UNIVERSITI TUNKU ABDUL RAHMAN

REPORT STATUS DECLARATION FORM

Title: **Using Surrogate Servers for Content Delivery Network Infrastructure with Guaranteed QoS**

Academic Session: _____________

I **WONG KHAI HSIANG**

(CAPITAL LETTER)

declare that I allow this Final Year Project Report to be kept in

Universiti Tunku Abdul Rahman Library subject to the regulations as follows:

1. The dissertation is a property of the Library.
2. The Library is allowed to make copies of this dissertation for academic purposes.

Verified by,

_________________________	_________________________
(Author's signature)	(Supervisor's signature)
Address:	
_________________________	
_________________________	_________________________
_________________________	Supervisor's name
Date: _____________________	**Date**: ____________________

Using Surrogate Servers for Content Delivery Network Infrastructure with Guaranteed QoS

By

Wong Khai Hsiang

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

In partial fulfillment of the requirement

for the degree of

BACHELOR OF INFORMATION TECHNOLOGY (HONS)

COMMUNICATIONS AND NETWORKING

Faculty of Information and Communication Technology

(Perak Campus)

MAY 2012

DECLARATION OF ORIGINALITY

I declare that this report entitled "**Using Surrogate Servers for Content Delivery Network Infrastructure with Guaranteed QoS**" is my own work except as cited in the references. The report has not been accepted for any degree and is not being submitted concurrently in candidature for any degree or other award.

Signature	:	___________________________
Name	:	WONG KHAI HSIANG
Date	:	___________________________

ACKNOWLEDGEMENTS

I would like to express my sincere thanks and appreciation to my supervisor, Dr. Lau Phooi Yee who has given me this bright opportunity to engage in an Future Media Internet project. It has been a very eye opening experience for me. A million thanks to you.

Finally, I must say thanks to my parents and my family for their love, support and continuous encouragement throughout the course.

ABSTRACTS

This project is a Research project on Using Surrogate Servers for Content Delivery Network Infrastructure with Guaranteed QoS. The research focuses on *Soarin*, a novel contents delivery system to increase network bandwidth dynamically by deploying delivery servers in a wide area. What this project does is to simulate the concepts of *Soarin* and analyze if it would be viable to be put to used just like the un-released technology used currently in AKAMAI. On providing broadband contents, especially in the views of on-demand contents, having the required capacity of network bandwidth are very important. CDN has some combination of a content-delivery infrastructure, a request-routing infrastructure, a distribution infrastructure, and an accounting infrastructure. The content-delivery infrastructure consists of a set of "surrogate" servers that deliver copies of content to sets of users. CDN can increase network bandwidth so that delivery servers are distributed in a wide area all over the Internet. Before using CDN to deliver contents, contents provider estimate the amount of the access to provision the enough processing power and network bandwidth. However CDN cannot provide their services during overload. This is because CDN cannot increase its network bandwidth and processing power flexibly. Cloud computing makes it possible to increase processing power dynamically by increasing servers. However, current cloud-computing systems cannot increase network bandwidth. This is because it increases servers only in a local area. Servers have to be deployed in a wide area to increase network bandwidth. The end result from this project from the simulation of *Soarin* will be able to determine whether *Soarin* would be a suitable concept to be put into full practical use.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

CDN	Content Delivery Network
QoS	Quality Of Service
ISP	Internet Service Provider
VMM	Virtual Machine Monitor
ES	Execution Server
DS	Distribution Server
OS	Observation Server
AS	Autonomous System
CS	Control Server

Chapter 1: Introduction

What is Content Delivery Network (CDN)? What it does is that it replicates contents over several mirrored web servers (i.e., surrogate servers) which is strategically placed at various locations in order to deal with the *flash crowds*. A CDN the following structures:

1. Content-delivery infrastructure
2. Request-routing infrastructure
3. Distribution infrastructure
4. Accounting infrastructure

CDN improves network performance by maximizing bandwidth usage, improving content accessibility and maintaining content updates through content replication thus offering fast and reliable applications and services by distributing contents to proxies servers located close to users. Figure 1 illustrates a CDN and its components.

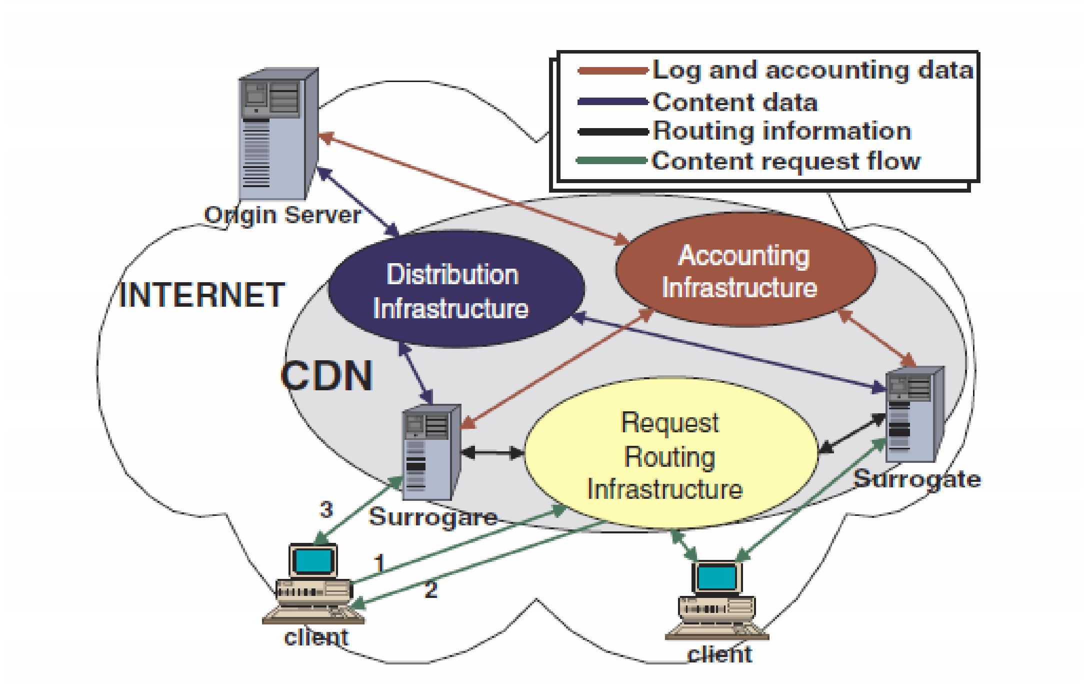


Figure 1: Content Distribution Network Infrastructures

The content delivery architecture consists of a set of **surrogate servers** that deliver copies of content to one or more user while combining different activities such as

a) web storage services
b) file transfer services
c) E-commerce services
d) web applications
e) directory services
f) live(on-demand) services

The **request-routing** infrastructure consists of mechanisms to redirect content requests from a client to a suitable surrogate. The **distribution** infrastructure consists of mechanisms to move contents from the origin server to the surrogates. The **accounting** infrastructure tracks and collects data on request-routing, distribution, and delivery functions within the CDN creating logs and reports of distribution and delivery activities. The client interacts with the CDN through the request routing infrastructure and surrogate servers.

The **origin server** (hosting the content to be delivered) interacts with the CDN in two ways (see Figure 1):

a) It pushes new content to the replica servers, (the replica themselves request content updates from the origin server through the distribution infrastructure).
b) It requests logs and other accounting data from the CDN or the CDN itself provides this data to the origin server through the accounting infrastructure.

Figure 1 shows one of the possible scenarios of interaction between two clients, the access routers, the replica servers and the origin server. A client sends

1. a content request to the routing infrastructure, that redirects
2. the client’s request to a surrogate server, to which the client subsequently asks
3. The desired content.

The design of a CDN requires, together with the distribution of replica servers at the edge of the network, a set of supporting services and capabilities. In order to be efficient for

1. a significant number of users

2. a considerably wide area, the edge servers must be deployed in thousands of networks, at different geographically spread locations.

The optimal performance and reliability depend on the granularity of the distribution of those edge servers. The establishment of a CDN therefore requires the design of some important features the following:-

a) **Replica placement mechanisms** are needed to decide the locations of the replica server and to adaptively update the contents prior to the request arrival (pre-fetching). Thus servers are not updated upon request, i.e. unlike in traditional proxy caching, but are pro-actively updating its content. Adaptivity in replica placement is required, especially to cope with changing traffic conditions though not related to pull behavior as in traditional caching.

b) **Content update mechanisms** must be provided to automatically check the host site for changes and retrieve updated content for delivery to the edges of the network, thus ensuring content freshness. Standard mechanisms adopted in proxy caching do not guarantee content freshness since content stored on standard cache servers does not change as the source content changes.

c) **Active measurement mechanisms** must be added to cooperative access routers in order to provide immediate access to a real-time picture of the Internet traffic, i.e. to the ability to recognize the fastest route from the requesting users to the replica servers in any type of traffic situations, especially in presence of "flash crowds". Flash crowds refers to the sudden heavy demand, expected or not, for a single site. A measurement activity is at the basis of the replica selection mechanism.

d) **Replica selection mechanisms** must be added to cooperative access routers to accurately locate the closest and most available edge server from which the end users can retrieve the required content. A robust service must also keep its servers from getting overloaded by means of access control and load balancing.

e) **Re-routing mechanisms** must be able to quickly re-route content requests in response to traffic bursts and congestion as revealed by the measurement activity. Also, the CDN infrastructure, must allow the service providers to directly access the caches and control their availability and to get the statistics information about the accesses to the site, from the cooperative access routers.

Chapter 2: Project Background

Over the last decades, users have witnessed the growth and maturity of the Internet. As a consequence, there has been an enormous growth in network traffic, driven by rapid acceptance of broadband access, along with increases in system complexity and content richness [1]. The over-evolving nature of the Internet brings new challenges in managing and delivering content to users. As an example, popular Web services often suffer congestion and bottleneck due to the large demands made on their services. A sudden spike in Web content requests may cause heavy workload on particular Web server(s), and as a result a *hotspot* [2] can be generated. Coping with such unexpected demand causes significant strain on a Web server. Eventually the Web servers are totally overwhelmed with the sudden increase in traffic, and the Web site holding the content becomes temporarily unavailable. Content providers view the Web as a vehicle to bring rich content to their users. A decrease in service quality, along with high access delays mainly caused by long download times, leaves the users in frustration. Companies earn significant financial incentives from Web-based e-business. Hence, they are concerned to improve the service quality experienced by the users while accessing their Web sites. As such, the past few years have seen an evolution of technologies that aim to improve content delivery and service provisioning over the Web. When used together, the infrastructures supporting these technologies form a new type of network, which is often referred to as content network [3].

Several content networks attempt to address the performance problem through using different mechanisms to improve the Quality of Service (QoS). One approach is to modify the traditional Web architecture by improving the Web server hardware adding a high-speed processor, more memory and disk space, or maybe even a multi-processor system. This approach is not flexible [4]. Moreover, small enhancements are not possible and at some point, the complete server system might have to be replaced. Caching proxy deployment by an ISP can be beneficial for the narrow bandwidth users accessing the Internet. In order to improve performance and reduce bandwidth utilization, caching proxies are deployed close to the users. Caching proxies may also be equipped with technologies to detect a server failure and maximize efficient use of caching proxy resources. Users often configure their browsers to send their Web request through these caches rather than sending directly to origin servers. When this configuration is properly done, the user's entire browsing session goes through a specific caching proxy. Thus, the caches contain most popular content viewed by all the users of the caching proxies. A provider may also deploy different levels of local, regional, international caches at geographically distributed locations. Such arrangement is referred to as *hierarchical caching*. This may provide additional performance improvements and bandwidth savings [5]. A more scalable solution is the establishment of server farms. It is a type of content network that has been in widespread use for several years. A server farm is comprised of multiple Web servers, each of them sharing the burden of answering requests for the same Web site [4]. It also makes use of a Layer 4-7 switch, Web switch or content switch that examines content request and dispatches them among the group of servers. A server farm can also be constructed with surrogates [63] instead of a switch. This approach is more flexible and shows better scalability [4]. Moreover, it provides the inherent benefit of fault tolerance [1]. Deployment and growth of server farms progresses with the upgrade of network links that connects the Web sites to the Internet. Although server

farms and hierarchical caching through caching proxies are useful techniques to address the Internet Web performance problem, they have limitations. In the first case, since servers are deployed near the origin server, they do little to improve the network performance due to network congestion. CDN can increase network bandwidth so that delivery servers are distributed
in a wide area all over the Internet. Before using CDN to deliver contents, contents provider estimate the amount of the access to provision the enough processing power and network bandwidth. However CDN cannot provide their services during overload. This is because CDN cannot increase its network bandwidth and processing power flexibly. Cloud computing makes it possible to increase processing power dynamically by increasing servers.

However, current cloud-computing systems cannot increase network bandwidth. This is because it increases servers only in a local area. Servers have to be deployed in a wide area to increase network bandwidth. There are three problems in deploying servers in a wide area. These problems are composed of three main parts:

(1) how to deploy the servers
(2) where to deploy the servers
(3) when to deploy the servers

There are some research contributions for wide area live migration [6][7]. The results of these researches can be used in server deployment in a wide area. However, the purpose of these studies focuses how to deploy the servers. We tackle the problem when to deploy the servers.

2.1 Literature Review

There have been numerous proposed methodologies in the placement and usage of Surrogate Servers in the past. One particular paper [8] proposed a new CDN architecture by creating server clusters in a single content delivery network. According to the proposed architecture, it can achieve better server load balance and it is fit for the future large storage requirement for CDN services. With the coordination among content serves, client requests are satisfied by different servers within a cluster. It is on the basis that in a CDN, a content server has no knowledge about the data stored on its neighbouring servers. If it cannot satisfy a client request, it has to forward the request to the original server even in case the required data is available in its neighbouring servers. Moreover, if a server is overloaded, its neighbouring servers are not aware of that and cannot help to reduce its workload even their workload are very low. Another serious issue is the cache space limitation, since more and more content providers use CDN service, the storage requirement increases dramatically, new services such as high quality videos, video on demand, real-time video conferences and real time stock information generate even higher storage requirement and exacerbate the problem, thus the cache hit rate on these CDN servers keeps decreasing.

On another paper [9], techniques for building the dissemination tree, a dynamic content distribution network were explored. Several replica placement algorithms which reduce the number of replicas deployed and self-organized them into a balanced dissemination tree. A peer-to-peer location service used for better scalability and locality was also proposed. The methodology is proposed based on the purpose of dynamically choosing the number and placement of replicas while satisfying QoS requirements and server capacity constraints and to disseminate updates to the replicas on small delay and bandwidth consumption. On a third paper [10] a novel content delivery system was proposed with the basis that it can increase network bandwidth dynamically by delivery servers in a wide area. It is also mentioned that it can use various server deployment policy to deploy delivery servers and to decide on server suitability for the role of content deliverer. The issue that were tackled were on the inability to dynamically increase delivery servers in existing CDNs concluding that existing CDNs lack flexibility to deploy distribution servers.

In the first paper[8] where a server clustering was proposed is rather unrealistic. Firstly, to have a cluster of servers, it would mean deploying numerous physical servers within an area meaning spending a lot of money just to focus on a singular geographical area to provide QoS within the area. If the concept were to be used by a CDN provider, it would mean building numerous physical servers in each geographical area which would add up to a very big number of servers. The cost itself would be a big step back. Secondly, the congestion in an area needed to be taken in consideration. There would be no point to be deploying clusters of servers in areas that are not congested. The physical distance between servers would also be a stumbling block. How far would a server need to be to be classified as another cluster and how near one should be? What about the network bandwidth within an area? Just because the distance between two surrogates are low does not guarantee that the bandwidth would suffice, there would be scenarios where the distance is not a factor but the bandwidth would be. A physical surrogate may be nearer but if the network bandwidth between the server and client isn’t ideal, a further surrogate might be a better option. The proposal of the second paper [9] is a plausible thing that may be put into real-world conditions. However, it would only be satisfy a small to medium area of servers. Take Akamai, for instance. They have a total of 95 000 surrogates globally. The overhead of calculating everything in such a large scale would overwhelm the whole network. CDN is all about providing availability whenever there is a request. Let’s say, calculation is done in every geographical area. That would able to work, however, as data of how many clients request changes in real-time (i.e. the sudden emergence of a flash crowd), discrepancies in those data might overthrow the whole calculation process and a need to re-calculate everything would therefore cause a massive delay of content delivery.

In comparison to the current un-published technology of Akamai[7], which consists of “EdgeComputing” and “NOCC” which provides delivery of web-based applications that scale on demand, a proactive monitoring and troubleshooting of all servers in the global Akamai network respectively, the third paper[10] proposed a rather similar concept. “Soarin” as proposed in the third paper make uses of Virtual Machines Machines (VMM) within physical servers to monitor network performance and these physical machines execute virtual machines that make up Execution Servers(ES), Distribution Servers(DS). By usage of virtual machines within physical machines, using multiple Distribution Servers (DS) would be easily done as a the number of physical Distribution Servers(DS) might prove to be a bottleneck. The only drawback is that “Soarin” has yet to be tested and there is no physical evidence to back up the methodology proposed.

Chapter 3: Project Scope and Objectives

The project aims to solidify a new methodology in the deployment of surrogate servers in CDNs to guarantee Qos. Research will be done on SOARIN about Server Proliferation (VMM,ES,DS) and the Server Deployment Policy of Soarin.

After that, the concept of SOARIN would be put to the test in a simulation that replicates real-world environments and see how it would perform.

