Bootstrapping the Internet of the Future

Mohammad Shafahi mohammad.shafahi@os3.nl

Supervisors:
Rudolf Strijkers rudolf@strijkers.eu
M.X.Makkes Makkes m.x.makkes@kr85.org

System and Network Engineering
University of Amsterdam
Amsterdam, Netherlands

February 20, 2012

Abstract

Virtual Internet makes use of cloud providers to create application specific overlay networks under control of a single user. Application-specific optimization can be reached by strategically placing virtual machines at cloud providers based on measurements done on the Internet. In this research we have studied the feasibility and viability of constructing such a network by suggesting an architecture for the Virtual Internet and also showing that the Virtual Internet can act as a feasibility testbed and even improve some application specific parameters of the Internet.We do so by studding the possibilities of reducing latency as an application specific need.

Contents

List of Figures

1 Introduction

The design of the Internet did not account for network evolution. But since its existence, the Internet needed amendments to address problems or new protocols for new uses. The explosive increase of network devices and their increasing mobility currently threatens the stability of the Internet. Solutions to these problems, larger address space and keeping track of address locations, require changes to the network layer protocol.

Changes to the network layer show a fundamental problem in the Internet architecture: the convergence layer is fixed. Though, technically feasible, due to the vast variety of Internet service providers and their distribution between geographical and political borders it would be a complex managerial task to make jointly agreements on architectural changes to the Internet. A solution to this problem could be designing a architecture for the Internet that accounts network evolution.To do so first we must design an architecture that accounts network evolution and later on do the difficult task of having jointly agreements once and for ever.Although effort is put into doing this task by the scientific community (See [3, 4, 2]) one can always argue that how would we elect such an architecture and how can we know that an elected solution is the best solutions.In fact there is no way of proving this, so we will end up with the problem of convincing the Internet stakeholders that a specific so called evolution-accounted network architecture is the best architectural solution.

This issue has caused the evolution of the Internet to come to a dead end.Although simulations,testbeds and overlays are ways of overcoming this issue, usage of such environments has it's own limitations an alternative to overcome the limitations of these solutions is virtualization (See [6]). Different methods of virtualization have been suggested such as [6, 7] but usage of such solutions is limited to the scope of there community.

Nowadays that computing power is available everywhere and at anytime it is more and more feasible for anyone that wants to boot there own Virtual Internet to do so. Because of the fast growth of cloud providers both in term of growth and global distribution cloud-based computing is now the most suitable computing power to build such a Virtual Internet (See figure 1).

Figure 1: Examples of Cloud Providers

Although research has been proposed to build network experiments on the Amazon EC2 (See [1]) no research has been done on the Inter-Cloud and [1] has also been halted [1] Here we will suggest an architecture for building such a Virtual Internet in the Inter-Cloud and we will provide proof that such a Virtual Internet can provide more efficient routes between nodes than the current Internet.

1.1 Applications of the Virtual Internet

The Virtual Internet could have an important role in network evolution by providing the ability of change the convergence layer of the Internet by not just preforming as a virtual testbed but also being a cheap, easy accessible and trivially constructible one.Other than that because of the ability of having programmatic control over router placement,easily being scalable and simplicity in management features of the Virtual Internet it can also have applications such as:

Application Specific Optimization Each application on the Internet may or may note be sensitive to some parameters of it's connection.One can be sensitive on bandwidth and other can be sensitive on latency.Consider a situation that an application would want to stream a football game to it's end user in this case although the user would like to see the game exactly when it is happening but the application is mainly sensitive on bandwidth to give a better quality of the game and is less sensitive on latency.

Now think of a more complex scenario like an online surgery in this case although the image transmission from the surgery room is both sensitive in latency and bandwidth, the control signals sent by the surgeon to the surgery room is highly sensitive on latency because a late move may cause a change in the surgery situation and cause the

[1]Since November 2007 based on the website

surgery knife to cut the wrong place but it is not sensitive on bandwidth.Such real-time applications can use the ability that the virtual Internet provides them by giving them the ability to choose which path they want to take to gain their requirements instead of constructing their own network.Although this would only be applicable if such paths actually exist in the Internet and it is only a matter of choosing to go through that path(We will come to this later on)

Online Changes We can not change the convergence layer of the Internet but what we can do is to replace end points. The Virtual Internet provides the ability to make changes in the convergence layer by adding end points that would act as Virtual Routers. Other than that because these Virtual Routers are only images on different cloud providers, it is possible to dynamically add, remove or replace these routers based on real time network analysis.

1.2 Research Question

As mentioned before, the applications of the Virtual Internet are accomplished by making end points that we have control over act as Virtual Routers. These Virtual Routers enforce the Internet to choose a different path that is more optimum than the original one for our specific application to get from the source to the final destination. The main goal of this research is to see if Virtual Routers could actually achieve this task. To do so we can propose two questions and a hypothesis for each one.

Question 1 What are the necessary architectural components to boot a virtual Internet on clouds?

Hypothesis 1 *The necessary architectural components to boot a Virtual Internet are Virtual Routers, Virtual Links,The Virtual Internet Operating System and for sake of simplicity a central controller (A Decentralized solution is also possible)*

Question 2 Can a virtual Internet provide more efficient routes between nodes than the current Internet does?

Hypothesis 2 *A Virtual Internet can provide more efficient routes between nodes than the current Internet does.*

2 Architecture

The minimum components needed for a network are the routers and the links between them .Although other components such as the hosts (as the end points) and the gateways (as entry and exit points) are also necessary parts of getting the network to function, the network itself just consists of routers and links that can also act as hosts or gateways.Now the question is that are these components enough to build the Virtual Internet or are there more necessarily components as proposed in hypothesis 1 ? Another important thing that has to be considered is that how would these components look like in the Virtual Internet?

