Preliminary Results on an Inverted Capacity Network Simulation

W.A.Vanhonacker[1]
Centre for Advanced Internet Architectures. Technical Report 040227B
Swinburne University of Technology
Melbourne, Australia

***Abstract*- We report the results of several experiments that were performed to evaluate the benefits and draw backs of an inverted capacity network. The results were obtained from a tool created in our lab: Simple Web Inverted System Simulation, SWISS. This tool allows us to simulate a web page request on any kind of network architecture and compute the download time of the requested web page. This tool is based on a module called Dummynet. The download time is computed from an extended wget command.**

Keywords- Simulation, inverted capacity network, download time.

I. Introduction

This technical report outlines the results gathered by several test conducted at the Center for Advanced Internet Architecture (CAIA) to define the characteristics and benefits of an inverted capacity network. In such a network, the actual low-bandwidth last-mile becomes a high bandwidth service in the order of multi-megabits or even gigabits per seconds. The current ratio of edge to core bandwidth will be inverted. This highly increased access network allows any customer with sufficient storage capacity to act as a possible content cache for other nearby neighborhoods. Being able to push the content close to the user obviously brings a relevant decrease in download time. The aim of our experiments is to measure this decrease and thus 'measure' the benefits of this type of architecture.

This report will show the influence of the web page content and the delay on the download time. How people, such as web designers, can decrease content download time by a good factor when structuring the web content the right way.

The Simple Web Inverted System Simulator (SWISS) is the tool used to gather those results.

II. Tool description

A. Tool modules

The tool is based on two basic modules for network architecture and traffic analysis: Dummynet [1] and wget [2]. Dummynet is a FreeBSD kernel functionality providing configurable control on selected packets going through a 'link', such as delays, random loss or bandwidth. This will thus be our basic tool for network simulation adding delay to represent real distance between machines. The second module is wget which is a simple utility for non-interactive download of files from the web. Wget has been extended in order to output the actual total download time of the file. Wget acts like a Web client, but it doesn't follow the same procedure as in regular browsers, the connection protocol is different (wget can only keep one connection at a time, compared to most browsers who can keep more). We thus realize here that this utility will not be precise in the results, but in our application, the main concern was actually to first generate a tool who would perform the simulation of the download action, ultimately, we can improve the tool by implementing a more precise computing of download time. Once the tool is operational, another utility could be then developed that will actually sense the traffic and analyse by itself the download times and so on.

In order to compare the traditional architecture that we use nowadays and an inverted capacity network, we needed a simple tool that will send a web page request and gather the difference of download time at the output between the two possible architecture. We didn't want to use to much hardware, so the network had to be mostly virtual.

B. Network architecture

The tool had to allow us to simulate any kind of architecture, especially allowing liberty in the choice of the distance between the client (who will request a web page) and the server. Our main interest is to compare an inverted capacity network to a conventional network. The traditional network basically involves a client, a server and also a cache that will be used when the document or one of its objects is cachable. The delay between those three machines is going to be configurable. The inverted network also involves the same architecture, but here the cache is going to be as close as possible to the client, effectively at a 0ms delay.

The basic testbed architecture is shown in Figure 1.

The hardware used in our tests are as follows:

- The client is a PC with IP address 136.186.229.99. It is a Compaq EV) 500 P4 1.6GHz 256MB RAM running FreeBSD 4.8 with an Intel Pro/100 Ethernet card.
- The bridge is a PC with IP address 136.186.229.72. It is a Compaq EV) 500 P4 1.6GHz 256MB RAM

1 This work was performed while working for Swinburne University of Technology. Wendy Vanhonacker can be contacted at wendy@vanhonacker.ch

running FreeBSD 4.8 with an Intel Pro/100 Ethernet card.

- The servers are virtual hosts set up on one PC. The PC is a mini ITX motherboard VIA C3 Samuel 2, 533.36MHz 256MB RAM running FreeBSD 4.8 with a VIA VT6102 Rhine II 10/100BaseTX. This PC is using the Jail Host Toolkit (JHT) [3] in order to have three virtual hosts (jails) which each have their own IP address:
  - 136.186.229.213 for the main remote server,
  - 136.186.229.212 for the cache used in real network simulation,
  - 136.186.229.211 for the cache used in inverted capacity network simulation.