The project scopes include:

1. Understanding how SOARIN works in terms of Server Proliferation and the Server Deployment Policy
2. Simulating the concept of SOARIN in a real-world environment to solidify the concept as proposed in [10].
3. Analyzing the result of simulation in (2)
4. Proposing an extended-SOARIN for new environment./scenario

Chapter 4: REQUIREMENT AND DESIGN

4.1 Introduction

Server Proliferation deploys physical machines that are installed with virtual machine monitor throughout the Internet in advance. These physical machines would execute virtual machines on them. These physical machines are named *Execution Server* (ES). The other server is *Deployment Server* (DS).DS stores the virtual machines HDD images. In Server Proliferation, services (i.e. Web server, Streaming server) will be executed inside virtual machines. When a new virtual machine is required, the virtual machine's HDD image will be distributed from DS to one of the ES. The distributed virtual machine is executed on the ES

Server Proliferation realizes the increasing and decreasing processing power and network bandwidth of server system dynamically by increasing and decreasing servers in a wide area dynamically. Figure 1 shows architecture of Server Proliferation.

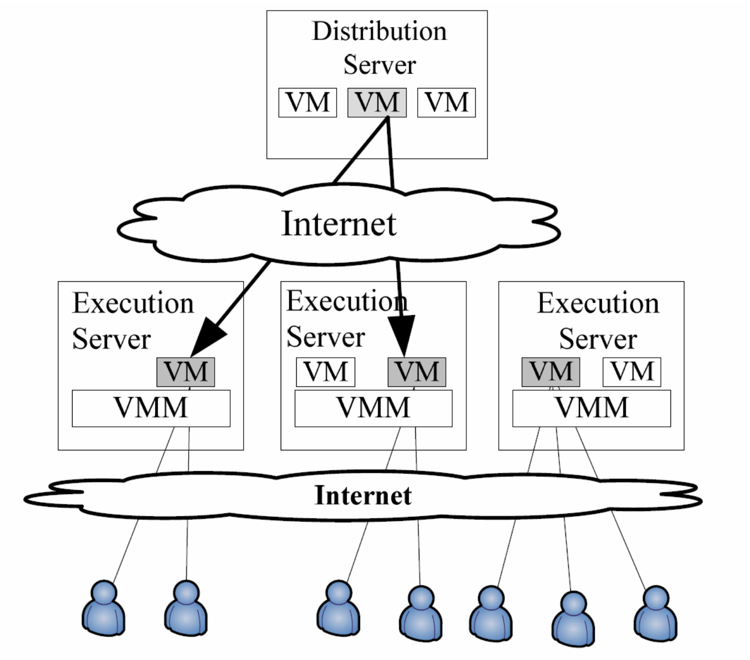


Figure 2: General View of Server Proliferation

We introduce two types of the servers in Server Proliferation.

1. *Execution Server* (ES)
2. *Distribution Server* (DS)

Without Server Proliferation, one CDN will be just like Cloud computing which makes it possible to increase servers dynamically. By increasing servers, it is possible to use the CPU and the network of the increased server. Thus processing power and network bandwidth is increased. Therefore cloud-computing systems can increase the processing power and network bandwidth. However, compared with processing power, it is difficult to increase network bandwidth. This is due to network bottleneck. Typical cloud-computing system is constructed in an iDC (Internet Data Center). The uplink network of the iDC may become network bottleneck of the cloud computing system.

By contrast, Server Proliferation can increase both of processing power and network bandwidth. It is because it can deploy servers in a wide area; therefore, deployed servers can use different uplink network each other. In this case, where SOARIN is concerned, it does not matter how many servers are to be deployed in the specific area, with virtualization, the mountain of costs of server deployments can be avoided. As it turned out, it is possible to increase network bandwidth of the system. Server Proliferation uses virtual machine as a basis. It is because virtual machine is easy to increase and decrease dynamically. Moreover using virtual machine can reduce cost since physical machines can be shared with other system that uses virtual machines. It is possible to execute another virtual machine besides virtual machine executed by Server Proliferation. We can say that Server Proliferation is high-cost performance.

4.2 SOARIN

In this section, we describe Flexible Contents Delivery System with Dynamic Server Deployment: **SOARIN**. SOARIN enables flexible increase in network bandwidth. SOARIN increases distribution servers to increase network bandwidth. SOARIN can increase distribution server anytime. It is possible to add network bandwidth even after content distribution is started. In addition, SOARIN can decide when and where to increase distribution server flexibly. Upon increasing distribution servers dynamically, the new problem when and where to increase surrogates happens. In SOARIN, distribution servers are constructed inside virtual machines, this problem is equal to a problem what criteria a physical machine to execute a virtual machine is chosen. This selection criterion is called *Server Deployment Policy*. There is a variety of server deployment policy and SOARIN is able to use various server deployment policies for contents holders' requirements. The examples of the server deployment policy are discussed later in this section.

SOARIN uses Server Proliferation to deploy distribution servers. Server Proliferation is able to provide on how to deploy a virtual machine dynamically. However it lacks capability to decide timing and location of deploying virtual machine. Hence, the introduction of *Observation Server* (OS) and *Control Server* (CS) in addition to *Execution Server* (ES) and *Deployment Server* (DS) in Server Proliferation. OS collects several kinds of metrics using server deployment policy. For example, it collects the CPU load and network traffic of ESs, calculates the distance from ESs to DSs. CS controls all over SOARIN's system. CS selects ES to deploy new virtual machine using information from OS based on server deployment policy. CS will then direct DS to transfer the HDD image of the virtual machine to the selected ES to deploy new distribution server. After that, CS directs ES to execute the virtual machine for new distribution server. Finally CS updates request navigation policy to use the new distribution server. Figure 2 will illustrate a general overview of how OS , CS ,DS and ES work.

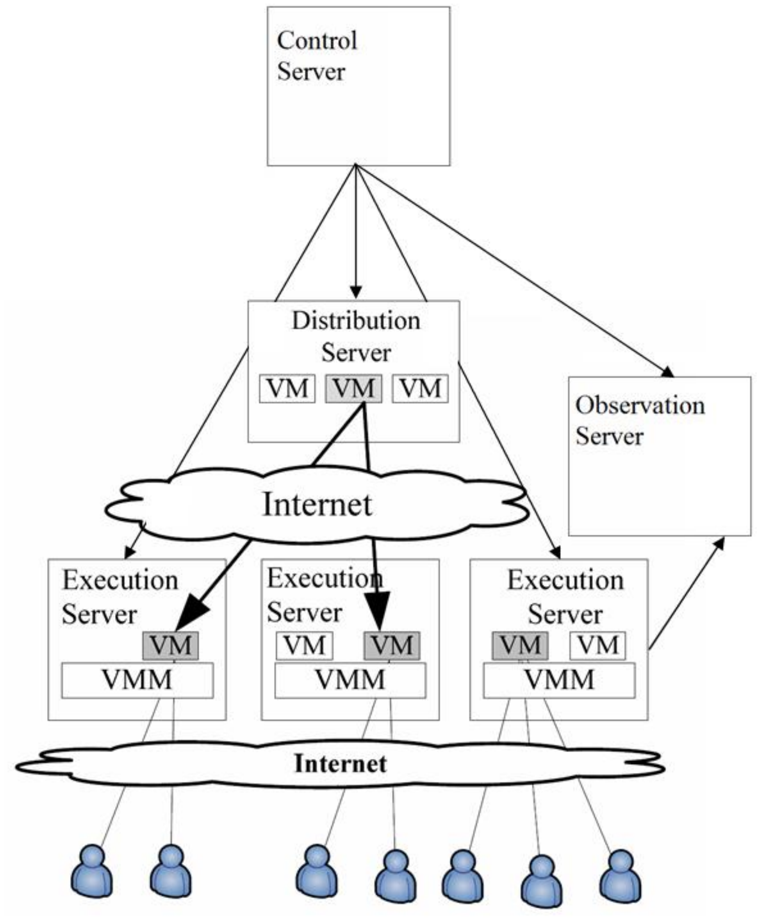


Figure 3: General Overview of CS and OS

4.2.1 Server Deployment Policy of SOARIN

As mentioned above SOARIN is able to use various server deployment policies. In this project however, only one Server Deployment Policy of SOARIN will be used to deal with a scenario of a sudden spike in user connected (peak hour) which is where we take into account the distance between the execution server and deployment server. i.e, how far an execution server is from the deployment server (origin server).

4.2.2 Distance between Execution Server and Deployment Server

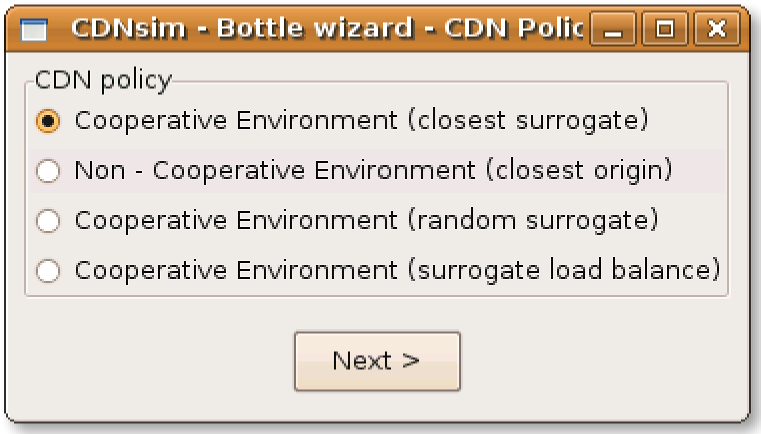
This policy uses the distance between Deployment Server and Execution Server as a criterion. Some metrics can be used to calculate the distance which includes number of hops, round trip time, and AS (Autonomous System) path length between DS and ES. DS measure this information periodically. OS collects this information from DS. This policy chooses the nearest ES from DS; therefore it may deploy new distribution server in a short time. As a result, it is possible to correspond to a sudden surge in the volume of request (peak hour). In this project, when a surrogate is not able to fulfill the client's request, the request will be directed to the nearest surrogate or origin to fulfill that request.

4.2.3 The Simulation Software

CDNSim has been designated to provide a realistic simulation for CDNs, simulating the surrogate servers, the TCP/IP protocol, and the main CDN functions. The main advantages of this tool are its high performance, its extensibility, and its user interface, which is used to configure its parameters. CDNSim provides an automated environment for conducting experiments and extracting client, server, and network statistics.

To run CDNSim, open a terminal (assuming bash or dash shell and CDNsim is installed in the home directory of the current user) and type the following in order to invoke the wizard (change path as necessary).

The first screen that appears defines the CDN redirection policy: The client is redirected to a surrogate server according to the policy. Upon a cache miss the surrogate server is redirected also according to the same policy.


CDNsim - Bottle wizard - CDN Polic

CDN policy

● Cooperative Environment (closest surrogate)
○ Non - Cooperative Environment (closest origin)
○ Cooperative Environment (random surrogate)
○ Cooperative Environment (surrogate load balance)

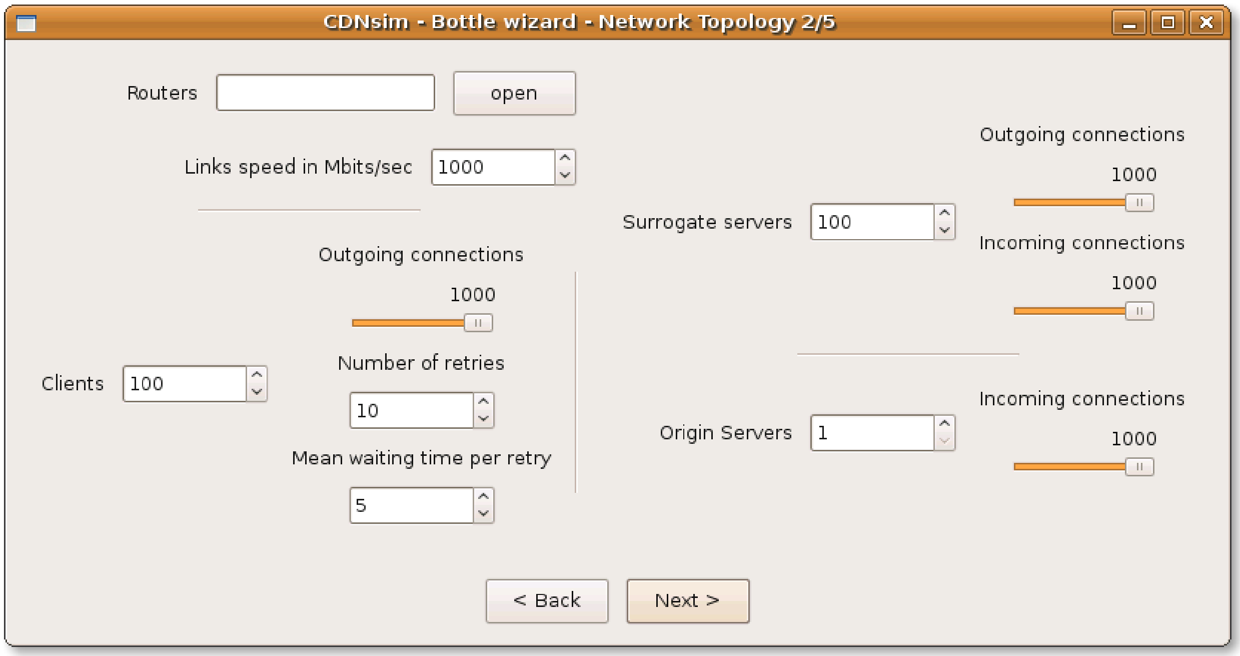
- **Cooperative Environment (closest surrogate).** The client is redirected to the closest surrogate server in terms of network hops. Upon a cache miss, the surrogate server retrieves the object from the closest alternative surrogate server that contains the requested object. The object is stored in the cache and then it is served to the client. If the object is not outsourced at all in any surrogate server then the surrogate server retrieves the object from the closest origin server.

Consecutive cooperations may occur if in the meantime the alternative surrogate server removed the object from the cache causing another cache miss.

- **Non - Cooperative Environment (closest origin).** The client is redirected to the closest surrogate server in terms of network hops. Upon a cache miss, the surrogate server retrieves the object from the closest origin server, stores it in cache and serves it to the client.
- **Cooperative Environment (random surrogate).** The client is redirected to a random surrogate server. Upon a cache miss, the surrogate server retrieves the object from a random alternative surrogate server that contains the requested object. The object is stored in the cache and then it is served to the client. If the object is not outsourced at all in any surrogate server then the surrogate server retrieves the object from a random origin server. Consecutive cooperations may occur if in the meantime the alternative surrogate server removed the object from the cache causing another cache miss.
- **Cooperative Environment (surrogate load balance surrogate).** The client is redirected to the closest surrogate server in terms of network hops. If the surrogate server is loaded ~95% then the client is redirected to the least loaded surrogate server. Upon a cache miss, the surrogate server retrieves the object from the closest alternative surrogate server that contains the requested object. Again if the load is ~95% the surrogate server is redirected to the least loaded surrogate server that contains the object. The object is stored in the cache and then it is served to the client. If the object is not outsourced at all in any surrogate server then the surrogate server retrieves the object from the closest origin server. If the origin server load is ~95% then the surrogate is redirected to the least loaded origin

server. Consecutive cooperations may occur if in the meantime the alternative surrogate server removed the object from the cache causing another cache miss.

The next screen provides options for the network topology. CDNsim may build any kind of wired network topology as long as the appropriate configuration is provided. Advanced tuning can be performed by editing the base.ned file once the simulation bottles are generated by the end of this tutorial. In this page we try to simplify the procedure by offering some basic options:


CDNsim - Bottle wizard - Network Topology 2/5

Routers

Links speed in Mbits/sec 1000

Clients 100

Outgoing connections 1000

Number of retries 10

Mean waiting time per retry 5

Surrogate servers 100

Outgoing connections 1000

Incoming connections 1000

Origin Servers 1

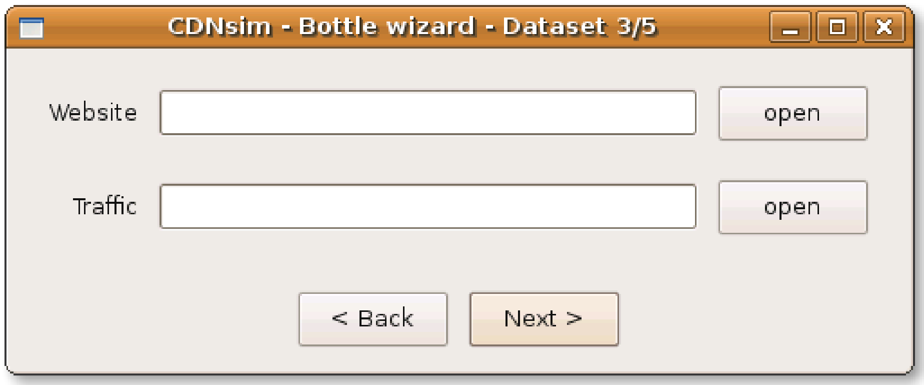
Incoming connections 1000

- **Routers.** The user must provide a text file that describes the backbone network topology represented by a "cloud" of routers. The file format is "**int int\n**" without the quotes. Each integer represents a node id, each line represents a link between the two nodes and "\n" is the new line character. There is no need to define "double" links i.e. 1 3 and 3 1. The default behaviour is to create double links. The node ids must be in the range **[0, number_of_routers)**. All the other network nodes (surrogate servers, origin servers, clients) are randomly attached to the routers backbone. If you provide the same options you will get the same network topology each time you run the wizard. You may find useful to use one of these files (as 3037, random 1000, random 50, transit stub 1008, waxman 1000)

- **Link speed.** This option defines the speed of each link. All the links will have the same speed. Creating a network with varying speeds require manual editing of the generated network topology file.
- **Number of clients.** This is the number of clients that exist in the network.
- **Clients' number of outgoing connections.** Each client may handle more than one connection for requesting objects. This option sets an upper limit.
- **Clients' number of retries.** Each client can retry that much times if a request is failed.
- **Clients' mean waiting time per retry**. Following the exponential distribution with this mean the each client remains idle until next retry.
- **Number of surrogate servers.** This is the number of surrogate servers the CDN has.
- **Surrogate servers' number of outgoing connections.** Upon cache misses the surrogate servers have to retrieve the requested objects from alternative servers. This option defines the maximum number of such connections. Denial of service occurs if this limit is met.