Fist of all in the Internet, routers and links are actually physical components but in the Virtual Internet we don't have physical devices so we should have Virtual Routers and Virtual Links instead. Other than that these Virtual Routers and Virtual Links should either be created and managed manually by the scientist (user) or automated with a software. Because we are interested in dynamically managing the servers as a feature of the Virtual Internet we should have a specific software (or set of softwares) to do so we call this the Virtual Internet Operating System.We will cover the Virtual Internet Operating System in details in section 3 but for the architectural point of view the Virtual Internet Operating System could do its management tasks in two ways either a distributed where the decisions are made in a distributed way like an election or in a centralized way where one central controller is in charge of the decisions making or a hybrid of the two (see chapter two of [8]) .

Although a distributed model may be better for reasons such the fact that the controller could actually be a failure point in the Virtual Internet in terms of management, because it is not a bottle neck in terms of performance in the Virtual Internet for the sake of simplicity we will consider the controller to be a centric one(see chapter eight of [8]). To have a better view of the architecture of the Virtual Internet see figure 2. In the following section we will cover the details of each of the 3 components of the architecture.

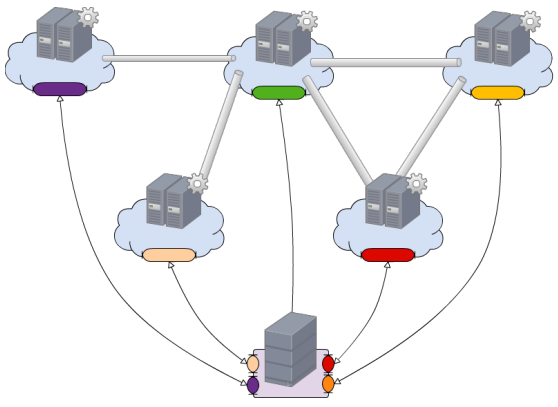

Figure 2: The Virtual Internet Operating System

2.1 Virtual Routers

As mentioned before, the Virtual Routers are one of the main components of the Virtual Internet. The Virtual Routers are virtual machines that have been instantiated in the Inter-Cloud.It is also possible that one virtual machine can act as multiple Virtual Routers by using virtualization techniques for virtualizing networks such as the methods shown in [7]. For simplicity in our study we will consider each virtual machine to be only one Virtual Router.

The Virtual Routers could be images of actual physical routers operating systems (like Cisco IOS Images) or could be images of operating systems (such as Linux) with customized routing softwares. What we select as image for the virtual machine depends on the application we are willing to have for the Virtual Internet. It is good to mention that there is no need for all the Virtual Routers to have the same image nor is there a need to have all the Virtual Routers as virtual machines inside the single cloud provider.

These Virtual Routers are connected to each other via virtual links (that we will discuss later on about links) but they are still not capable of routing traffic. Before being able to start routing, they have to first find the paths available to other routers and build their routing tables with a routing protocol. There are no limitations in choosing what routing protocols we would like to use or implement according to our needs. Even in some cases, there could be other ways of routing other than using routing tables for example having flow tables in the open flow protocol (See [5]). The protocols if needed can be built into the image of the virtual machine before deployment or can be deployed after instantiating the virtual machine.

As we will cover in section 3 in most cases the scientist (user)will only see an abstraction of the routers and will not be involved in instantiating the virtual machines and will just request virtual machines with specific configurations. Sometimes routers may act also as gateways (entrance and exit points) for hosts of the network but not all routers are required to act as entrance points but some should do so for the network to actually flow traffic. In some cases the Virtual Routers can also act as end points for example in cases of running a dynamic routing protocol (like OSPF) or in case of acting as a traffic generator.

2.2 Virtual Links

As mentioned before, Virtual Links are another of the main components of the Virtual Internet. Virtual links in the Virtual Internet could be considered as tunnels that connect the virtual machines to each other. To clarify our definition of tunnels, tunnels could be a broad band of encapsulation protocols using additional IP headers from light-weight protocols such as IPV4-in-IPV4, IPV4-in-IPV6 to complicated tunneling protocols with encryption payloads based on application.

These Virtual Links could pass thorough multiple physical links and routers and these physical paths may vary in time based on the routing decisions made by the Internet. For the Virtual Internet to use optimum paths, we should measure and keep track of parameters of the link such as latency and bandwidth. These measurements could either

be done in real time, periodically or based on specific demand.

The controller is responsible for building the links between the routers based on parameters delivered to it by the routers that have been collecting the needed parameters. The controller will build these links based on different layers of abstractions, constructed by the Virtual Internet Operating System. Creating Virtual Links will not be an issue for the scientist (user) because the Virtual Internet Operating System will create an abstraction for the scientist (user) to make the process of creating such a link a trivial task. We will discuss these layers of abstraction in more details in section 3.

2.3 The Controller

The most important component of the Virtual Internet is the controller. The controller could be either a centralized independent machine in the Internet or could be a virtual machine in Inter-Cloud. It could also be the case that the controller would be decentralized in term of services (multiple virtual virtual machines responsible of different tasks of the controller) or in terms of distributed decision making such as election. Another alternative for the placement of the controller could be that the controller is implemented inside 1 or multiple Virtual Routers. Although as we suggested the controller may not exist and its tasks could be decentralized, the controller's tasks play an important role in the Virtual Internet. For the sake of simplicity we will consider the controller as being centralized and we will concentrate on the tasks of the controller in detail.