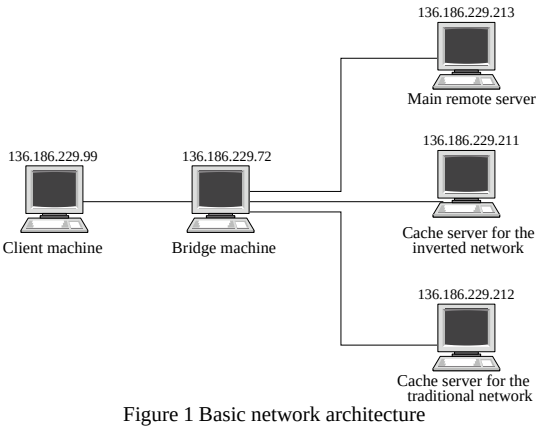

Figure 1 Basic network architecture

Dummynet is installed on the bridge [4]. It controls the delays between the client, and the server, and between the client and the caches (one for regular simulation, and one for inverted network where the delay is always set to 0).

C. Downloaded page contents

In order to see the influence of the web page contents in the download time, the user can only choose a web page from a list of predefined ones or can create their own. Once the page is created and copied to the server, the SWISS tool simulates the download request and compute the download time. The download time is mainly influenced by two factors: the delay which is taken care by dummynet and the web page being downloaded. The content of the page and some other characteristics such as its cachability or its size may increase or decrease the download time. We thus need in our simulation to allow the user to choose between a multitude of possible documents so that he can see the influence each feature has on the simulation.

The list of predefined pages is meant to be representative to the real web contents of todays Internet. In order to define this set, we analyzed in a parallel project [5] the contents of a big set of web pages from the Internet and defined the main characteristics of a typical web page. Out of those results, we created a representative set of pages which is the list of predefined pages for our tool. We categorized web pages into 8 different classes. We then created a sample of each classes so that it represents a global image of all web pages.

So the user of the tool has to choose between one of the classes of web page he wants to download. He also needs to choose the number of objects, the number of cachable objects (if there are any) and the size of the objects.

We didn't want to let the user to much free in his decisions so we imposed a choice of 5 or 30 objects only (or 0 of course). From an earlier research [5], we found that more than 80% of objects on a web page are cachable in average. Thus the number of cachable objects was reduced to 0, 4 or 5 if the total number of objects is 5 and 0, 24 or 30 if the total was 30 (4 is 80% of 5 and 24 is 80% of 30).

The user also has to choose the order of the objects (ordered, reversed or mixed), since the sequence of the objects also influences the download time. The order are defined like this:

1. ordered: the non-cachable objects come first then the cachable ones
2. reversed: the cachable ones come first then the non-cachable
3. mixed: no particular order, cachable and non-cachable objects are mixed.

D. The SWISS GUI

A GUI has been implemented in order to facilitate the user experience. Figure 2 and Figure 3 show snapshots of the tool.

Figure 2 is a snapshot of the first part of the setup: the population setup. In order to simulate a download of a page, the tool first needs to actually create those pages. Before the simulation, the use has to define a few set of pages; the tool will create them, and send them on the servers. This is the opportunity for the user to actually decide what kind of documents he wants to simulate, since the structure of a document influences the download time (See ch IV.C).

To create a page the user needs to define the number of objects in the document, their size, their cachability, the sequence in which the pages will be displayed (what order will have the cachable objects and the non-cachable objects) and the cachability of the document also.

Figure 3 is the second part of the setup: the simulation setup. This is where the user will calibrate the parameters for the download process. The upper part of the GUI defines the simulation parameters and the lower part defines the web page the user want to download. He can choose one or several pages to download all in one. The user can only choose the pages he has created in the first part, at the population setup.