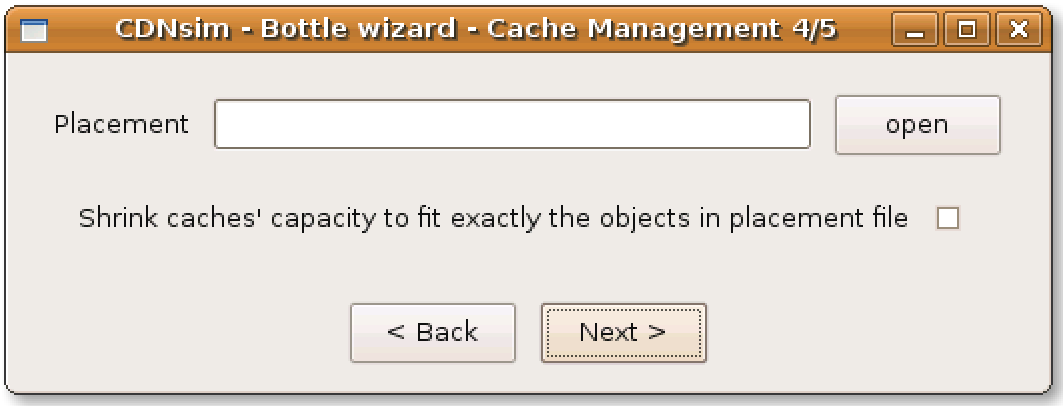
- **Surrogate servers' number of incoming connections.** This sets the maximum number of connections a surrogate may receive from other surrogate servers or clients for objects. Denial of service occurs if limit is reached.
- **Number of origin servers.** Defines the number of origin servers in the network. The origin servers contain the whole web site and they are the last resort for a request to be satisfied.
- **Origin servers' number of incoming connections.**. Defines the number of incoming requests for objects the origin servers can handle simultaneously. Denial of service occurs if this limit is exceeded.

In the next screen the user inputs the web site and the clients trace file. More specifically:



- **Website.** This file contains all the objects existing in the system and can be transferred. The format is "**int int\n**", where the first integer is a unique object id and the second integer is the size in bytes. The object ids must be in the range **[0,number_of_objects)**. For example you may use this file web_site16000.

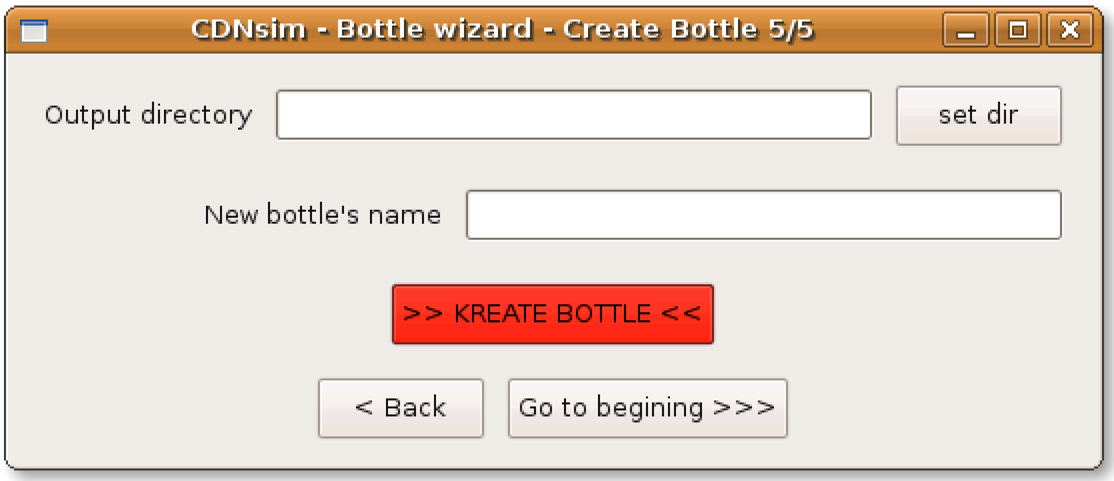
- **Traffic.** This file describe which clients, when and what they will be request from theCDN. The file format is "**double int int\n**". The first number is the timestamp of the request, the second is the client and the last is the requested object. The timestamp takes values >=0 and the file must be sorted by increasing timestamp. The clients take values in >= 0. If the client numbers are in the range [0, number_of_clients) wherenumber_of_clients is the one defined in step4 then it is all OK. Otherwise the client numbers in the trace file are reassigned randomly to meet the range [0,number_of_clients) but preserving the requests each client is originally assigned to. If you provide the same input you will get the same assignment each time you run the wizard. The requested objects must be in the range [0 number_of_objects). Here is a trace file for your convenience trace_file50000.


CDNsim - Bottle wizard - Cache Management 4/5

Placement

Shrink caches' capacity to fit exactly the objects in placement file ☐

This step configures each surrogate server`s local cache. The configuration is set by a file which describes the contents, the capacity and the cache replacement policy of every surrogate server. The file contains records, each one referring to a surrogate server. There is an extra option "Shrink..." this would make caches to fit the initial contents exactly. For instance if you configure a cache to be 1GB and the file that contains the objects to be stored contains e.g. 1MB of objects, them the cache will become 1MB.



Here the user must provide input for the following:

- **Output directory.** This is the directory where the output (not the results of the simulation) will be stored.
- **New bottle's name. Descriptive** name for the new experiment.

At this step all the input provided in the previous steps and all the necessary libraries are verified and bundled inside a compressed archive called bottle. If anything is invalid then you get an error message. You may open the generated archive (it is tar.gz format) and take a look. All the procedure can be repeated to create more bottles.. Advanced network topology tuning can be performed by editing the base.ned file inside the bottles.

The purpose of CDNSim is to be used as a testbed for CDN evaluation and experimentations subsequently making it a suitable choice to be used for the objectives of my project. In terms of simulating a CDN, there are numerous software out there that can serve such purpose (i.e. NS-2, OPNET). However, the reason CDNsim is used is because it was built on top of OMNET as an extension specifically used to simulate CDNs. It is used on a Linux Platform. Unlike other simulation softwares, I do not have to start from scratch in the purpose of doing my project. With CDNsim, a lot of functions (request routing, traffic files and number of components in the topology) are all provided as sample and to edit them, it is a matter of changing the parameters in the files. In a nutshell, with CDNsim, a lot of the works which I would have to do from scratch have already been simplified for me. All I needed to do was take a bit of time to study the architecture and codes of CDNsim and I am ready to use it in my project.

In terms of SOARIN, the ability to manipulate topologies in CDNSim allows me to use create a simple topology of 3 nodes for my objective. By using the CDNsim.py, I am able to build the topology for any kind of CDN which includes all the options of manipulating the number of clients, servers and origins. By running the sh script, it will simulate the topology by using the .ned and .hi inside the topology and by calling the .cc file in CDNsim lib, INET lib and HACK folder. All of the .cc files were built by the author of CDNsim. After running the simulation, the whole simulation process will be logged into stats a STDOUT file. From the analyzation of the result from STDOUT file, I will be able to come to grasp on what had been simulated.

On modifying the topology, CDNsim allows us to control the model of the topology. Also, built in is a provided request routing to route the request and other protocol and libraries. In terms of my project, I will be using 10 nodes with the IDs of S1 – S0. . CDNsim provides the basic model for building a CDN; hence it helps in simplifying the amount of work that I have to do. By assuming that the central unit is the request routing system, I am able to proceed to only work on the modification of topology and the different roles that each node plays. If say one of the nodes does not a certain file, central unit being the one that controls all of the surrogate servers within the singular CDN will route the missing file from the origin server or from the closest node that has the file. All in all, after the whole simulation is done, from the *stat* file, I will be able find the proof of routing from CDN which shows the traffic of the packet transfer between nodes.

Chapter 5: Simulation and Results

5.1 Preliminary Investigation Case

This section discusses the topology of the CDN that will be simulated will be simulated. Referring to Figure 4, the total number of routers in the topology: 15 (R0 – R 14), and when we refer to Figure 5, we are able to see that the number of clients: 10 (C15 – C24) , surrogates: 10(S25 – S34) and 1 origin. The full topology on how each component connects with each other is illustrated both in a diagram (Figure 5) and the router topology is illustrated in Figure 4. In terms of Table 1, the two rows of routers that is shown is the connection between routers (i.e. Router 0 is connected to Router 1) whilst in Table 2, the two rows of components are the inter-connected network devices and their bandwidth. Each client's request will is put into tables (Table 3) to show at what timeline each request cross-referenced with which object being requested. The client tables are divided into 3 rows which includes Client (which identifies the client), Timeline (the timeframe of at which the clients makes a request) and lastly, Object ID (which is the ID of the object which they will make a request of). In Table 4, we will see the Object ID and its' size. It is important to see the table as CDNSim's traffic files only shows the object size instead of the object ID. The STDOUT file and STATS file outputs are also included as the results of the simulation. However, both the STDOUT and STATS file outputs are only limited to 0 – 10 timeline because the output is too long and the recorded output is enough to illustrate the results of the simulation.

Router Topology Table 1 as illustrated in Figure 4(on the right)

Router	Router
0	1
1	2
1	7
2	3
2	4
2	5
3	7
3	6
4	5
5	8
5	9
6	10
6	12
7	10
8	11
9	14
10	12
12	13
13	14

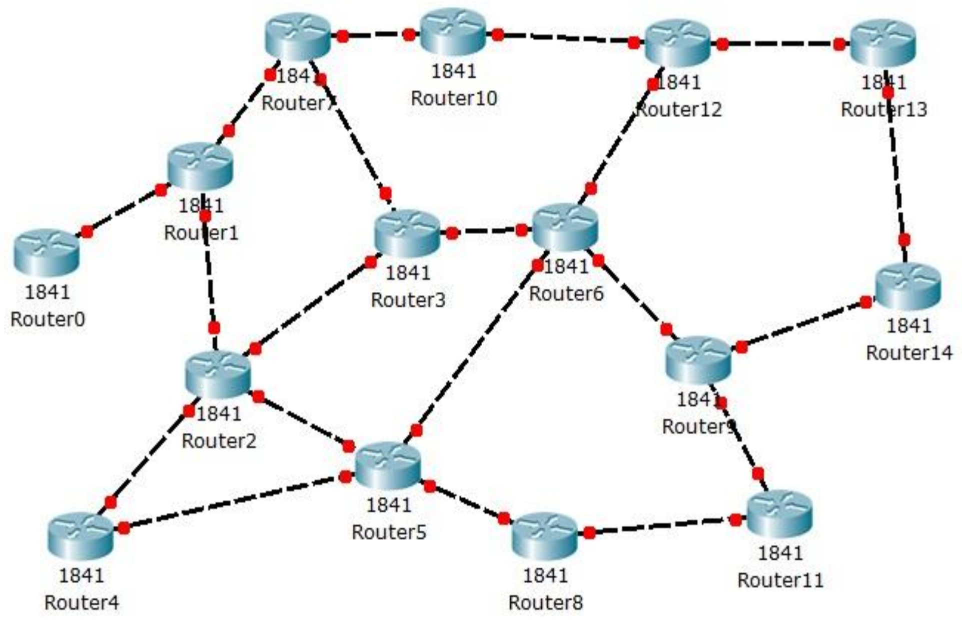


Figure 4: Router Placement (Connection Table shown in Table 1)

Topology Table 2 as illustrated in Figure 5

Component	Component	Bandwidth(Mbps)
r0	r1	100
r0	o39	100
r1	r2	100
r1	r7	100
r1	c16	100
r2	r1	100
r2	r3	100
r2	r4	100
r2	r5	100
r2	s32	100
r2	c22	100
r3	r2	100
r3	r7	100
r3	r6	100
r3	s28	100
r4	r2	100
r4	r5	100
r4	s26	100
r4	c15	100
r5	r2	100
r5	r4	100
r5	r9	100
r5	r8	100

Component	Component	Bandwidth(Mbps)
r10	r7	100
r10	r6	100
r10	r12	100
r10	c23	100
r10	s33	100
r11	r8	100
r11	r9	100
r11	s30	100
r11	c19	100
r12	r10	100
r12	r16	100
r12	r13	100
r12	s29	100
r12	c24	100
r13	r12	100
r13	r14	100
r13	s31	100
r14	r13	100
r14	r9	100
r14	c20	100
c15	r4	100
c16	r1	100
c17	r9	100

r5	s27	100
r6	r3	100
r6	r9	100
r6	r12	100
r6	r10	100
r6	c18	100
r7	r1	100
r7	r10	100
r7	r3	100
r7	s25	100
r8	r5	100
r8	r11	100
r8	c21	100
r9	r6	100
r9	r5	100
r9	r11	100
r9	r14	100
r9	s34	100
r9	c17	100
c18	r6	100
c19	r11	100
c20	r14	100
c21	r8	100
c22	r2	100
c23	r10	100
c24	r12	100
s25	r7	100
s26	r4	100
s27	r5	100
s28	r3	100
s29	r12	100
s30	r11	100
s31	r13	100
s32	r2	100
s33	r10	100
s34	r9	100
o35	r0	100

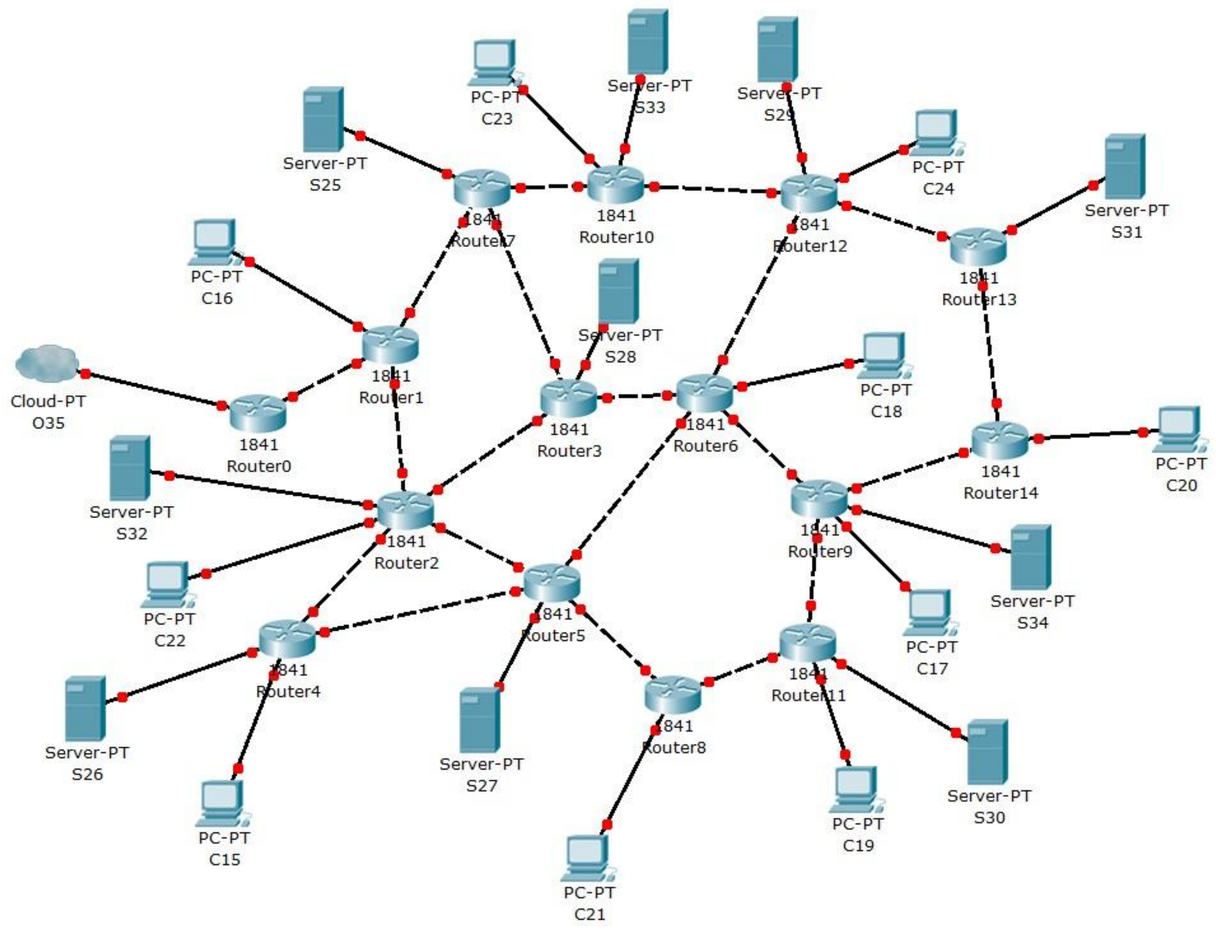


Figure 5: CDN Topology used in the simulation (Connection Table Information in Table 2)

Table 3: Client-Object Request Table

Client	Timeline	Object ID
15	0	10
	10	10
	20	3
	30	7
	40	6
	50	0
	60	9
	70	1
	80	5
	90	8
	100	2
16	1	2
	11	9
	21	5
	31	9
	41	7
	51	6
	61	6
	71	9
	81	1
	91	5
17	2	8
	12	2
	22	10
	32	5
	42	3
	52	7
	62	7
18	3	5
	13	8
	23	2
	33	10
	43	6
	53	3
	63	3
	73	6
	83	0
	93	9
19	4	1
	14	5
	24	8
	34	2
	44	10
	54	5
	64	5
	74	7
	84	6
	94	0
20	5	9
	15	1
	25	5
	35	8
	45	2
	55	10
	65	10
	75	3

	72	0
	82	9
	92	1

	85	7
	95	6

Client	Timeline	Object ID
21	6	0
	16	9
	26	1
	36	5
	46	8
	56	2
	66	10
	76	6
	86	3
	96	7

Client	Timeline	Object ID
24	9	3
	19	7
	29	6
	39	0
	49	9
	59	1
	69	5
	79	8
	89	2
	99	10

22	7	6
	17	0
	27	9
	37	1
	47	5
	57	8
	67	2
	77	10
	87	8
	97	3

23	8	7
	18	6
	28	0
	38	9
	48	1

	58	5
	68	8
	78	2
	88	10
	98	1

Table 4: Object ID

Object ID	Object Size
0	388637
1	71125
2	58512
3	101924
4	106333
5	368734
6	299117
7	108522
8	60248
9	79751
10	256125

5.2 Preliminary Case Study Experimental Results

This section discusses the experimental results.