User Interaction One of the main tasks of the controller is to interact with the scientist (user). Based on the scientist (user) request, the controller will be responsible of constructing the network. The controller interacts with the scientist (user) through multiple layers of abstraction constructed by the Virtual Internet Operating System. The user gives its instructions to the controller through the controller's interface that could be in the forms of command line interface, configuration files and graphical user interface. In some cases the user of the controller could be a program written by the user that instructs the controller by changing the configuration files or thorough the command line interfaces.

Instantiating Virtual Routers and Virtual Links The controller is responsible of instantiating virtual routers and virtual links in the network. It does so thorough different cloud provider APIs interfaces and thorough SSH access to virtual machines. It is important to know that the controller does these tasks by the Virtual Internet Operating System. After creating Virtual Internet components, the controller also can deploy specific configurations and applications to routers based on the users request. Because the controller is responsible of instantiating virtual routers and virtual links, it always has a global knowledge of existing routers and paths in the network.

Maintaining Virtual Routers and Virtual Links Because of its global knowledge of the network and based on the measurement data that it receives from the Virtual Routers, it can decide on removing or creating Virtual Links and Virtual Routers. It can also decide on moving a virtual router from one location to another and to do so it will reconfigure the links based on the new placement of the Virtual Router.

3 Virtual Internet Operating System

For the scientist (user) to have minimum worries about how he or she will implement a Virtual Internet and concentrate on the specific work he or she wants to do, we should have an abstraction for the Virtual Internet to do so we will need an application or a set of applications to create this abstraction and to manage each abstraction layer. We will call this application or set of applications the Virtual Internet Operating System. Figure 3 shows the layering of these abstractions and with the name of each services provided in the layer. To have a better feeling of how this abstraction is done by the Virtual Internet Operating System see figure 5.

Figure 5, shows an example of mapping abstraction layers by the Virtual Internet Operating System. The Virtual Internet Operating System does this kind of abstraction and mapping for each layer.

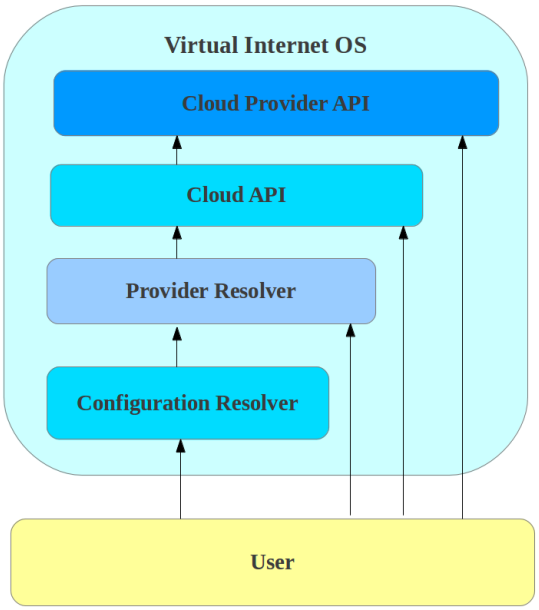

Figure 3: The Virtual Internet Operating System

3.1 Configuration Resolver

The Configuration Resolver has the task of resolving configurations(or as we will call router signaturess) provided by the scientist (user) to router names. Based on the router signatures requested by the scientist (user) the Configuration Resolver returns a router name to the scientist (user). If the scientist (user) has requested a router signatures that no router name with that router signatures exists, Configuration Resolver creates one and returns that. The Configuration Resolver can also create multiple Routers with the same router signatures based on the request of the scientist (user) or based on the internal management decisions.

Figure 4 shows the resolving done by the Configuration Resolver. The router signaturess could vary depending on application but in figure 4, they just consist of amount of RAM and located continent.

$$8GBRam, SouthAmerica \longrightarrow Router5$$

$$2GBRam, Asia \longrightarrow Router3$$

$$5GBRam \longrightarrow Router2$$

Figure 4: Resolving of the Configuration Resolver

Other than resolving router signaturess to router names the Configuration Resolver can also resolve link configurations (or as we will call link signatures) between the routers and just like routers it can create new links on request of the scientist (user). The link signature is actually a pair of router signatures plus configuration information for the link and what the Configuration Resolver returns is a pair of router names and a link name to the scientist (user).

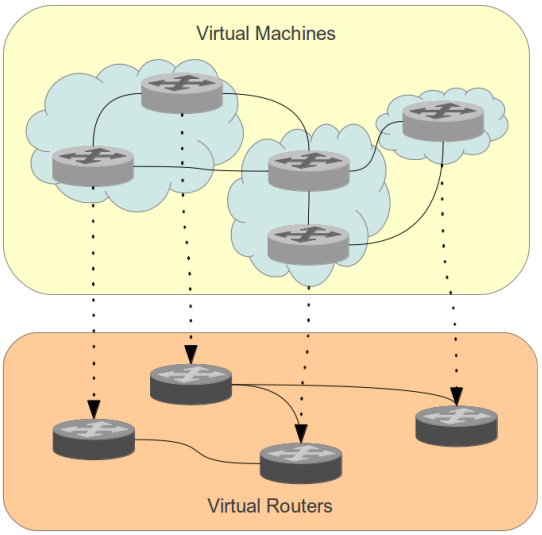

Figure 5: Abstraction done by the Virtual Internet Operating System

3.2 Provider Resolver

The Provider Resolver has the task of resolving router names to a triple consisting of provider, location (if available) and virtual machine names inside the provider (We will call this triple the virtual machine signature). If the lower layers have requested a router name that no virtual machine signature with that router name exists, a virtual machine signature is created by the Provider Resolver and then virtual machine signature is returned to the lower layer. In this case the Provider Resolver will create a random virtual machine(or it could be created based on internal decisions made by the Provider Resolver). If it is requested by the lower layer to have a virtual machine with specific configurations, the lower layer has to explicitly ask for such configurations when passing the router name to the Provider Resolver and the Provider Resolver will return a virtual machine signature to the lower layer.