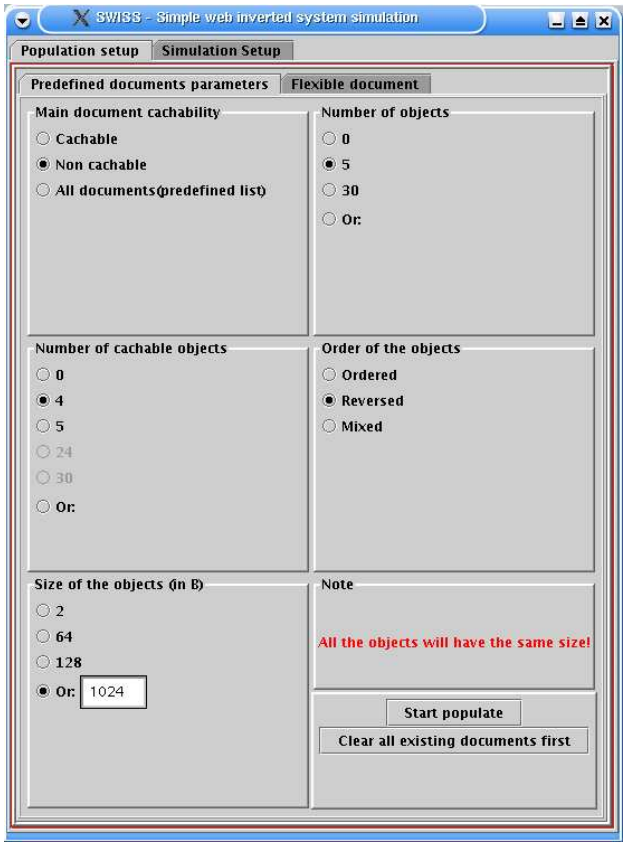

Figure 2 Snapshot of the tool : Population part

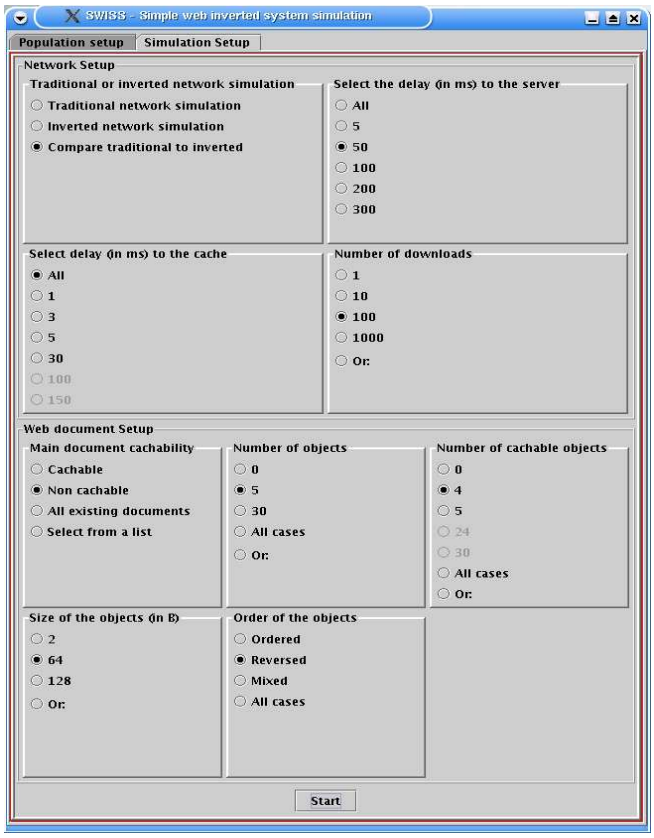

Figure 3 Snapshot of the tool : Simulation part

III. Results and Analysis

We are now going through the results of our simulations. There are a few different parameters the user can choose before starting the simulation. Those parameters such as delay or size of the objects in the web page will greatly influence the results of the download time. Some of the influences are obvious, but other might not have been so clear.