Table 5: STDOUT File Events.

Line	Status	Timeline	Component	Object Size	Object ID
1	UTIL_UP	0.000145712	o35	256125	10
2	UTIL_DOWN	0.0059406	s33	256125	10
3	UTIL_UP	0.00608536	s33	256125	10
4	UTIL_DOWN	0.0113115	c15	256125	10
5	UTIL_UP	1.00012	o35	314637	
6	UTIL_DOWN	1.0019	s28	58512	2
7	UTIL_UP	1.00204	s28	58512	2
8	UTIL_DOWN	1.00341	c16	58512	2
9	UTIL_UP	2.00015	o35	374885	
10	UTIL_DOWN	2.00214	s33	316373	
11	UTIL_UP	2.0023	s33	316373	
12	UTIL_DOWN	2.00384	c17	60248	8
13	UTIL_UP	3.0001	o35	743619	
14	UTIL_DOWN	3.00779	s30	368734	5
15	UTIL_UP	3.00792	s30	368734	5
16	UTIL_DOWN	3.01534	c18	368734	5

17	UTIL_UP	4.00012	o35	814744	
18	UTIL_DOWN	4.00216	s26	71125	1
19	UTIL_UP	4.00229	s26	71125	1
20	UTIL_DOWN	4.00391	c19	71125	1
21	UTIL_UP	5.00015	o35	894495	
22	UTIL_DOWN	5.00252	s33	396124	
23	UTIL_UP	5.00268	s33	396124	
24	UTIL_DOWN	5.00461	c20	79751	9
25	UTIL_UP	6.00012	o35	1283132	
26	UTIL_DOWN	6.00836	s28	447149	
27	UTIL_UP	6.00849	s28	447149	
28	UTIL_DOWN	6.01632	c21	388637	
29	UTIL_UP	7.00006	o35	1582249	
30	UTIL_DOWN	7.00613	s27	299117	6
31	UTIL_UP	7.00624	s27	299117	6
32	UTIL_DOWN	7.01231	c22	299117	6
33	UTIL_UP	8.00012	o35	1690771	
34	UTIL_DOWN	8.0029	s26	179647	
35	UTIL_UP	8.00303	s26	179647	
36	UTIL_DOWN	8.00538	c23	108522	7
37	UTIL_UP	9.00012	o35	1792695	

38	UTIL_DOWN	9.00276	s28	549073	
39	UTIL_UP	9.00291	s28	549073	
40	UTIL_DOWN	9.00527	c24	101924	3

Sample Result 1: Stats File Records

```
0.000000,SURROGATE,s33,MISS,10
0.000000,ORIGIN,o35,HIT,10
COMPLETED,CLIENT,c15,10,0,0.000000,0.011352,-

1.000000,SURROGATE,s28,MISS,2
1.000000,ORIGIN,o35,HIT,2
COMPLETED,CLIENT,c16,2,0,1.000000,1.003455,-

2.000000,SURROGATE,s33,HIT,8
COMPLETED,CLIENT,c17,8,0,2.000000,2.003894,-

3.000000,SURROGATE,s30,HIT,5
COMPLETED,CLIENT,c18,5,0,3.000000,3.015385,-

4.000000,SURROGATE,s26,HIT,1
COMPLETED,CLIENT,c19,1,0,4.000000,4.003953,-

5.000000,SURROGATE,s33,MISS,9
5.000000,ORIGIN,o35,HIT,9
COMPLETED,CLIENT,c20,9,0,5.000000,5.004657,-

6.000000,SURROGATE,s28,MISS,0
6.000000,ORIGIN,o35,HIT,0
COMPLETED,CLIENT,c21,0,0,6.000000,6.016357,-

7.000000,SURROGATE,s27,HIT,6
COMPLETED,CLIENT,c22,6,0,7.000000,7.012347,-

8.000000,SURROGATE,s26,MISS,7
8.000000,ORIGIN,o35,HIT,7
COMPLETED,CLIENT,c23,7,0,8.000000,8.005423,-

9.000000,SURROGATE,s28,MISS,3
9.000000,ORIGIN,o35,HIT,3
COMPLETED,CLIENT,c24,3,0,9.000000,9.005319,-
```

As we cross-reference the STDOUT (Table 5) and STATS (Sample Result 1) files' outputs, at the following event with line 1,2,3 and 4 of STDOUT file,

It is shown that(Sample Result 1) at time 0.00000, surrogate 33 got a request for object 10, but it does not have the content hence it forwards the request to the origin server, which receives the request and sends the object at which at time 0.011352, client 15 receives the object.

With the STDOUT (Table 5) file, UTIL_UP means that there is traffic of transfer while UTIL_DOWN means that the traffic of transfer is completed. As seen in the sample result 2, illustrated in line 5, the object size does not have an object ID. This is because it is the cumulative figure of the object sizes sent so far (line 1 – 4 + line 6). It is a general pointer on how much the load has the server gone through. To explain how to understand the STDOUT file, as we take a look at line 1, the origin server is sending the object toward surrogate 33. As seen at line 2, surrogate 33 has received the object and the traffic has completed. In Line 3, surrogate 33 proceeds to complete the request from client 15 for the object and in line 4, client 15 receives the object and the traffic is completed once more. Cross-referenced the results in both the STDOUT and STAT output files, and from there, we will be able to fully grasp the network's traffic. Though it may seem that the STAT's file output is a tad bit slower than in the STDOUT file, it is due to the run time problem which exists within the simulator (CDNSim) itself. The simulator proceeds to print the traffic into the STDOUT file before printing it to the stat file which explains the delay of traffic information.

The objective of the simulation was to show that the CDN is able to fulfill requests of clients by providing sources of closest surrogates OR if not available, re-direct requests to origin so that the origin will be able to fulfill the request. By cross-referencing the SDTOUT and stats files recorded above, we will able to see that. Each of the line referenced is made in regards to the STDOUT table above while the stats records will be used directly with each statement.

First off, at Line 8, time: 1.0, C16 requested for Object 2. Upon checking, the closest surrogate which responded is s28 replying that only the origin has object 2. Hence, s28 re-directs the request to the origin,o35 which then proceeds to fulfill the request at time 1.003455

```
1.000000,SURROGATE,s28,MISS,2
1.000000,ORIGIN,o35,HIT,2
COMPLETED,CLIENT,c16,2,0,1.000000,1.003455,-
```

Sample Record 2

At Line 12, time: 2.0, C17 requested Object 8 and the closest source with object 8(s33) responded and proceeded with fulfilling the request at time 2.003894

```
2.000000,SURROGATE,s33,HIT,8
COMPLETED,CLIENT,c17,8,0,2.000000,2.003894,-
```

Sample Record 3

At Line 16, time: 3.0, C18 requested Objet 5, the closest source s30, responded and fulfilled the request at time 3.015385

```
3.000000,SURROGATE,s30,HIT,5
COMPLETED,CLIENT,c18,5,0,3.000000,3.015385,-
```

Sample Record 4

Similar to line 12 and line 16, Line 20 at time: 4.0, C19 requests for object 1, which the closest source is s26, responded.

```
4.000000,SURROGATE,s26,HIT,1
COMPLETED,CLIENT,c19,1,0,4.000000,4.003953,-
```

Sample Record 5

At Line 24, time: 5.0, which is similar with line 8, C20 requested for object 9,but the closest source is the origin where s33 redirected the requested toward.

```
5.000000,SURROGATE,s33,MISS,9
5.000000,ORIGIN,o35,HIT,9
COMPLETED,CLIENT,c20,9,0,5.000000,5.004657,-
```

Sample Record 6

5.3 Experimental Set-up 1(Closest Surrogate)

This section will discuss the simulation in terms of Closest Surrogate. What is Closest Surrogate? In reference to CDNsim, Closest Surrogate is where the client is redirected to the closest surrogate server in terms of network hops. Upon a cache miss, the surrogate server retrieves the object from the closest alternative surrogate server that contains the requested object. The object is stored in the cache and then it is served to the client. If the object is not outsourced at all in any surrogate server then the surrogate server retrieves the object from the closest origin server. Consecutive cooperation may occur if in the meantime the alternative surrogate server removed the object from the cache causing another cache miss.

There will be 2 models which will be simulated in terms of Closest Surrogate where Model 1 has a fixed number of 10 surrogates but will have increasing number of clients (10, 15 and 20) and Model 2 where it has a fixed number of 20 clients but will have an increasing number of surrogates(5,10 and 20). Refer below (after 5.6) for the topologies for the model. (Figure 10 – 15).

5.4 Experimental Set-Up 1 Results

Referring to Table 6, 7 and Figure 6, 7, it is observed that “Closest Surrogate” is able to handle the increases of clients but needs time to stabilize the increased capacity. In Table 6, we are able to observe that when the number of clients climbed to 15, the throughput drops quite drastically. However, as the number of clients continues to increase, we are able to see that the throughput slowly improves and though not as good as the 1:1 surrogate-client ratio; is still quite impressive considering that at 20 clients, the surrogate-client ratio is at 1:2. As we look at Table 7, when the number of clients is fixed and the number of surrogates is increased, as the surrogate-client ratio goes back to 1:1, we are able to see that the throughput goes into an upward pattern (figure 7).

5.4.1 Results for varying number of clients (Surrogate Fixed)

Number Of Surrogates	Number of Client	Throughput(Mbps)
10	10	61.4839
10	15	51.8004
10	20	57.9785
10	25	56.6786
10	30	56.4982

Table 6: Throughput Results

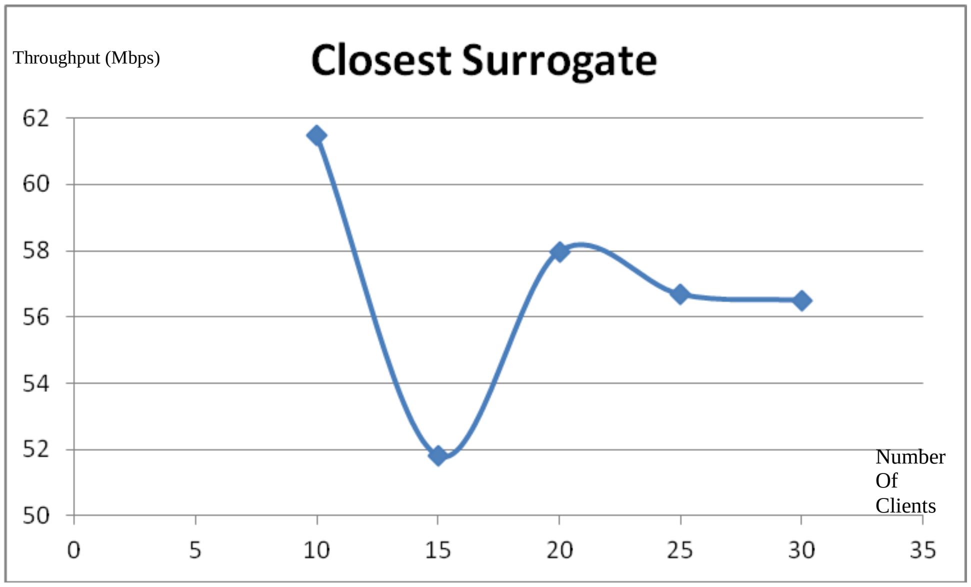


Figure 6: Graphical Interpretation of Throughput Results

5.4.2 Results for varying number of surrogates (Client Fixed)

Number Of Surrogates	Number of Client	Throughput(Mbps)
5	20	57.9324
10	20	57.9785
20	20	61.2917
25	20	62.7338
30	20	64.0003

Table 7: Throughput Results

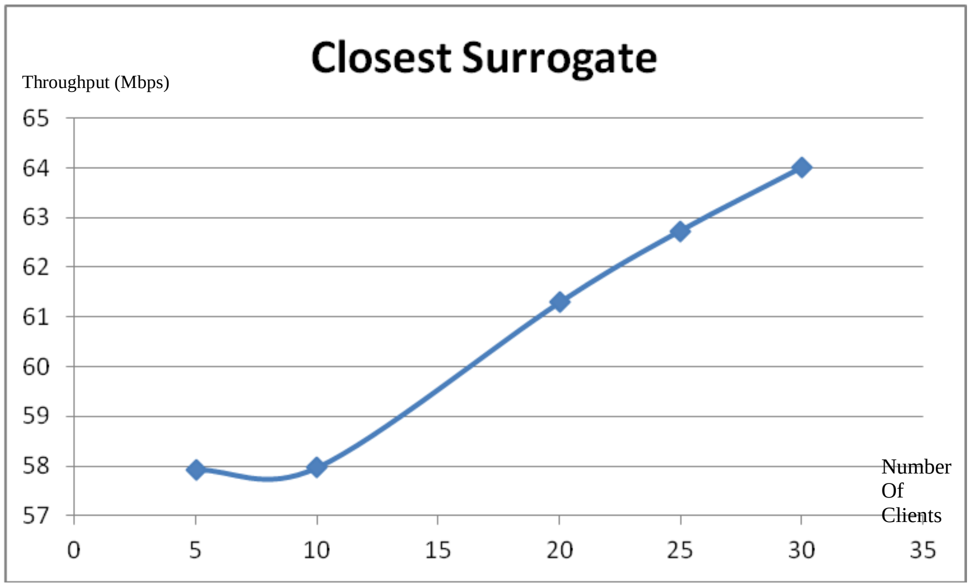


Figure 7: Graphical Interpretation of Throughput Results

5.5 Experimental Set-up 2(Random Surrogate)

This section will discuss the simulation in terms of Random Surrogate. What is Random Surrogate? In reference to CDNsim, The client is redirected to a random surrogate server. Upon a cache miss, the surrogate server retrieves the object from a random alternative surrogate server that contains the requested object. The object is stored in the cache and then it is served to the client. If the object is not outsourced at all in any surrogate server then the surrogate server retrieves the object from a random origin server. Consecutive cooperation may occur if in the meantime the alternative surrogate server removed the object from the cache causing another cache miss.

Again, there will be 2 models which will be simulated in terms of Random Surrogate where Model 1 has a fixed number of 10 surrogates but will have increasing number of clients (10, 15 and 20) and Model 2 where it has a fixed number of 20 clients but will have an increasing number of surrogates(5,10 and 20). Refer below (after section 5.6) for the topologies for the model. (Figure 10 – 15).

5.6 Experimental Set-Up 2 Results

Referring to Table 8, 9 and Figure 8, 9, it is observed that “Random Surrogate” works best at 1:1 Surrogate-Client ratio. In Table 8, we are able to observe that as the number of client increases, the throughput goes through a decline pattern. While when the number of clients is the same as the number of surrogates, the throughput is recorded to be quite good. However, as the number of clients starts outnumbering the number of surrogates, the throughput starts to go down. As we look at Table 9, when the number of clients is fixed and the number of surrogates is increased, the throughput slowly climbs back as the surrogate-client ratio goes back to 1:1.