Figure 6 shows the resolving done by the Provider Resolver. In figure 6, router 8 is resolved to a virtual machine signature that does not have a location. The meaning of this is that the Bright Box provider only has one location or the location is the default location defined in the configuration files of the Provider Resolver.

$$Router2 \longrightarrow AmazonEC2, AsianPasific, VirtualMachine1$$

$$Router1 \longrightarrow AmazonEC2, WesternAmerica, VirtualMachine2$$

$$Router8 \longrightarrow BrightBox, , VirtualMachine1$$

Figure 6: Resolving of the Provider Resolver

Other than having the responsibility of resolving router names the Provider Resolver also resolves link names and or pairs of router names into a tunnel configurations (or as we will call tunnel signatures). The tunnel signature is actually a triple of a pair of virtual machine signatures and tunnel specifications. Just like the case for routers if a tunnel dose not exist, the Provider Resolver will create one and will return its tunnel signature to the lower layer (The Configuration Resolver) or the scientist (user).

3.3 Cloud API

Different cloud providers have different APIs to interact with them. To be able to make the task of creation and maintaining of virtual machines in the Inter-Cloud trivial, it is necessary to have a unified API. This API is a universal API for all cloud providers. It makes it simple for the lower layers to communicate with different cloud providers through one unified interface. An example of such of this layer could be Ruby Fog or Apache Libcloud.

3.4 Cloud Provider API

The Cloud Provider API is the API provided by the cloud providers and differs based on the provider. What this layer provides is actually real access to instantiate and manage Virtual Machines in the providers environment.

In some cases the Cloud Provider API also gives us the ability to run commands on the instantiated Virtual Machines. This makes the process of configuring the Virtual Routers and even more trivial task.

4 Experimental Method

To prove hypothesis 2 only showing that the Virtual Internet can provide a more efficient path only based on one parameter would indeed be enough, therefore our proof of hypothesis can be reduced to proving

Hypothesis 3 *A Virtual Internet can provide lower latency between nodes than the current Internet does*

To prove hypothesis 3 we first have to have a clear definition for latency between nodes in the Virtual Internet so we first concentrate on that. For a given source and destination if we relabel the path between them as N_0 to N_n where N_0 is the source , N_n is the destination and N_i (where $0 < i < n$) is the ith virtual router. When ever a packet is routed through the Virtual Internet we will have a path of $N_0, N_1, ..., N_{n-1}, N_n$. And in case the packet is routed by normal Internet the path will be N_0, N_n.

Now if we would consider the cost of going through a virtual router N_i in the Virtual Internet in terms of latency to be ε_i and the latency between N_a and N_b in the normal Internet as $\delta_{a,b}$ then for that path we have ζ_n as the latency from N_0 to N_n through the Virtual Internet as

$$\zeta_n = \zeta_{n-1} + \varepsilon_{n-1} + \delta_{n-1,n} \tag{1}$$

where $\zeta_0 = 0$ and obviously $\varepsilon_0 = 0$

Now if $\sum_{0<i<n} \varepsilon_i << \zeta_n$ then we have

$$\zeta_n = \zeta_{n-1} + \delta_{n-1,n} \tag{2}$$

Now the definition for latency between a and b through path ρ inside the Virtual Internet can be formulated as

$$\vartheta_{a,b,\rho} = \zeta_n \text{ of } \rho \text{ with source as } a \text{ and destination as } b \text{ and } n = hops(\rho) + 1 \tag{3}$$

Now with equation (3) we can formulate hypothesis 3 as

Hypothesis 4 $\exists a, b, \rho : \vartheta_{a,b,\rho} < \delta_{a,b}$

To prove hypothesis 4 we can use proof by contradiction. Lets say that our hypothesis is wrong then we have

$$!\exists a, b, \rho : \vartheta_{a,b,\rho} < \delta_{a,b} \tag{4}$$

So all we need now is to show that equation 4 is wrong .To show that equation 4 is wrong we have to find end points that will have lower latency going through the Virtual Internet than the normal Internet.

4.1 Limitations and Restrictions

Actually building the Virtual Internet and doing latency measurements on it is not feasible because of limitation in time and the nature of the Virtual Internet as a dynamic application specific network.

To overcome these limitations instead of building the network we will do measurements on latency between different existing end points in the NLNOGRING and virtual machines instantiated in different locations of the Amazon EC2 to see if there are more efficient paths (in terms of latency) in the Internet that the ones chosen by the Internet.For running the test we are limited to ping and traceroute to measure latency because the NLNOGRING only allows us to preform such tasks on it.We can also only do the experiment on IPv4 paths because the Amazon EC2 is not IPv6 capable.

4.2 Experimental Setup

To gain more realistic results, we would like to have more number of nodes with the most global distributed locations. So our setup consists of 3 different environments.

The NLNOGRING The NLNOGRING consisted of 75 servers in 72 different ASs located in 21 different countries.The necessary measurements were collected using a python script that were executed by the NLNOGRING managers on all servers at the same time. The distribution of the 75 servers are shown in figure 7.

