i,3)=turflookup1(i,3)+ req_turf;
    end
end

global Hierarchy;
global peer_x_mode;
if peer_x_mode==2
```

```
   % Add load being received to the current AS
   switch own
      case 2
         global turflookup2;
         turflookup2(i,3)=turflookup2(i,3)+ req_turf;
      case 3
         global turflookup3;
         turflookup3(i,3)=turflookup3(i,3)+ req_turf;
      case 4
         global turflookup4;
         turflookup4(i,3)=turflookup4(i,3)+ req_turf;
   end
end

% Check if in the BLACK PEER LIST: then exit (requests aleady arrived
% here as a result of peering)

global BLACK_PEER_LIST;
if size(find(BLACK_PEER_LIST==i),2)~=0
   return;
end

% Add itself to the BLACK PEER LIST

BLACK_PEER_LIST(size(BLACK_PEER_LIST,2)+1)=i;

% Check if in the BLACK LIST: then exit (requests already arrived here
% through a hierarchical path)

global BLACK_LIST;
if size(find(BLACK_LIST==i),2)~=0
   return;
end

% Add lookup requests

if peer_x_mode==2
   % Add load to the queries being forwarded to the peers
   switch own
      case 1
        turflookup1(i,4)=turflookup1(i,4)+ req_turf;
      case 2
         turflookup2(i,4)=turflookup2(i,4)+ req_turf;
      case 3
         turflookup3(i,4)=turflookup3(i,4)+ req_turf;
      case 4
         turflookup4(i,4)=turflookup4(i,4)+ req_turf;
```

```
   end
end

% Forward to the peers
req_turf=req_turf*((population_under(MAX)+gain_by_peering(MAX))-population_under(i)-
gain_by_peering(i))/(population_under(MAX)+gain_by_peering(MAX));
if own~=1
   ROW=find(Provider(i,:)==-1);
   dim=size(ROW,2);
   if dim~=0
      for j=1:dim
          % Discount one hop in the scope and send if still scope remaining
         Remaining_scope=Scope;
         if Remaining_scope(own)>0
            Remaining_scope(own)=Remaining_scope(own)-1;
            % Recursive usage of function
            peer_TurfLookUp(Hierarchy(find(Hierarchy(:,1)==ROW(j),1),2), ROW(j), Provider,
req_turf, pop_turf, Remaining_scope, original);
         end
      end
   end
else
   ROW=find(Provider(i,:)==-1);
   dim=size(ROW,2);
   if dim~=0
      for j=1:dim
         if Hierarchy(find(Hierarchy(:,1)==ROW(j),1),2)==1
            % Discount one hop in the scope and send if still scope remaining
            Remaining_scope=Scope;
            if Remaining_scope(own)>0
               Remaining_scope(own)=Remaining_scope(own)-1;
              % Recursive usage of function
               peer_TurfLookUp(1, ROW(j), Provider, req_turf, pop_turf, Remaining_scope,
original);
            end
         end
      end
   end
end
```

Scoping.m (registration process)

```
function Scoping(own, i, Provider, original)

% Check if in the BLACK LIST (requests already arrived, so that they do not have to be resent)

global BLACK_LIST;
if size(find(BLACK_LIST==i),2)~=0
   return;
end

% Add itself to the BLACK LIST

BLACK_LIST(size(BLACK_LIST,2)+1)=i;


% Add entries for the population in the AS5

global Pop_under
Pop_under(i,original)=1;
switch own
   case 1
      global entries1;
      entries1(i,1)=entries1(i,1)+pop_turf;
   case 2
      global entries2;
      entries2(i,1)=entries2(i,1)+pop_turf;
   case 3
      global entries3;
      entries3(i,1)=entries3(i,1)+pop_turf;
   case 4
      global entries4;
      entries4(i,1)=entries4(i,1)+pop_turf;
end

% Forward
if own==1
   return;
end
ROW=0;
for k=1:(own-1)
   clear ROW
   % Find the level k providers of i
   ROW=find(Provider(i,:)==k);
   dim2=size(ROW,2);
   if dim2~=0
```

```
      for j=1:dim2
         % Recursive usage of function
         Scoping(k, ROW(j), Provider, original);
      end
   end
end
```

peer_Scoping (registration process)

```
function peer_Scoping(own, i, Provider, Scope, original)

global Hierarchy;
global peer_x_mode;

% Check if in the BLACK PEER LIST: then exit (requests aleady arrived
% here as a result of peering)

global BLACK_PEER_LIST;
if size(find(BLACK_PEER_LIST==i),2)~=0
   return;
end

% Add itself to the BLACK PEER LIST

BLACK_PEER_LIST(size(BLACK_PEER_LIST,2)+1)=i;

% Check if in the BLACK LIST: then exit (requests already arrived here
% through a hierarchical path)

global BLACK_LIST;
if size(find(BLACK_LIST==i),2)~=0
   return;
end

% Register the entries

global Peer_pop_gain
Peer_pop_gain(i,original)=1;

if peer_x_mode==1
   % must increase the tables size
   switch own
      case 1
         global entries1;
         entries1(i,2)=entries1(i,2)+pop_turf;
      case 2
         global entries2;
```

```
        entries2(i,2)=entries2(i,2)+pop_turf;
      case 3
        global entries3;
        entries3(i,2)=entries3(i,2)+pop_turf;
      case 4
        global entries4;
        entries4(i,2)=entries4(i,2)+pop_turf;
    end
end

% Forward to the peers
if own~=1
    % Find the peers
    ROW=find(Provider(i,:)==-1);
    dim=size(ROW,2);
    if dim~=0
      for j=1:dim
        % Discount one hop in the scope and send if still scope remaining
        Remaining_scope=Scope;
        if Remaining_scope(own)>0
          Remaining_scope(own)=Remaining_scope(own)-1;
          % Recursive usage of function
          peer_Scoping(Hierarchy(find(Hierarchy(:,1)==ROW(j),1),2), ROW(j), Provider,
Remaining_scope, original);
        end
      end
    end
else
      % Find the peers
      ROW=find(Provider(i,:)==-1);
      dim=size(ROW,2);
      if dim~=0
        for j=1:dim
          if Hierarchy(find(Hierarchy(:,1)==ROW(j),1),2)==1
            % Discount one hop in the scope and send if still scope remaining
            Remaining_scope=Scope;
            if Remaining_scope(own)>0
              Remaining_scope(own)=Remaining_scope(own)-1;
              % Recursive usage of function
              peer_Scoping(1, ROW(j), Provider, Remaining_scope, original);
            end
          end
        end
      end
end
```