5.6.1 Results for varying number of clients (Surrogate Fixed)

Number Of Surrogates	Number of Client	Throughput(Mbps)
10	10	64.0788
10	15	56.0951
10	20	55.9788
10	25	54.9965
10	30	49.9604

Table 8: Throughput Results

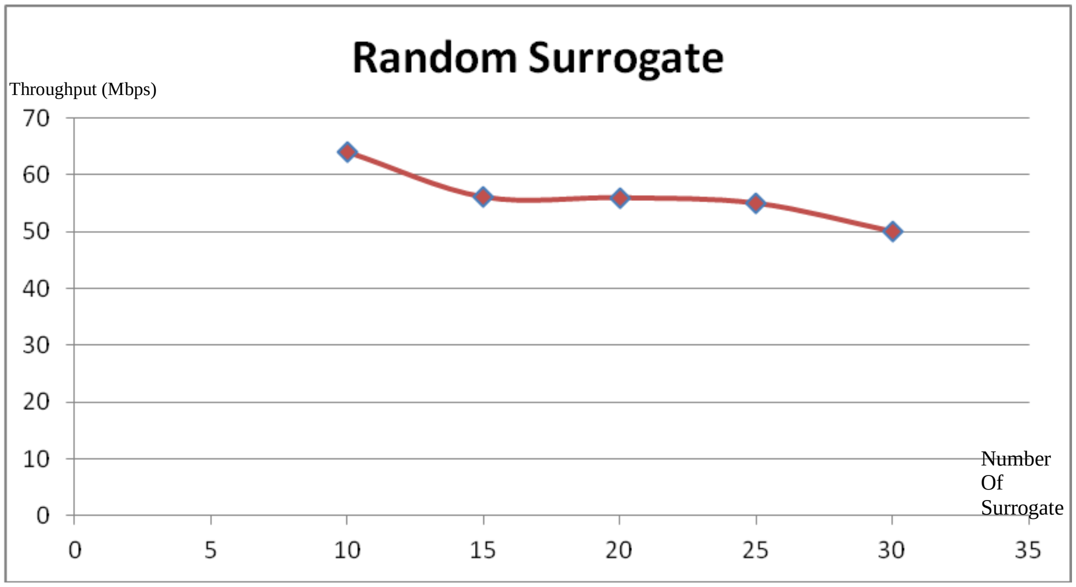


Figure 8: Graphical Interpretation of Throughput Results

5.6.2 Results for varying number of clients (Surrogate Fixed)

Number Of Surrogates	Number of Client	Throughput(Mbps)
5	20	48.1495
10	20	55.9788
20	20	62.0368
25	20	62.4888
30	20	64.2012

Table 9: Throughput Results

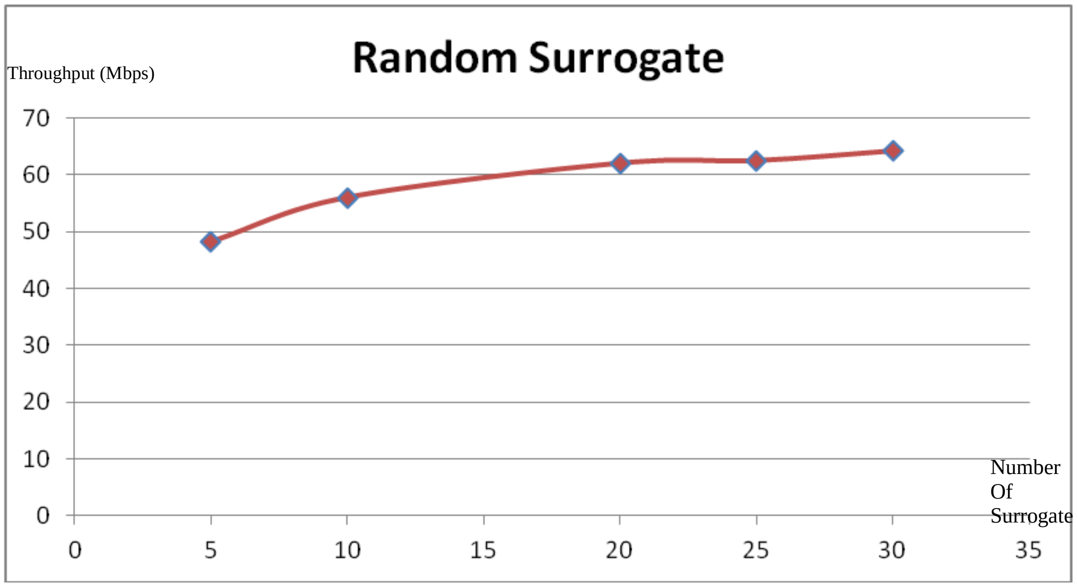


Figure 9: Graphical Interpretation of Throughput Results

5.7 Topology Tables and Figures

Topology Table 10 as illustrated in Figure 10

Component	Component	Bandwidth(Mbps)
r0	r1	10
r0	o39	10
r1	r2	10
r1	r7	10
r1	c16	10
r2	r1	10
r2	r3	10
r2	r4	10
r2	r5	10
r2	s32	10
r2	c22	10
r3	r2	10
r3	r7	10
r3	r6	10
r3	s28	10
r4	r2	10
r4	r5	10
r4	s26	10
r4	c15	10
r5	r2	10
r5	r4	10
r5	r9	10
r5	r8	10
r5	s27	10
r6	r3	10
r6	r9	10

Component	Component	Bandwidth(Mbps)
r10	r7	10
r10	r6	10
r10	r12	10
r10	c23	10
r10	s33	10
r11	r8	10
r11	r9	10
r11	s30	10
r11	c19	10
r12	r10	10
r12	r16	10
r12	r13	10
r12	s29	10
r12	c24	10
r13	r12	10
r13	r14	10
r13	s31	10
r14	r13	10
r14	r9	10
r14	c20	10
c15	r4	10
c16	r1	10
c17	r9	10
c18	r6	10
c19	r11	10
c20	r14	10

r6	r12	10
r6	r10	10
r6	c18	10
r7	r1	10
r7	r10	10
r7	r3	10
r7	s25	10
r8	r5	10
r8	r11	10
r8	c21	10
r9	r6	10
r9	r5	10
r9	r11	10
r9	r14	10
r9	s34	10
r9	c17	10

c21	r8	10
c22	r2	10
c23	r10	10
c24	r12	10
s25	r7	10
s26	r4	10
s27	r5	10
s28	r3	10
s29	r12	10
s30	r11	10
s31	r13	10
s32	r2	10
s33	r10	10
s34	r9	10
o35	r0	10

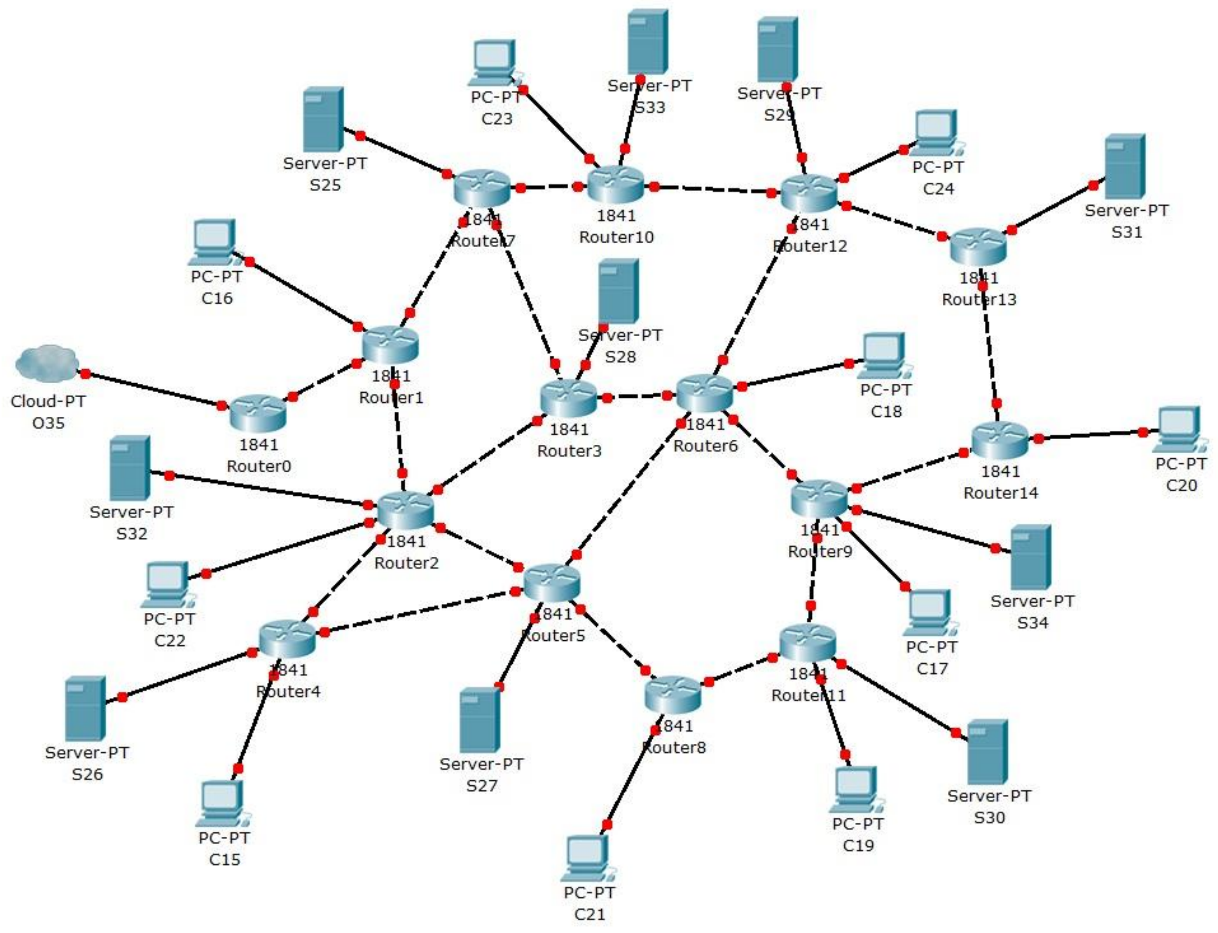


Figure 10: 10 Surrogates 10 Clients

Topology Table 11 as illustrated in Figure 11

Component	Component	Bandwidth(Mbps)
r0	r1	10
r0	o40	10
r1	r2	10
r1	r7	10
r1	c16	10
r2	r1	10
r2	r3	10
r2	r4	10
r2	r5	10
r2	s37	10
r2	c22	10
r3	r2	10
r3	r7	10
r3	c29	10
r3	r6	10
r3	s28	10
r4	o36	10
r4	r2	10
r4	r5	10
r4	s26	10
r4	c15	10
r5	r2	10
r5	r4	10
r5	r9	10
r5	r8	10
r5	c25	10
r5	s27	10
r6	r3	10
r6	r9	10
r6	r12	10
r10	r6	10
r10	r12	10
r10	c23	10
r10	s38	10
r11	r8	10
r11	r9	10
r11	s35	10
r11	c19	10
r12	r10	10
r12	r16	10
r12	r13	10
r12	s29	10
r12	c24	10
r13	c27	10
r13	r12	10
r13	r14	10
r13	s36	10
r14	o38	10
r14	r13	10
r14	r9	10
r14	c20	10
c15	r4	10
c16	r1	10
c17	r9	10
c18	r6	10
c19	r11	10
c20	r14	10
c21	r8	10
c22	r2	10
c23	r10	10

r6	r10	10
r6	c18	10
r7	c26	10
r7	r1	10
r7	r10	10
r7	r3	10
r7	s25	10
r8	o37	10
r8	c28	10
r8	r5	10
r8	r11	10
r8	c21	10
r9	r6	10
r9	r5	10
r9	r11	10
r9	r14	10
r9	s39	10
r9	c17	10
r10	o39	10
r10	r7	10

c24	r12	10
c25	r5	10
c26	r7	10
c27	r13	10
c28	r8	10
c29	r3	10
s30	r7	10
s31	r4	10
s32	r5	10
s33	r3	10
s34	r12	10
s35	r11	10
s36	r13	10
s37	r2	10
s38	r10	10
s39	r9	10
o40	r0	10

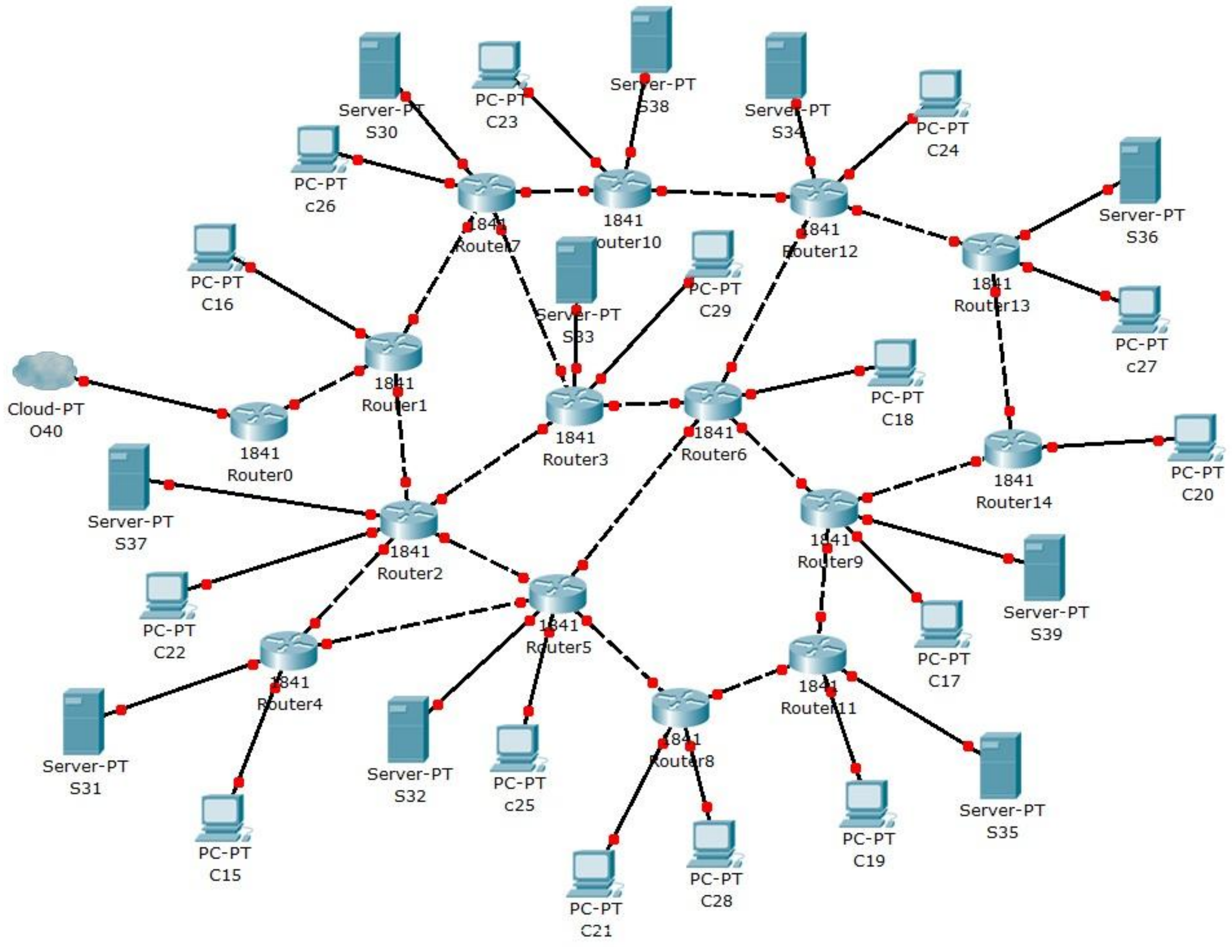


Figure 11: 10 Surrogates 15 Clients

Topology Table 12 as illustrated in Figure 12

Component	Component	Bandwidth(Mbps)
r0	r1	10
r0	o45	10
r1	c32	10
r1	r2	10
r1	r7	10
r1	c16	10
r2	r1	10
r2	r3	10
r2	r4	10
r2	r5	10
r2	s42	10
r2	c22	10
r3	r2	10
r3	r7	10
r3	c29	10
r3	r6	10
r3	s38	10
r4	s36	10
r4	c30	10
r4	r2	10
r4	r5	10
r4	c15	10
r5	r2	10
r5	r4	10
r5	r8	10
r5	c25	10
r5	s37	10
r6	r3	10
r6	r9	10
r6	r12	10

Component	Component	Bandwidth(Mbps)
r11	c19	10
r12	c33	10
r12	r10	10
r12	r6	10
r12	r13	10
r12	s39	10
r12	c24	10
r13	c27	10
r13	r12	10
r13	s41	10
r13	r14	10
r14	c34	10
r14	c31	10
r14	r13	10
r14	r9	10
r14	c20	10
c15	r4	10
c16	r1	10
c17	r9	10
c18	r6	10
c19	r11	10
c20	r14	10
c21	r8	10
c22	r2	10
c23	r10	10
c24	r12	10
c25	r5	10
c26	r7	10
c27	r13	10
c28	r8	10

r6	r10	10
r6	c18	10
r7	c26	10
r7	r1	10
r7	r10	10
r7	r3	10
r7	s35	10
r8	c28	10
r8	r5	10
r8	r11	10
r8	c21	10
r9	r6	10
r9	r11	10
r9	r14	10
r9	s44	10
r9	c17	10
r10	r7	10
r10	s43	10
r10	r12	10
r10	c23	10
r11	s40	10
r11	r8	10
r11	r9	10
c29	r3	10
c30	r7	10
c31	r4	10
c32	r5	10
c33	r3	10
c34	r12	10
s35	r7	10
s36	r4	10
s37	r5	10
s38	r3	10
s39	r12	10
s35	r7	10
s36	r4	10
s37	r5	10
s38	r3	10
s39	r12	10
s40	r11	10
s41	r13	10
s42	r2	10
s43	r10	10
s44	r9	10
o45	r0	10

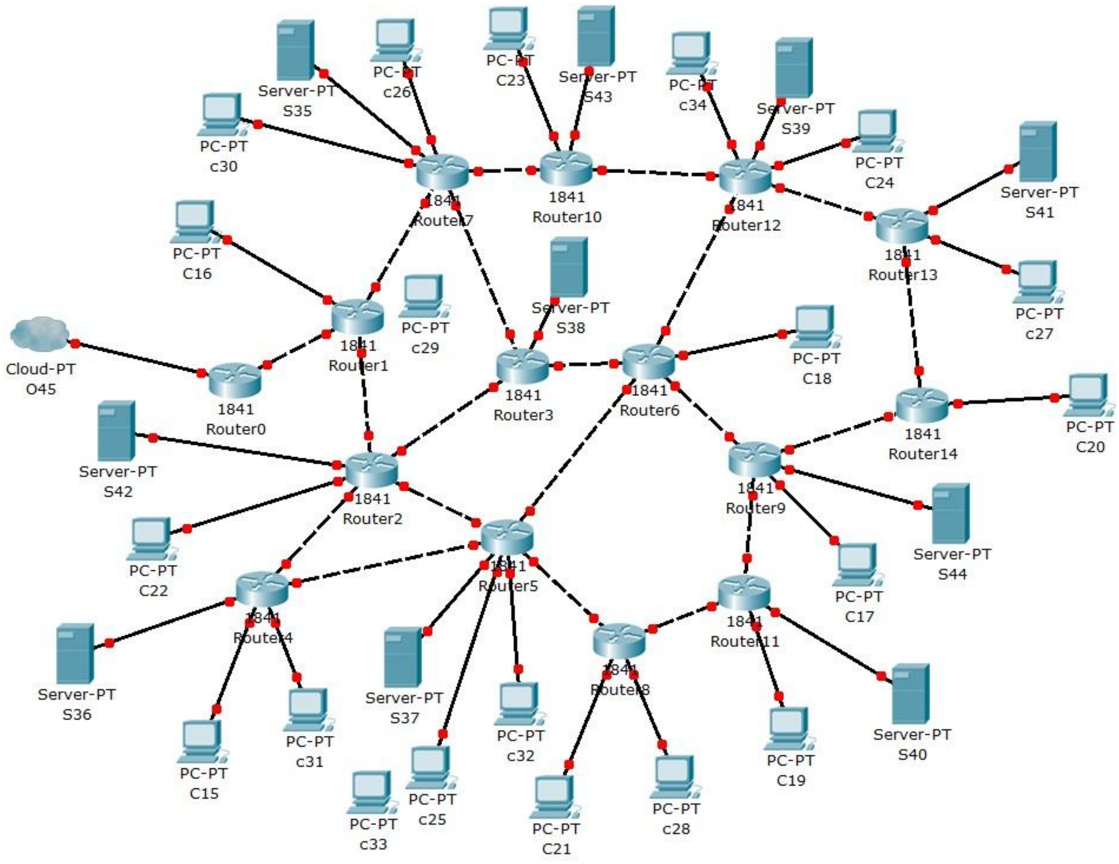


Figure 12: 10 Surrogates 20 Clients

Topology Table 13 as illustrated in Figure 13

Component	Component	Bandwidth(Mbps)
r0	r1	10
r0	o51	10
r1	c32	10
r1	r2	10
r1	s44	10
r1	c34	10
r2	r3	10
r2	r4	10
r2	r5	10
r2	s42	10
r2	c22	10
r3	r2	10
r3	r7	10
r3	c29	10
r3	r6	10
r3	s38	10
r4	s36	10
r4	c30	10
r4	r2	10
r4	r5	10
r4	c15	10
r5	r2	10
r5	r4	10
r5	r8	10
r5	c25	10
r5	s37	10
r6	r3	10
r6	r9	10
r6	r12	10
r6	r10	10