Figure 7: NLNOGRING region distribution

The Amazon EC2 We chose Amazon EC2 because of the number of regions supported by this platform. Also the Amazon EC2 is one of the most known and is a pioneer in providing virtual computing power in the cloud[2].

To have a better distributed network, 17 virtual machines in all 7 regions of the Amazon EC2 that were located in 5 different countries were instantiated manually. We only instantiated 17 virtual machines to reduce experimental cost and experimental

[2]The Amazon EC2 platform has been introduced in August 2006.

time. Because we could think of region of the Amazon EC2 to be well connected. Just to make sure that this is the case, we did not limit a number of virtual machines to one per region. What we did is that we created at least two virtual machines in each region and we chose to add one extra virtual machine to regions that had multiple locations(Distributed between different cities closed to each other). The distribution of the Amazon EC2 regions are shown in figure 8 and the number of virtual machines in each region are shown in table 1.

Figure 8: Amazon region distribution

Table 1: The number of virtual machines in Amazon regions

Region Name	Locations	# VMs
US East(North Virginia)	East of United States	4
US West 1(Oregan)	West of United Stated	2
US West 2(North California)	West of United States	2
Eu West	Western Europe	3
Asia Pacific 1(Singapore)	Singapore	2
Asia Pacific 2(Tokyo)	Japan	2
South America(Sao Paulo	Brazil	2

The OS3 Student Server The OS3 student server (minsk.studilab.os3.nl) was setup to act as the experiment controller. After instantiating the virtual machines manually and gaining the key and the domain name provided by Amazon EC2, the keys and the list of all Amazon EC2 servers were stored on the controller. Other than that a complete list of all the NLNOGRING servers were also stored on the controller as a text file.

The Controller was also setup with a shell program specially written for this research that would automatically deploy needed software and data to the virtual machines, compile and install necessary software on the virtual machines, execute required experiments (ping, traceroute and ...) and finally collect the results from the virtual machines. This process was done periodically in the experiment time slice.

The controller code was also able to reset the virtual machines to their initial state it did so by undoing the changes made on each virtual machine. Another ability of the controller was to retrieve and report the current status of the virtual machines based on log files generated by the scripts that were executed on the virtual machines. Although we could have executed all necessary experiments(scripts) manually on the 17 servers but the controller code was written to show that deploying, installing and executing programs on multiple virtual machines via a controller can be a trivial task.

4.3 Experimental Results

Two days worth of measurements were collected by the Amazon EC2 virtual machines and the NLNOGRING servers. These measurements of the experiment were saved to result files(plain text outputs) that totally had the size of about 5GB.

To be able to analyze the data and have a more structured data set all the result files were converted into one large file containing source, destination and round trip time in the CSV format.

It is important to note that the ping process was done five times for each period by each server to all other servers. The round trip time reported by the output of the ping program for each of these five pings, was not accurate enough so instead the ping statistics were used as a basis of the round trip(See figure 9).

```
PING 31.3.104.43 (31.3.104.43) 56(84) bytes of data.
64 bytes from 31.3.104.43: icmp_seq=1 ttl=51 time=84.6 ms
64 bytes from 31.3.104.43: icmp_seq=2 ttl=51 time=83.4 ms
64 bytes from 31.3.104.43: icmp_seq=3 ttl=51 time=85.7 ms
64 bytes from 31.3.104.43: icmp_seq=4 ttl=51 time=84.8 ms
64 bytes from 31.3.104.43: icmp_seq=5 ttl=51 time=88.9 ms

--- 31.3.104.43 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4095ms
rtt min/avg/max/mdev = 83.487/85.537/88.980/1.902 ms
```

Figure 9: A sample view of the ping result file

The CSV file was finally imported to R as a dataset and using R script programming the data was analyzed. The analysis script did the task of calculating distance between nodes inside the Virtual Internet based on given number of Virtual Routers in between and the normal distance between the nodes in th Internet.

5 Analysis

After doing the calculations on our dataset, we were able to find some examples of improvement in latency using one hop in between instead of going thorough a direct path. As we have shown in figure 10, the direct path between a node in Singapore to a node in the Netherlands costs less in terms of latency by going through another node located in Finland.The direct path average latency for the given example was 335.18 ms but the average latency when going through Finland was 244.17 ms that shows a 91.01 ms or 27.5 percent improvement(blue path demonstrates the cheaper path and the red one shows the expensive one in terms of latency).

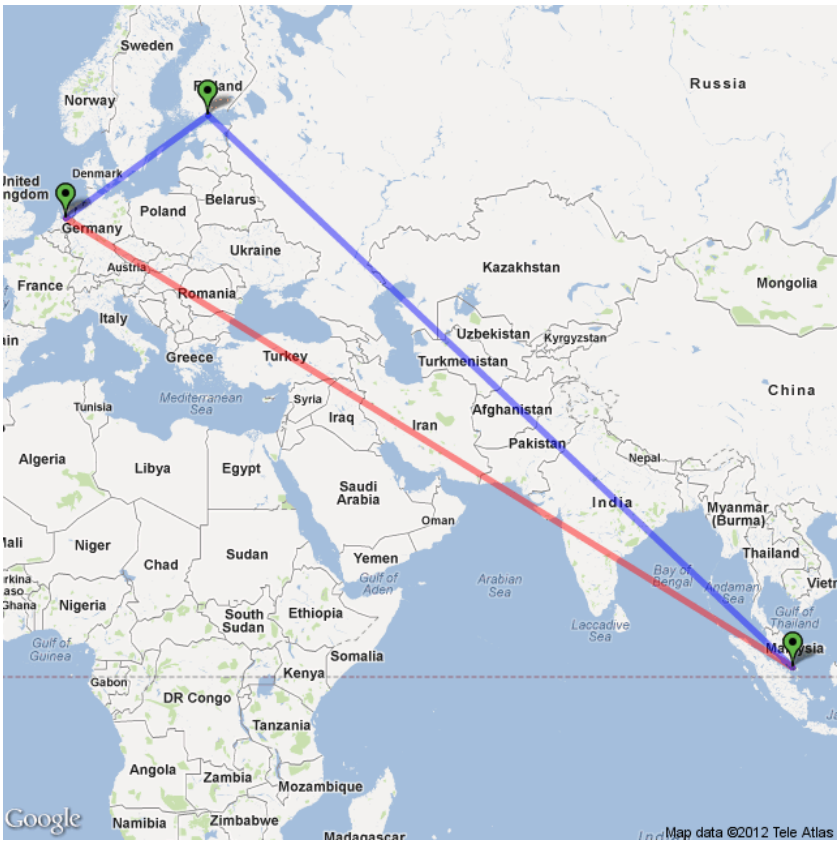

Figure 10: An example of improvement in the Virtual Internet

5.1 Amount of Improvement

Although having one example proves hypothesis 4 but that can not be a good reason for building a Virtual Internet. An initial calculation showed that 78 percent of the paths had an improvement via the Virtual Internet. This percentage is really considerable but an more important factor is how these improvements have been distributed based on amount of improvements in milliseconds because maybe most improvements are less

than a needed amount for specific purpose. Figure 11 shows this distribution.

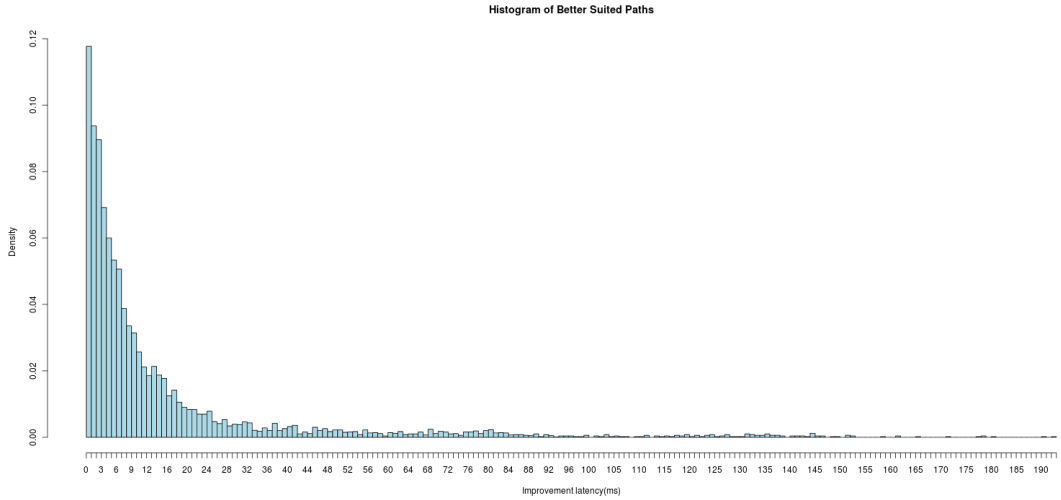

Figure 11: Amount of improvements in terms of milliseconds

As we can see in figure 11 there is an exponential decrease in density when a higher latency improvement is required. For example 12 percent of the paths have less than 1 milliseconds improvements and 4 percents of the paths have the latency improvement between 11 and 12 milliseconds. What is interesting in figure 11, is that there are even improvements in latency of more than 190 milliseconds that although are not much but they ring a bell that maybe there are faulty paths chosen by the Internet.

Figure 11 shows us interesting information but it could also trick us about the actual amount of improvement. For example when the actual latency thorough the Internet path is 5 seconds an improvement of 100 milliseconds will only improve the latency 2 percent but on the other hand when the actual latency thorough the Internet path is 4 milliseconds, an improvement of 3 milliseconds will improve the latency 75 percent that is a considerable amount. So we should also consider distribution of latency in terms of percent that is shown in figure 12.