Component	Component	Bandwidth(Mbps)
r12	c19	10
r12	c33	10
r12	r10	10
r12	r6	10
r12	r13	10
r12	s39	10
r12	c24	10
r13	c27	10
r13	r12	10
r13	s41	10
r13	r14	10
r14	c34	10
r14	c31	10
r14	r13	10
r14	r9	10
r14	c20	10
c15	r4	10
c16	r1	10
c17	r9	10
c18	r6	10
c19	r11	10
c20	r14	10
c21	r8	10
c22	r2	10
c23	r10	10
c24	r12	10
c25	r5	10
c26	r7	10
c27	r13	10
c28	r8	10

r6	c18	10
r7	c26	10
r7	r1	10
r7	r10	10
r7	r3	10
r7	s35	10
r8	c28	10
r8	r5	10
r8	r11	10
r8	c21	10
r9	r6	10
r9	r11	10
r9	r14	10
r9	s44	10
r9	c17	10
r10	r7	10
r10	s43	10
r10	r12	10
r10	c23	10
r10	s40	10
r11	r8	10
r11	r9	10
r11	c23	10
r11	s40	10
r11	r8	10
r11	r9	10
r12	s42	10
r12	s36	10

c29	r3	10
c30	r4	10
c31	r14	10
c32	r1	10
c33	r12	10
c34	r14	10
c35	r7	10
c36	r4	10
c37	r5	10
c38	r3	10
c39	r12	10
c35	r7	10
s36	r4	10
s37	r5	10
s38	r3	10
s39	r12	10
s40	r11	10
s41	r13	10
s42	r2	10
s43	r10	10
s44	r9	10
s45	r0	10
s46	r13	10
s47	r9	10
s48	c20	10
s49	r4	10
s50	r1	10
o51	r0	10

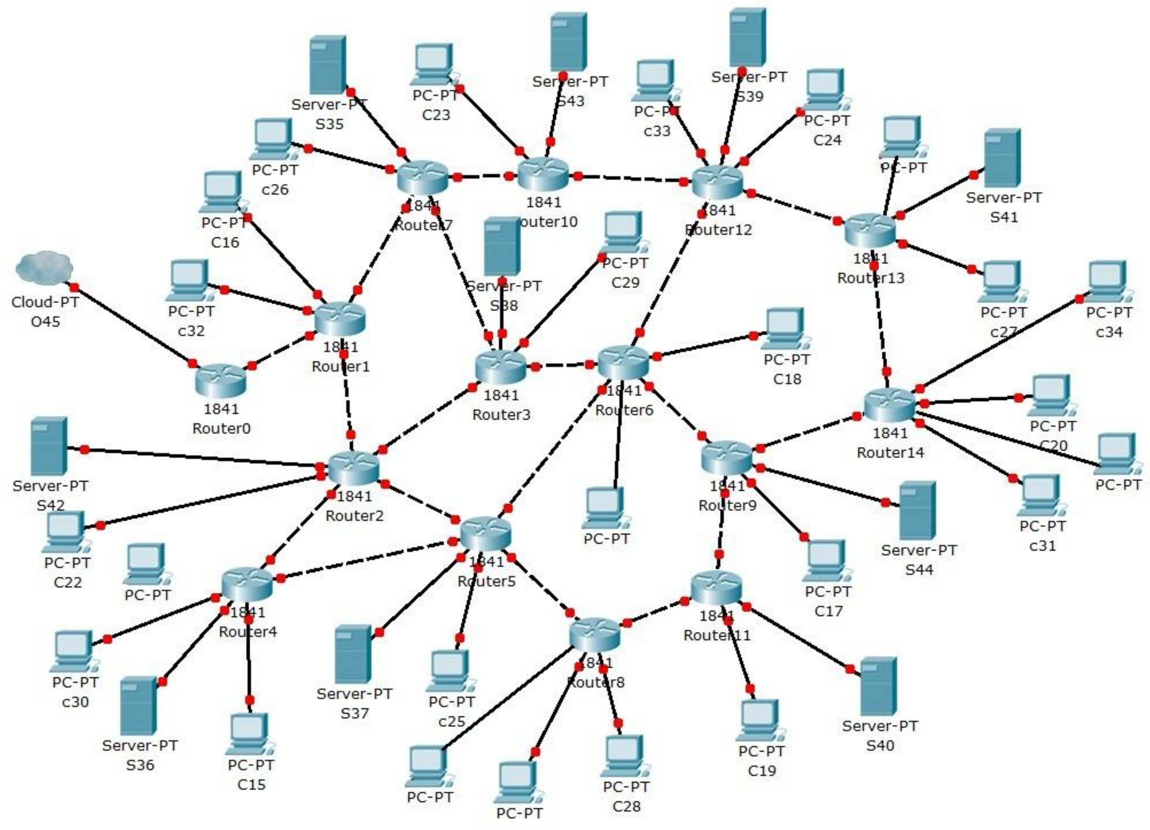


Figure 13: 10 Surrogates 25 Clients

Topology Table 14 as illustrated in Figure 14

Component	Component	Bandwidth(Mbps)
r0	r1	10
r0	o57	10
r1	c32	10
r1	r2	10
r1	s44	10
r1	c34	10
r1	s44	10
r1	c22	10
r2	r2	10
r2	r7	10
r2	c29	10
r2	r6	10
r2	s38	10
r2	s36	10
r2	c30	10
r2	r2	10
r2	r5	10
r3	c15	10
r3	r2	10
r3	c29	10
r3	r6	10
r3	s38	10
r4	s36	10
r4	c30	10
r4	r2	10
r4	r5	10
r4	c15	10
r5	r2	10
r5	r4	10
r5	r8	10

Component	Component	Bandwidth(Mbps)
r12	r13	10
r12	r9	10
r12	r10	10
r12	r6	10
r12	r13	10
r12	s39	10
r12	c24	10
r13	c27	10
r13	r12	10
r13	s41	10
r13	r14	10
r14	c34	10
r14	c31	10
r14	r13	10
r14	r9	10
r14	c20	10
c15	r4	10
c16	r1	10
c17	r9	10
c18	r6	10
c19	r11	10
c20	r14	10
c21	r8	10
c22	r2	10
c23	r10	10
c24	r12	10
c25	r5	10
c26	r7	10
c27	r13	10
c28	r8	10

r5	c25	10
r5	s37	10
r6	r3	10
r6	r9	10
r6	r12	10
r6	r10	10
r6	c18	10
r7	c26	10
r7	r1	10
r7	r10	10
r7	r3	10
r7	s35	10
r8	c28	10
r8	r5	10
r8	r11	10
r8	c21	10
r9	r6	10
r9	r11	10
r9	r14	10
r9	s44	10
r9	c17	10
r10	r7	10
r10	s43	10
r10	r12	10
r10	c23	10
r10	s40	10
r11	r8	10
r11	r9	10
r11	c23	10
r11	s40	10
r11	r8	10
r11	r9	10
r12	s42	10

c29	r3	10
c30	r4	10
c31	r14	10
c32	r1	10
c33	r12	10
c34	r14	10
c35	r7	10
c36	r4	10
c37	r5	10
c38	r3	10
c39	r12	10
c35	r7	10
c36	r4	10
c37	r5	10
c38	r3	10
c39	r12	10
c40	r11	10
s41	r13	10
s42	r2	10
s43	r10	10
s44	r9	10
s45	r0	10
s46	r13	10
s47	r9	10
s48	r2	10
s49	r4	10
s50	r1	10
s51	r9	10
s52	r9	10
s53	r6	10
s54	r11	10
s55	r14	10
s56	r8	10

r12	s36	10

o57	r0	10

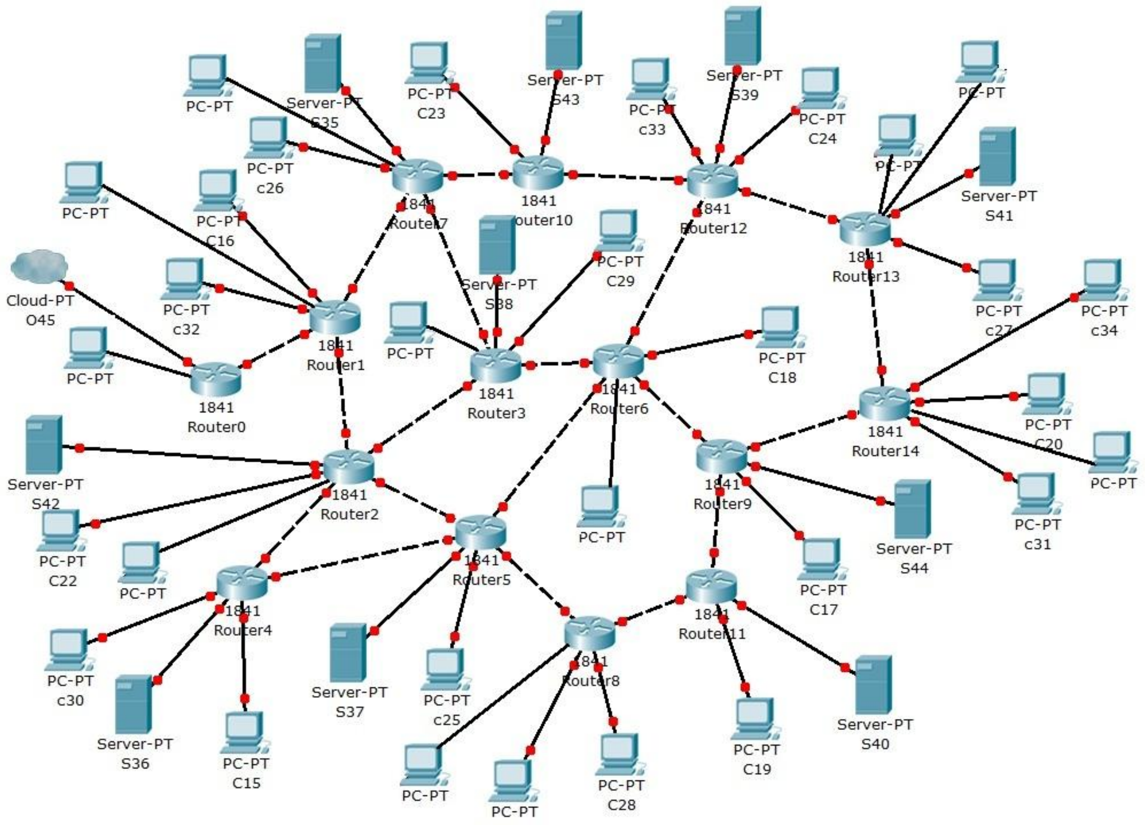


Figure 14: 10 Surrogates 30 Clients

<u>Topology Table 15 as illustrated in Figure 15</u>

Component	Component	Bandwidth(Mbps)
r0	r1	10
r0	o40	10
r1	c32	10
r1	r2	10
r1	r7	10
r1	c16	10
r2	r1	10
r2	r3	10
r2	r4	10
r2	r5	10
r2	c22	10
r3	r2	10
r3	r7	10
r3	c29	10
r3	r6	10
r3	s38	10
r4	s36	10
r4	c30	10
r4	r2	10
r4	r5	10
r4	c15	10
r5	r2	10
r5	r4	10
r5	r8	10
r5	c25	10
r5	s37	10
r6	r3	10
r6	r9	10
r6	r12	10
r6	r10	10

Component	Component	Bandwidth(Mbps)
r10	r12	10
r10	c23	10
r11	r8	10
r11	r9	10
r11	c19	10
r12	c33	10
r12	r10	10
r12	r6	10
r12	r13	10
r12	s39	10
r12	c24	10
r13	c27	10
r13	r12	10
r13	r14	10
r14	c34	10
r14	c31	10
r14	r13	10
r14	r9	10
r14	c20	10
c15	r4	10
c16	r1	10
c17	r9	10
c18	r6	10
c19	r11	10
c20	r14	10
c21	r8	10
c22	r2	10
c23	r10	10
c24	r12	10
c25	r5	10

r6	c18	10
r7	c26	10
r7	r1	10
r7	r10	10
r7	r3	10
r7	s35	10
r8	c28	10
r8	r5	10
r8	r11	10
r8	c21	10
r9	r6	10
r9	r11	10
r9	r14	10
r9	c17	10
r10	r7	10

c26	r7	10
c27	r13	10
c28	r8	10
c29	r3	10
c30	r4	10
c31	r14	10
c32	r1	10
c33	r12	10
c34	r14	10
s35	r7	10
s36	r4	10
s37	r5	10
s38	r3	10
s39	r12	10
o40	r0	10

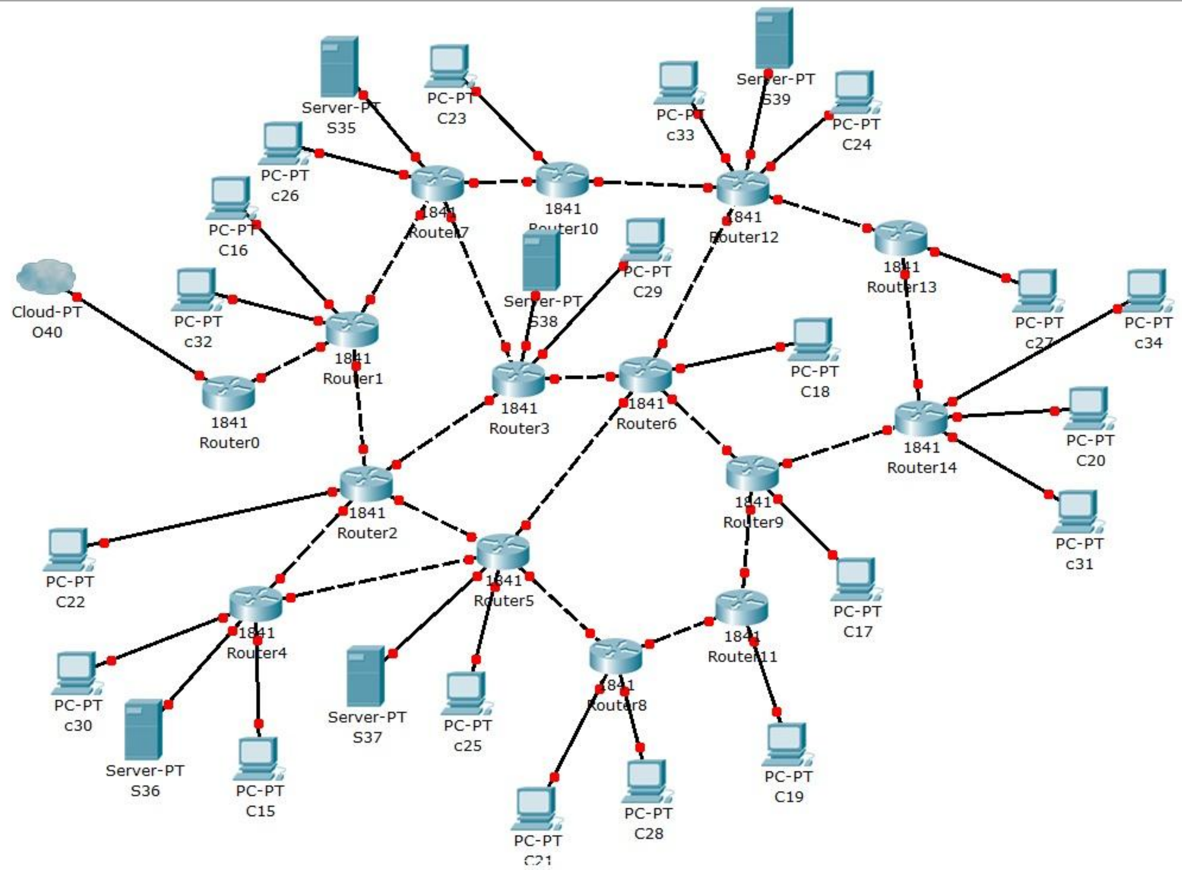


Figure 15: 20 Clients 5 Surrogates

Topology Table 16 as illustrated in Figure 16

Component	Component	Bandwidth(Mbps)
r0	r1	10
r0	o45	10
r1	c32	10
r1	r2	10
r1	r7	10
r1	c16	10
r2	r1	10
r2	r3	10
r2	r4	10
r2	r5	10
r2	s42	10
r2	c22	10
r3	r2	10
r3	r7	10
r3	c29	10
r3	r6	10
r3	s38	10
r4	s36	10
r4	c30	10
r4	r2	10
r4	r5	10
r4	c15	10
r5	r2	10
r5	r4	10
r5	r8	10
r5	c25	10
r5	s37	10
r6	r3	10
r6	r9	10
r6	r12	10