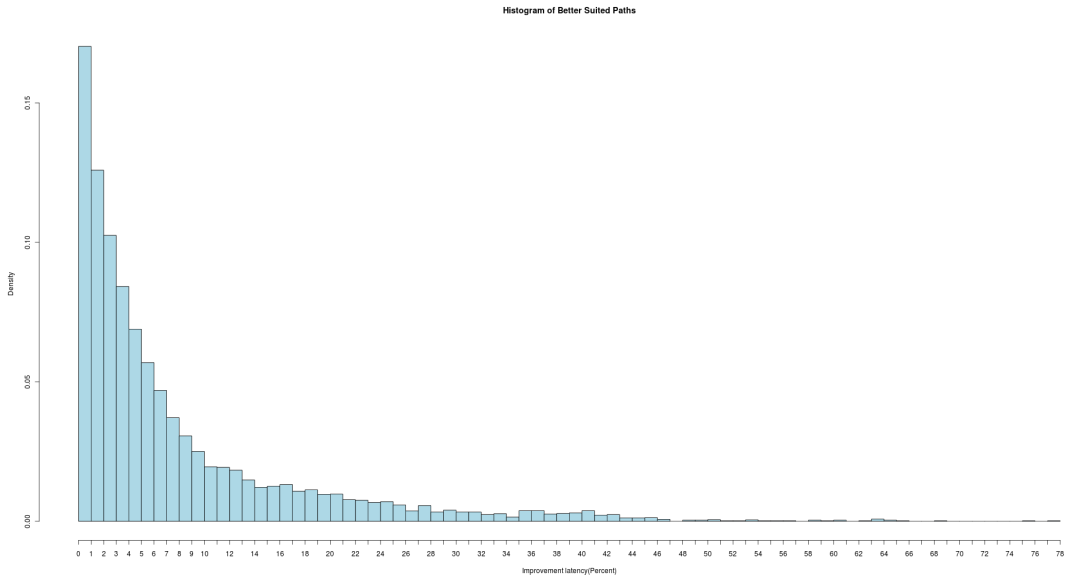

Figure 12: Amount of improvements in terms of percent

As we can in figure 12 there is also an exponential decrease in density when a higher percentage of improvement is required but with a smaller slope. For example about 18 percent of the paths have less than 1 percent improvements and 4 percents of the paths have the latency improvement between 6 and 7 percents.

Figures 11 and 12 are mainly similar so we can conclude the number of paths that could trick us in figure 11 are not that much. We also see cases high amount of improvement for example 78 percent that still rings the bell a faulty path in the Internet.

5.2 Placements of Improvement

To dynamically be able to decide to go thorough the Virtual Internet or not, we should have an estimation of possible improvement in the Virtual Internet based on measurements done with the current Internet path. What would be handy such a case is knowing how much improvement we will have in average or maximum or even minimum. Figures 13,14 and 15 show the average, the maximum and the minimum improvement of going through a path in the Virtual Internet for a given Internet latency range between two nodes.

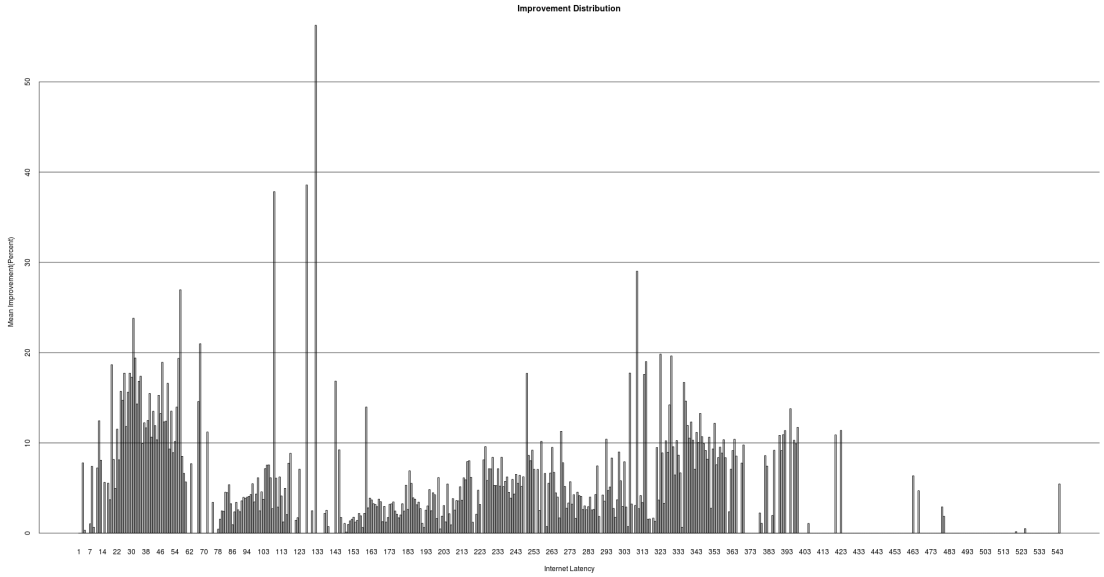

Figure 13: Amount of average improvements in percentage based on the Internet latency ranges

For example figure 13 shows spikes of 56 percent and 38 percent improvements in the ranges of 132-133 and 107-108 milliseconds that would probably be great ranges to choose to go through the Virtual Internet. Further research is needed to investigate exactly why such spikes are happening. This kind of spikes could be caused by a temporarily faulty node or could have permanent reasons such as economic decision making in routing.