Component	Component	Bandwidth(Mbps)
r11	c19	10
r12	c33	10
r12	r10	10
r12	r6	10
r12	r13	10
r12	s39	10
r12	c24	10
r13	c27	10
r13	r12	10
r13	s41	10
r13	r14	10
r14	c34	10
r14	c31	10
r14	r13	10
r14	r9	10
r14	c20	10
c15	r4	10
c16	r1	10
c17	r9	10
c18	r6	10
c19	r11	10
c20	r14	10
c21	r8	10
c22	r2	10
c23	r10	10
c24	r12	10
c25	r5	10
c26	r7	10
c27	r13	10
c28	r8	10

r6	r10	10
r6	c18	10
r7	c26	10
r7	r1	10
r7	r10	10
r7	r3	10
r7	s35	10
r8	c28	10
r8	r5	10
r8	r11	10
r8	c21	10
r9	r6	10
r9	r11	10
r9	r14	10
r9	s44	10
r9	c17	10
r10	r7	10
r10	s43	10
r10	r12	10
r10	c23	10
r11	s40	10
r11	r8	10
r11	r9	10

c29	r3	10
c30	r4	10
c31	r14	10
c32	r1	10
c33	r12	10
c34	r14	10
s35	r7	10
s36	r4	10
s37	r5	10
s38	r3	10
s39	r12	10
s35	r7	10
s36	r4	10
s37	r5	10
s38	r3	10
s39	r12	10
s40	r11	10
s41	r13	10
s42	r2	10
s43	r10	10
s44	r9	10
o45	r0	10

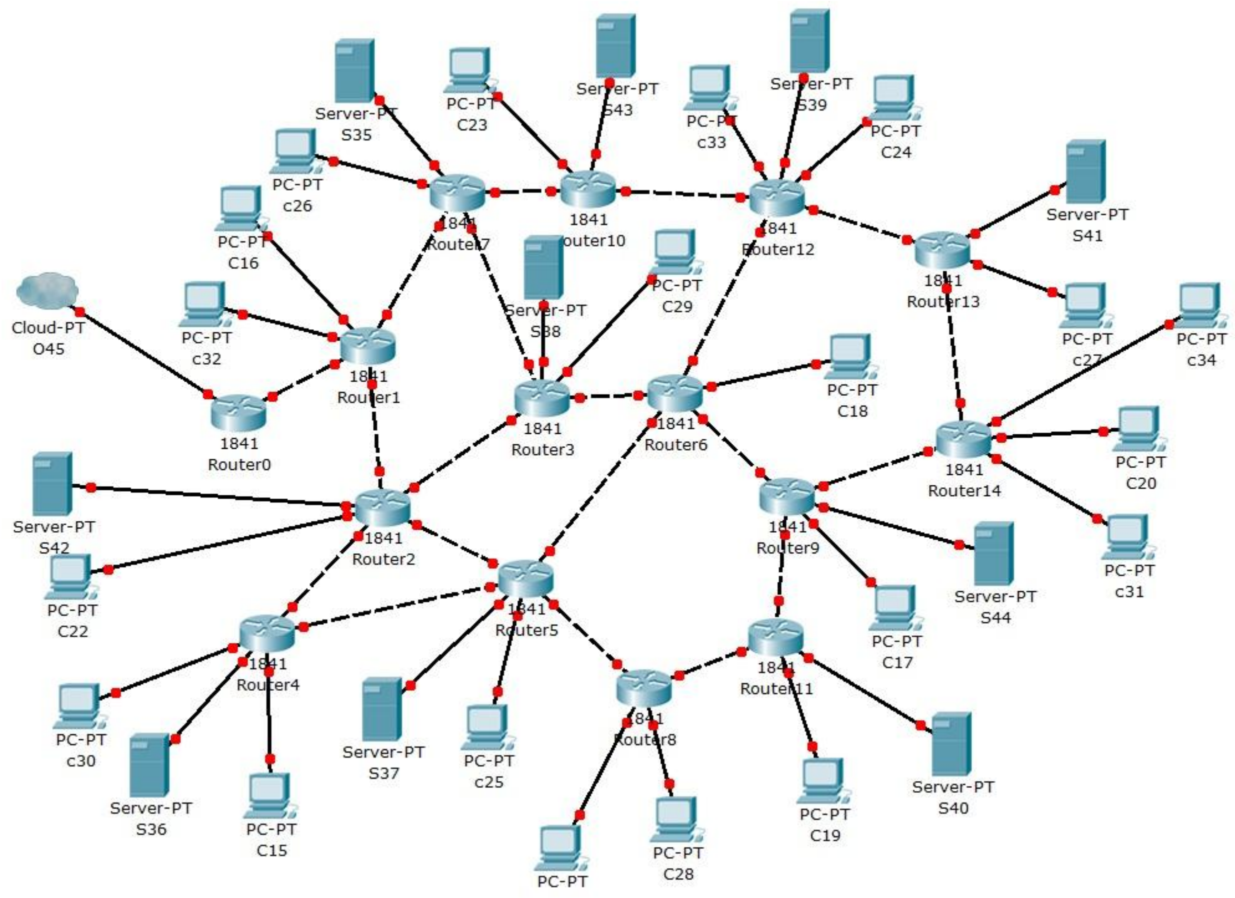


Figure 16: 20 Clients 10 Surrogates

<u>Topology Table 17 as illustrated in Figure 17</u>

Component	Component	Bandwidth(Mbps)
r0	r1	10
r0	s54	10
r0	o55	10
r1	s51	10
r1	c32	10
r1	r2	10
r1	r7	10
r1	c16	10
r2	r1	10
r2	r3	10
r2	r4	10
r2	r5	10
r2	s42	10
r2	c22	10
r3	r2	10
r3	r7	10
r3	c29	10
r3	r6	10
r3	s38	10
r4	s36	10
r4	c30	10
r4	s45	10
r4	r2	10
r4	r5	10
r4	c15	10
r5	r2	10
r5	r4	10
r5	r8	10
r5	c25	10
r5	s37	10

Component	Component	Bandwidth(Mbps)
r12	r6	10
r12	r13	10
r12	s39	10
r12	c24	10
r13	c27	10
r13	s50	10
r13	r12	10
r13	s41	10
r13	r14	10
r14	s47	10
r14	c34	10
r14	c31	10
r14	r13	10
r14	r9	10
r14	c20	10
c15	r4	10
c16	r1	10
c17	r9	10
c18	r6	10
c19	r11	10
c20	r14	10
c21	r8	10
c22	r2	10
c23	r10	10
c24	r12	10
c25	r5	10
c26	r7	10
c27	r13	10
c28	r8	10
c29	r3	10

r6	r3	10
r6	r9	10
r6	r12	10
r6	r10	10
r6	c18	10
r7	c26	10
r7	r1	10
r7	r10	10
r7	r3	10
r7	s35	10
r8	c28	10
r8	r5	10
r8	s53	10
r8	s46	10
r8	r11	10
r8	c21	10
r9	r6	10
r9	r11	10
r9	r14	10
r9	s44	10
r9	c17	10
r10	r7	10
r10	s43	10
r10	r12	10
r10	c23	10
r11	s40	10
r11	r8	10
r11	r9	10
r12	c33	10
r12	s52	10
r12	r10	10

c30	r4	10
c31	r14	10
c32	r1	10
c33	r12	10
c34	r14	10
s35	r7	10
s36	r4	10
s37	r5	10
s38	r3	10
s39	r12	10
s35	r7	10
s36	r4	10
s37	r5	10
s38	r3	10
s39	r12	10
s40	r11	10
s41	r13	10
s42	r2	10
s43	r10	10
s44	r9	10
s45	r4	10
s46	r8	10
s47	r14	10
s48	r12	10
s49	r14	10
s50	r13	10
s51	r1	10
s52	r12	10
s53	r8	10
s54	r0	10
o55	r0	10

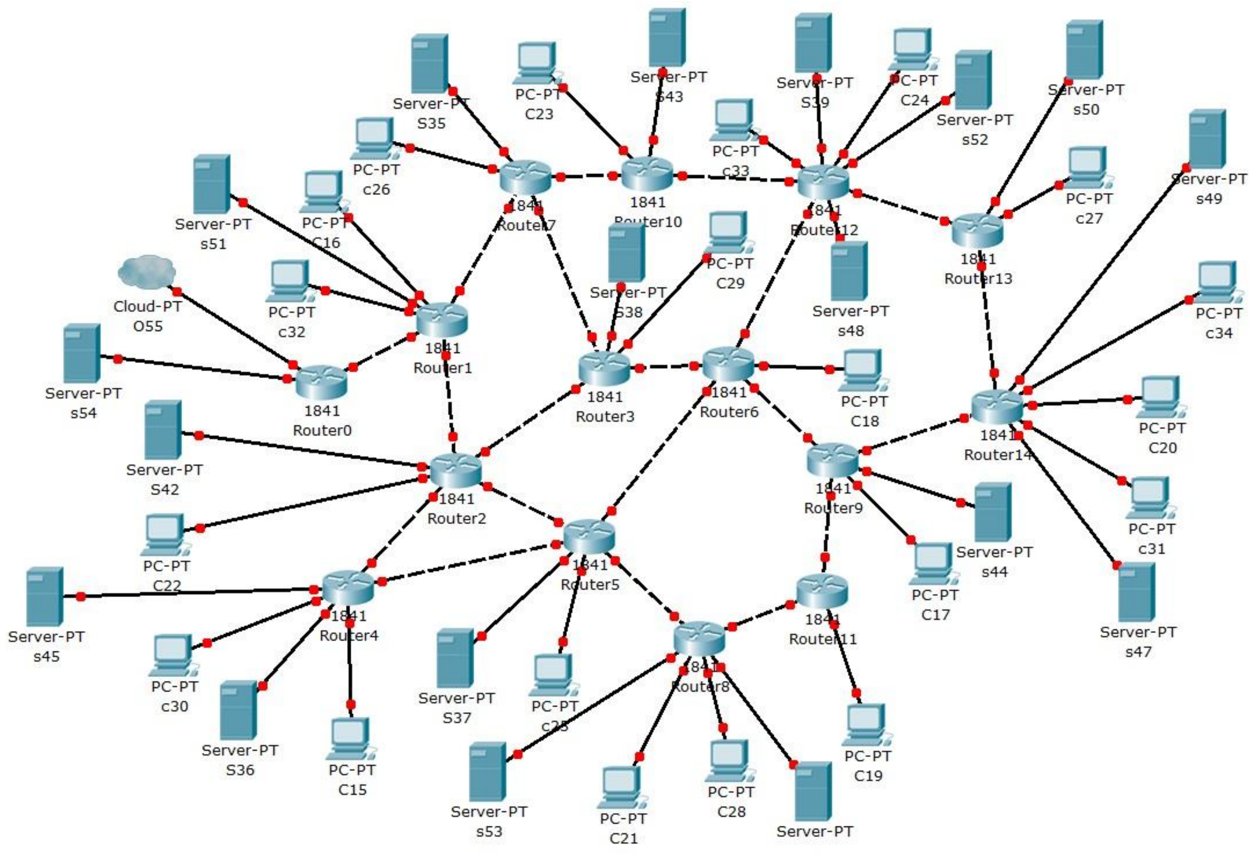


Figure 17: 20 Clients 20 Surrogates

<u>Table 18 as illustrated in Figure 18</u>

Component	Component	Bandwidth(Mbps)
r0	r1	10
r0	s54	10
r0	o55	10
r1	s51	10
r1	c32	10
r1	r2	10
r1	r7	10
r1	c16	10
r2	r1	10
r2	r3	10
r2	r4	10
r2	r5	10
r2	s42	10
r2	c22	10
r3	r2	10
r3	r7	10
r3	c29	10
r3	r6	10
r3	s38	10
r4	s36	10
r4	c30	10
r4	s45	10
r4	r2	10
r4	r5	10
r4	c15	10
r5	r2	10
r5	r4	10
r5	r8	10
r5	c25	10
r5	s37	10

Component	Component	Bandwidth(Mbps)
r12	r6	10
r12	r13	10
r12	s39	10
r12	c24	10
r13	c27	10
r13	s50	10
r13	r12	10
r13	s41	10
r13	r14	10
r14	s47	10
r14	c34	10
r14	c31	10
r14	r13	10
r14	r9	10
r14	c20	10
c15	r4	10
c16	r1	10
c17	r9	10
c18	r6	10
c19	r11	10
c20	r14	10
c21	r8	10
c22	r2	10
c23	r10	10
c24	r12	10
c25	r5	10
c26	r7	10
c27	r13	10
c28	r8	10
c29	r3	10

r6	r3	10
r6	r9	10
r6	r12	10
r6	r10	10
r6	c18	10
r7	c26	10
r7	r1	10
r7	r10	10
r7	r3	10
r7	s35	10
r8	c28	10
r8	r5	10
r8	s53	10
r8	s46	10
r8	r11	10
r8	c21	10
r9	r6	10
r9	r11	10
r9	r14	10
r9	s44	10
r9	c17	10
r9	r7	10
r10	s43	10
r10	r12	10
r10	c23	10
r10	s40	10
r10	s43	10
r10	r12	10
r10	c23	10
r11	s40	10
r11	r8	10
r11	r9	10
r12	c33	10

c30	r4	10
c31	r14	10
c32	r1	10
c33	r12	10
c34	r14	10
s35	r7	10
s36	r4	10
s37	r5	10
s38	r3	10
s39	r12	10
s35	r7	10
s36	r4	10
s37	r5	10
s38	r3	10
s39	r12	10
s40	r11	10
s41	r13	10
s42	r2	10
s43	r10	10
s44	r9	10
s45	r4	10
s46	r8	10
s47	r14	10
s48	r12	10
s49	r14	10
s50	r13	10
s51	r1	10
s52	r12	10
s53	r8	10
s54	r0	10
s55	r12	10
s56	r11	10
s57	r13	10

r12	s52	10
r12	r10	10

s58	r2	10
s59	r10	10
o60	r0	10

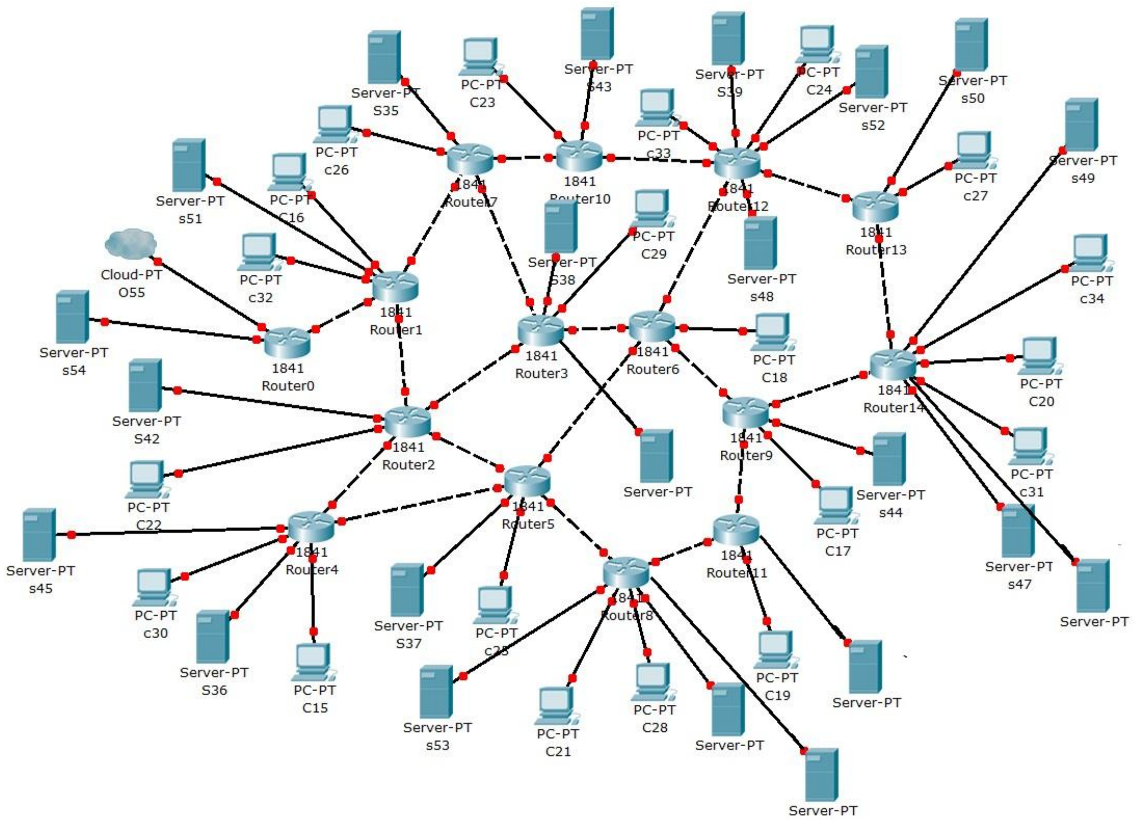


Figure 18: 20 Clients 25 Surrogates

Table 19 as illustrated in Figure 19

Component	Component	Bandwidth(Mbps)
r0	r1	10
r0	s54	10
r0	o55	10
r1	s51	10
r1	c32	10
r1	r2	10
r1	r7	10
r1	c16	10
r2	r1	10
r2	r3	10
r2	r4	10
r2	r5	10
r2	s42	10
r2	c22	10
r3	r2	10
r3	r7	10
r3	c29	10
r3	r6	10
r3	s38	10
r4	s36	10
r4	c30	10
r4	s45	10
r4	r2	10
r4	r5	10
r4	c15	10
r5	r2	10
r5	r4	10
r5	r8	10
r5	c25	10
r5	s37	10

Component	Component	Bandwidth(Mbps)
r12	s39	10
r12	c24	10
r13	c27	10
r13	s50	10
r13	r12	10
r13	s41	10
r13	r14	10
r14	s47	10
r14	c34	10
r14	c31	10
r14	r13	10
r14	r9	10
r14	c20	10
c15	r4	10
c16	r1	10
c17	r9	10
c18	r6	10
c19	r11	10
c20	r14	10
c21	r8	10
c22	r2	10
c23	r10	10
c24	r12	10
c25	r5	10
c26	r7	10
c27	r13	10
c28	r8	10
c29	r3	10
c30	r4	10
c31	r14	10

r6	r3	10
r6	r9	10
r6	r12	10
r6	r10	10
r6	c18	10
r7	c26	10
r7	r1	10
r7	r10	10
r7	r3	10
r7	s35	10
r7	c28	10
r7	r5	10
r8	s53	10
r8	s46	10
r8	r11	10
r8	c21	10
r8	r6	10
r8	r11	10
r8	r14	10
r9	s44	10
r9	c17	10
r9	r7	10
r9	s43	10
r9	r12	10
r9	c23	10
r9	s40	10
r10	s43	10
r10	r12	10
r10	c23	10
r10	s40	10
r11	s43	10
r11	r12	10
r11	c23	10

c32	r1	10
c33	r12	10
c34	r14	10
s35	r7	10
s36	r4	10
s37	r5	10
s38	r3	10
s39	r12	10
s35	r7	10
s36	r4	10
s37	r5	10
s38	r3	10
s39	r12	10
s40	r11	10
s41	r13	10
s42	r2	10
s43	r10	10
s44	r9	10
s45	r4	10
s46	r8	10
s47	r14	10
s48	r12	10
s49	r14	10
s50	r13	10
s51	r1	10
s52	r12	10
s53	r8	10
s54	r0	10
s55	r12	10
s56	r11	10
s57	r13	10
s58	r2	10
s59	r10	10

r11	s40	10
r11	r8	10
r11	r9	10
r12	c33	10
r12	s52	10
r12	r10	10
r12	r6	10
r12	r13	10

s60	r0	10
s61	r11	10
s62	r13	10
s63	r2	10
s64	r10	10
o65	r9	10
s64	r10	10
o65	r0	10

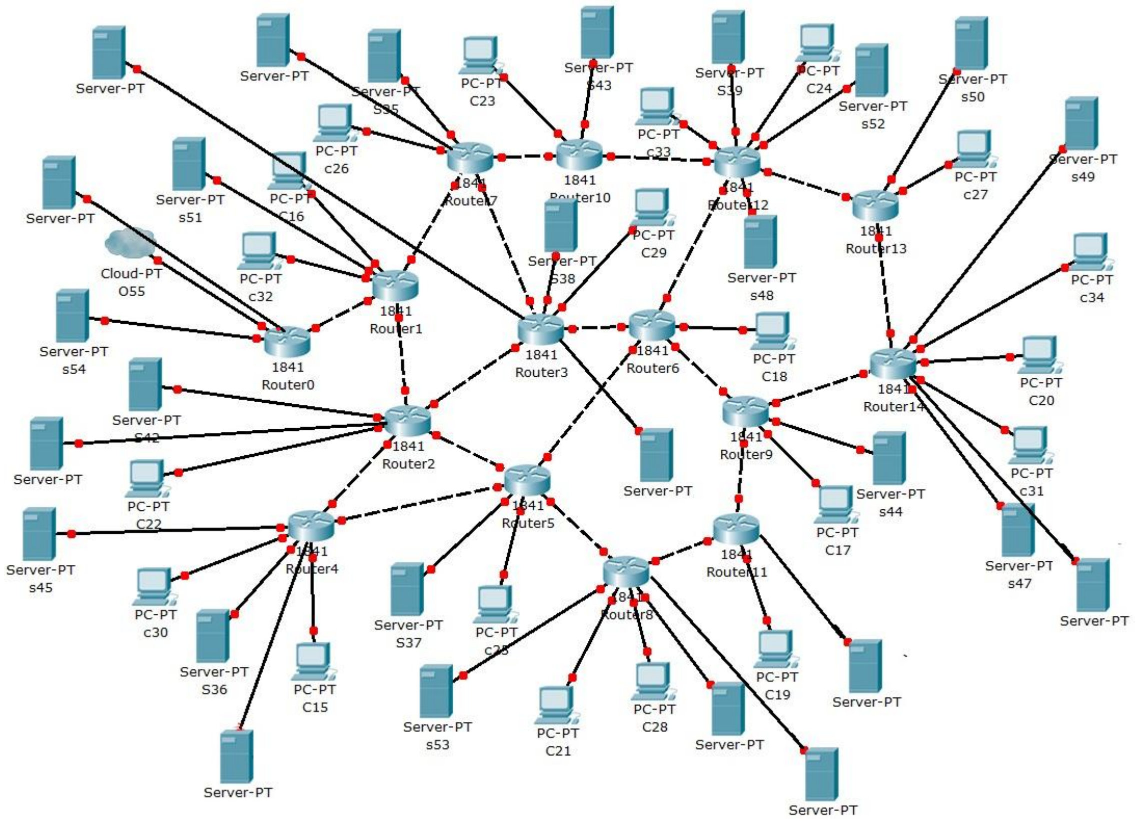


Figure 19: 20 Clients 30 Surrogates

Chapter 6: Discussion and Conclusion

6.1 Discussion

6.1.1 Simulations Scenario Comparison

Referring to Table 20, 21 and Figure 20, 21, for the 2 case studies, it was found that "Closest Surrogate" is better when compared side by side with "Random Surrogate". Both the "Closest Surrogate" and "Random Surrogate" is put to the test with the same models and while we are able to see that "Closest Surrogate" needs a bit of time to accommodate the increases in capacity, "Random Surrogate" works best only at 1:1 surrogate-client ratio. While in the perfect world, for every number of clients we will have a surrogate accommodating its requests, it is impractical in real life. Hence, for anyone to expect that we are able to have a 1:1 surrogate-client ratio in real life is impossible and that we can only expect that a CDN uses "Closest Surrogate" protocols as it is clearly seen that, despite needing some time to get used to an increase in capacity, it generally is able to recover and perform just as well as a 1:1 surrogate-client ratio.

Number of Client	Closest Surrogate	Random Surrogate
10	61.4839	64.0788
15	51.8004	56.0951
20	57.9785	55.9788
25	56.6786	54.9965
30	56.4982	49.9604

Table 20: Throughput comparisons between Closest and Random Surrogate Models (Surrogate Fixed)

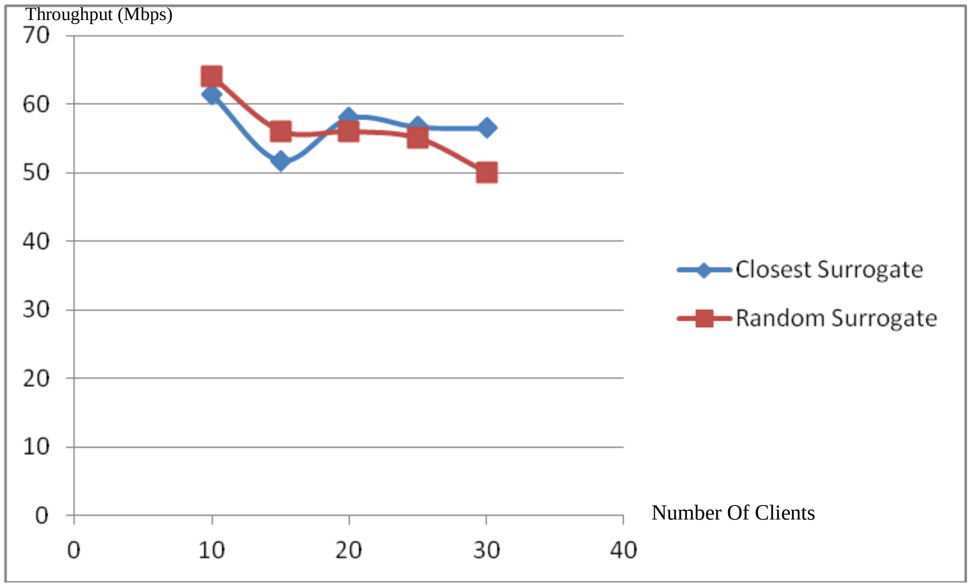


Figure 20: Graphical Interpretation Of Table 17

Number Of Surrogates	Closest Surrogate	Random Surrogate
5	57.9324	48.1495
10	57.9785	55.9788
20	61.2917	62.0368
25	62.7338	62.4888
30	64.0003	64.2012

Table 21: Throughput comparisons between Closest and Random Surrogate Models (Client Fixed)

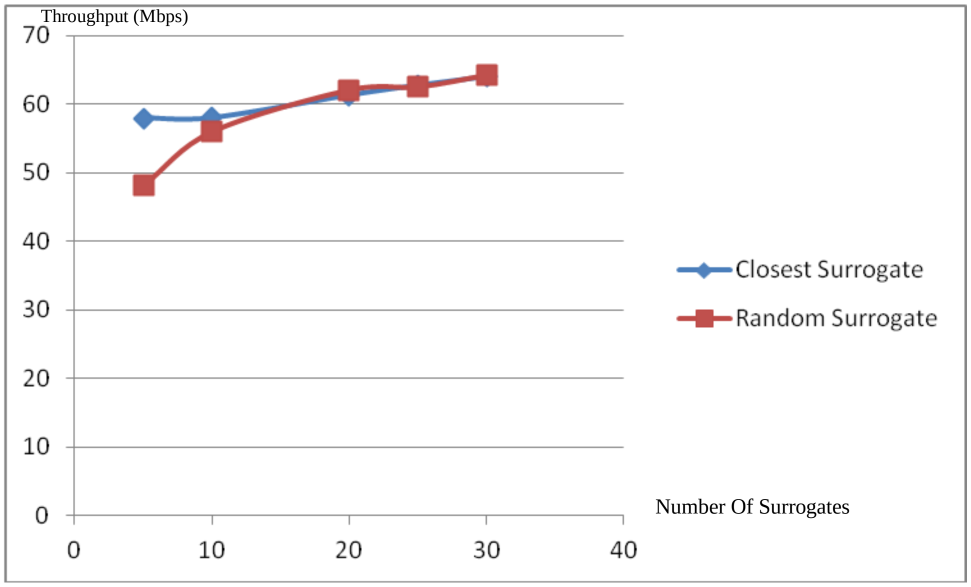


Figure 21: Graphical Interpretation Of Table 18

6.2 Conclusion and Future Works

Existing CDNs have always been struggling to use the perfect replica placement mechanism to fully expose the abilities of CDN:- to provide ON-DEMAND content with no jitters, no delays and at best, no frustrations from all the waiting. Akamai has been very successful in doing so and doing it on a large scale all over the world. However, the technology that they have been implementing has never been revealed nor shared to the general public in the hopes of monopolizing the CDN market. Throughout the years, numerous men have came up and proposed numerous different approaches to try and solve the issue but none has really been able to hit the mark and say they've finally done it. And then, there is *Soarin*. The concept of *Soarin* is rather similar to what has been seen with Akamai's technology. Though we may never be able to fully be sure that it is similar to what Akamai's been using, we make do with what we've got and *So*arin looks a rather good fit for the vague description that fits Akamai's technology. The biggest stumbling block however, is that *Soarin* has never been put to the test and be proven to be a good concept to work with. This is where the objective of this project comes in. This project was started to delve into *Soarin* and find out how it works. We would then simulate the concepts, analyze the results and go on to propose an extended-*Soarin* for a new scenario. While we have understood that using the “Closest Surrogate” protocols works best in a CDN, we are still far from fully able to utilize *Soarin*. To be able to one day put *Soarin* to real life practices, it has to go through many studies and simulation before it fully qualifies as a proper model to Akamai’s unrevealed technology. For future works in terms of an extended-*Soarin*, one should to go full scale and not just simulate and actually use a proper physical Observation Server to monitor the uplinks, downlinks, network traffic, throughput etc etc of a CDN, and a more in-depth study done on how to improve “Closest Surrogate” (maybe of not using network hops as reference and using another metric).

References

[1] M. Hofmann, and L. R. Beaumont, *Content Networking: Architecture, Protocols, and Practice*, Morgan Kaufmann Publishers, San Francisco, CA, USA, pp. 129-134, 2005.

[2] S. Adler, “The Slash Dot Effect: An Analysis of Three Internet Publications,” *Linux Gazette Issue*, Vol. 38, 1999.

[3] M. Day, B. Cain, G. Tomlinson, and P. Rzewski, “A Model for Content Internetworking (CDI),” Internet Engineering Task Force RFC 3466, February 2003. www.ietf.org/rfc/rfc3466.txt

[4] M. Hofmann, and L. R. Beaumont, *Content Networking: Architecture, Protocols, and Practice*, Morgan Kaufmann Publishers, San Francisco, CA, USA, pp. 129-134, 2005.

[5] N. Bartolini, E. Casalicchio, and S. Tucci, “A Walk Through Content Delivery Networks,” In *Proceedings of MASCOTS 2003*, LNCS Vol. 2965/2004, pp. 1-25, April 2004.

[6] J. Dilley, B. Maggs, J. Parikh, H. Prokop, R. Sitaraman, and B. Weihl, “Globally Distributed Content Delivery,” *IEEE Internet Computing*, pp. 50-58, September/October 2002.

[7] Akamai Technologies, Inc., www.akamai.com, 2007

[8] Z. Xu, Y. Hu,L. Bhuyan ,”Efficient Server Cooperation Mechanism in Content Delivery Network”, 2006

[9] Y. Chen, R.H. Katz, J.D. Kubiatowicz, “Dynamic Replica Placement for Scalable Content Delivery”, 2008

[10] Y. Kamiya, T. Shimokawa, F. Tanizaki, N. Yoshida. “Scalable Contents Delivery System with Dynamic Server Deployment”, 2010

Appendices

FINAL YEAR PROJECT BIWEEKLY REPORT

(Project I / Project II)

Trimester,Year: Year3T2	**Study week no.:** Week 2
Student Name & ID : WONG KHAI HSIANG (1001293)	
**Supervisor : ** Dr.LAU PHOOI YEE	
Project Title : Using Surrogate Servers for Content Delivery Network Infrastructure with Guaranteed QoS	

WORK DONE Discuss simulation to be done
WORK TO BE DONE Research on suitable simulation software
PROBLEMS ENCOUNTERED Several simulation software that needs to be studied
SELF EVALUATION OF THE PROGRESS Satisfactory progress

Supervisor's signature

Student's signature

FINAL YEAR PROJECT BIWEEKLY REPORT

(Project I / Project II)

Trimester,Year: Year3T2	**Study week no.:** Week 4
Student Name & ID : WONG KHAI HSIANG (1001293)	
**Supervisor : ** Dr.LAU PHOOI YEE	
Project Title : Using Surrogate Servers for Content Delivery Network Infrastructure with Guaranteed QoS	

WORK DONE Decision on simulation software
2.WORK TO BE DONE Further study to be done on selected simulation software
PROBLEMS ENCOUNTERED Not much documentation to be studied of Simulation Software
SELF EVALUATION OF THE PROGRESS Satisfactory progress

Supervisor's signature

Student's signature

FINAL YEAR PROJECT BIWEEKLY REPORT

(Project I / Project II)

Trimester,Year: Year3T2	**Study week no.:** Week 6
Student Name & ID : WONG KHAI HSIANG (1001293)	
**Supervisor : ** Dr.LAU PHOOI YEE	
Project Title : Using Surrogate Servers for Content Delivery Network Infrastructure with Guaranteed QoS	

WORK DONE Chap 4 Direction
2.WORK TO BE DONE Decision on simulation scope to be documented
PROBLEMS ENCOUNTERED Still exploring of Simulation Software
SELF EVALUATION OF THE PROGRESS Satisfactory progress

Supervisor’s signature

Student’s signature

FINAL YEAR PROJECT BIWEEKLY REPORT

(Project I / Project II)

Trimester,Year: Year3T2	**Study week no.:** Week 7
Student Name & ID : WONG KHAI HSIANG (1001293)	
Supervisor **:** Dr.LAU PHOOI YEE	
Project Title **:** Using Surrogate Servers for Content Delivery Network Infrastructure with Guaranteed QoS	

WORK DONE Chap 4 Progress
2.WORK TO BE DONE Completion of Chap 4 and Start of simulation
3.PROBLEMS ENCOUNTERED Unsure of all the simulation software's parameters
SELF EVALUATION OF THE PROGRESS Satisfactory progress

Supervisor's signature

Student's signature

FINAL YEAR PROJECT BIWEEKLY REPORT

(Project I / Project II)

Trimester,Year: Year3T2	**Study week no.:** Week 9
Student Name & ID : WONG KHAI HSIANG (1001293)	
**Supervisor : ** Dr.LAU PHOOI YEE	
Project Title : Using Surrogate Servers for Content Delivery Network Infrastructure with Guaranteed QoS	

WORK DONE Review of Chap 4 and Review of Preliminary Simulation Case
2.WORK TO BE DONE Coming up with proper models for Simulation
3.PROBLEMS ENCOUNTERED Unsure of what models to use
SELF EVALUATION OF THE PROGRESS Satisfactory progress

Supervisor’s signature

Student’s signature

FINAL YEAR PROJECT BIWEEKLY REPORT

(Project I / Project II)

Trimester,Year: Year3T2	**Study week no.:** Week 11
Student Name & ID : WONG KHAI HSIANG (1001293)	
**Supervisor : ** Dr.LAU PHOOI YEE	
Project Title : Using Surrogate Servers for Content Delivery Network Infrastructure with Guaranteed QoS	

WORK DONE Narrowed down area of simulation and model
2.WORK TO BE DONE Simulation and Documentation
3.PROBLEMS ENCOUNTERED Huge amounts of data to be documented.
4.SELF EVALUATION OF THE PROGRESS Satisfactory progress

Supervisor’s signature

Student’s signature