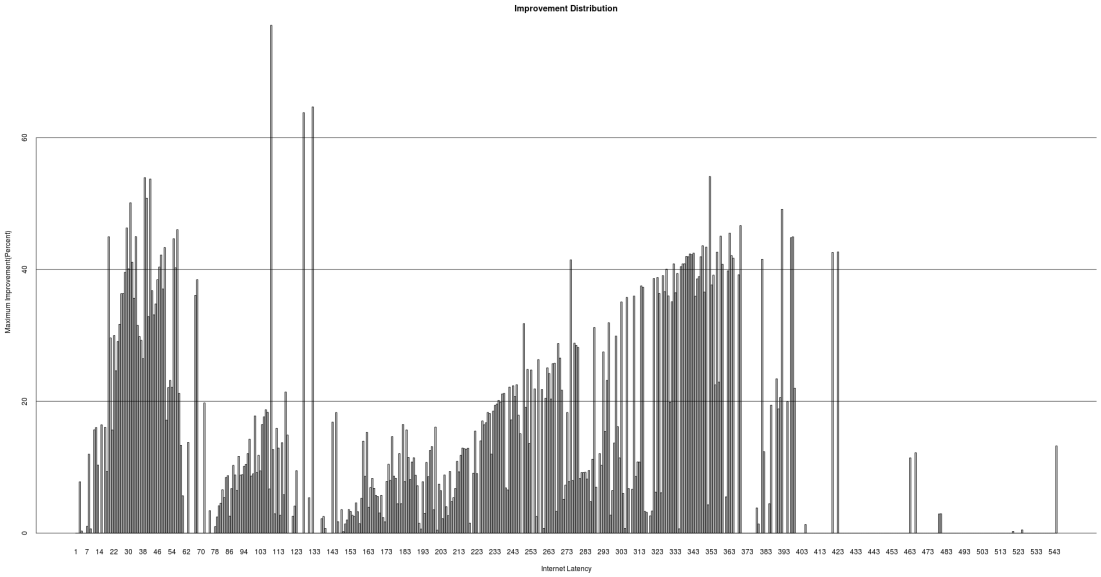

Figure 14: Amount of maximum improvements in percentage based on the Internet latency ranges

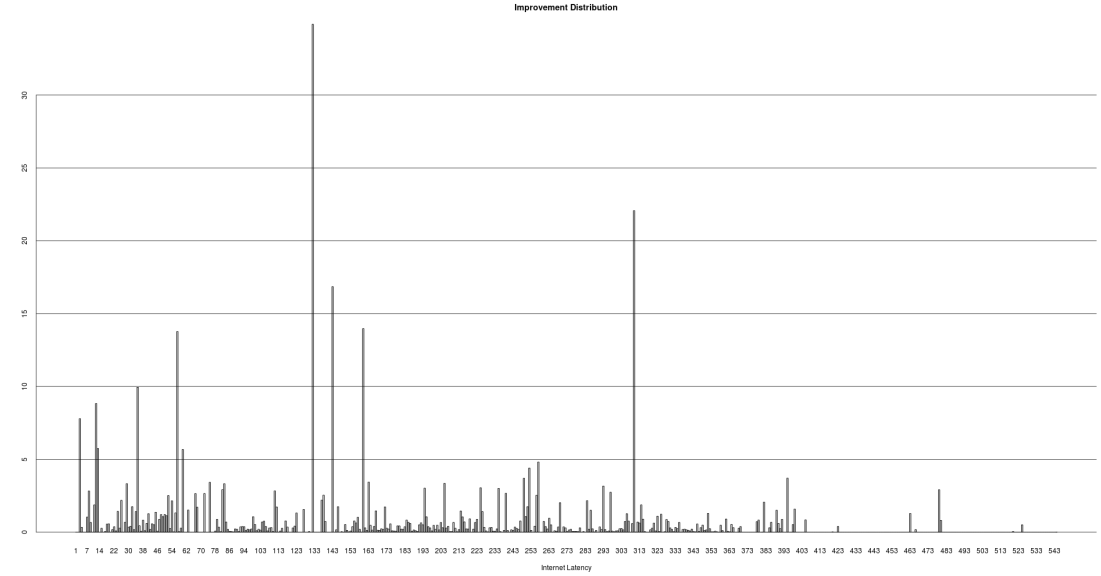

Figure 15: Amount of minimum improvements in percentage based on the Internet latency ranges

6 Conclusion

As we saw in our research, the necessary architectural components to boot a Virtual Internet are Virtual Routers, Virtual Links,The Virtual Internet Operating System and for sake of simplicity a central controller (A decentralized solution is also possible). We also experimented booting such a Virtual Internet for collecting data by writing a controller script that gives us the feeling that it is a trivial task to do so.

We also found out that a Virtual Internet can provide more efficient routes between nodes than the current Internet does by studying latency improvements. Our results showed that 78 percent of the path actually did improve and in some cases had up to more than 50 percent improvements although our results also showed that 12 percent of the cases had less than 1 milliseconds improvement but for some real time applications such as online surgery this one milliseconds could be a life saver.

Overall we can conclude having such a Virtual Internet, can not only act as a good test bed for scientists it can have other application by providing application specific optimization and online changes in it's architecture.

7 Further Work

Further studies could be done on the Virtual Internet in two main subjects

Technical Studies More technical studies such as measuring the stability of a Virtual Internet could be done to show if after constructing a Virtual Internet for how long the conditions of the network (parameters between nodes, existence of nodes and links) can stay in a fairly stable condition or if we are building a dynamic network based on a specific parameter how long would it take the network to get to a fairly stable condition and how long this stability will last.

Internet governance issues Other subjects that further research can be done on are governance and management issues such as how governmental and global regulations could be applied to the subject and also how we could prevent specific content to enter the Virtual Internet and how we can guarantee privacy related issues in such an environment and so on.

8 Acknowledgment

I would like to thank Job Snijders that provided access to the NLNOGRING and also my wife that was very supportive in the process of doing this research.

References

[1] Cloudlab.

[2] H. Balakrishnan. A layered naming architecture for the internet.

[3] B. Braden, T. Faber, and M. Handley. From protocol stack to protocol heap - role-based architecture.

[4] D. R. Cheriton and M. Gritter. A new next generation internet architecture.

[5] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. Openflow: Enabling innovation in campus networks.

[6] Larry Peterson, Scott Shenker, and Jonathan Turner. Overcoming the internet impasse through virtualization.

[7] Gregor Schaffrath, Christoph Werle, Panagiotis Papadimitriou, Anja Feldmann, Roland Bless, Adam Greenhalgh, Andreas Wundsam, Mario Kind, Olaf Maennel, and Laurent Mathy. Network virtualization architecture:proposal and initial prototype.

[8] Andrew S.Tanenbaum and Maarten Van Steen. *Distributed Systems: Principles and Paradigms*. Secound edition.