A. The download process

In order to understand how the download time is computed, we need to know how a web page is actually downloaded from any server. This process depends a lot on the utility used: the most common one is our web browser, Netscape, Mozilla, etc... In our case we use wget which is a simple tool called from the command prompt. The most useful property is that wget can act as a web robot (also called crawlers or spiders) by downloading recursively all the contents of a web server (with the '-r' argument). Because of this ability to steal everything, various robot exclusion schemes have been devised as a means for the server administrators and document authors to protect chosen portions of their sites. The more popular mechanism is the Robots Exclusion Standard, or RES. It specifies the format of a text file containing directives that instruct the robots which URL paths to avoid. So when a recursive download is requested, the server sends back to the client this robots.txt file, if the clients asks for it. Since most of the web browsers and other search engines use this robots.txt, we forced wget to request this robots.txt.

Figure 4 shows in details how a non-cachable page with 5 objects, 4 cachable in ordered and mixed order is transferred to the client. An ordered sequence of objects means that in the document all the non-cachable objects come first then the cachable ones. A mixed sequence is when cachable and non-cachable objects don't have any particular order. Note that any cachable object or document will be stored in a cache server, thus its' download is always shorter than if it wasn't cachable and on the remote server (since the cache is always closer to the client than the remote server).

1. First the client connects to the server and sends a request with the name of the html page to be downloaded. The server sends back the main html page. It then reads through the document and finds the location of each objects: whether it is on the same server or if it has been cached on another closer server). It remembers the order of those locations.
2. Now, the client needs to get the robots.txt file from each server. It is going to connect to each server in the same order as the order the objects appear on the document. Thus the client connects to the server where the first object is (if it is the main remote server, it is reusing the old connection, if not, it opens a new connection after closing the other one.) and requests the robots.txt file. If it exists the server will send it to the client. It does this for each server. Note that in our simulation, none of the servers had a robots.txt file.

3. The client will then connect to the server again in the same order to request each file linked to the main document. Depending on the argument of wget, it will either only ask for the children of this document(which is all the objects needed to display the html page), or it will also request for the linked html pages. In our simulation we will only use the first case. That is,we want to know how long it takes to download on page with all its correspondent objects (images, etc..) only.

   When requesting the objects of the page, if one of these objects is an embedded object that is located maybe on another server, the client will close the connection to the main server and create a new connection to the second server and request/download the object. Then it will reconnect to the first server if it has other objects to download from this server.

4. All those connections and re-connections obviously add unwanted time to the process. This is why the web page structure is really important and can add a lot of unnecessary time if it is badly written.

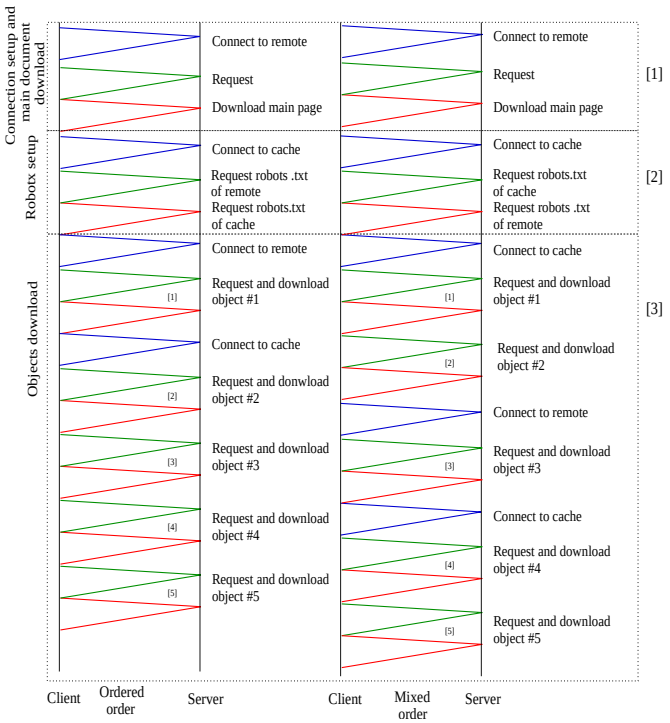

Figure 4 Wget sequence diagram when downloading a non-cachable document with 5 objects, 4 cachable in an ordered and mixed sequence

B. The delay's impact

The delay from the client to the servers has an obvious impact on the download time of any web page. In Figure 5, we simulated a document with the following characteristics:

i. Non-cachable document,

ii. 5 objects of same size (64B),

iii. All objects are cachable

The simulation shows the difference of time between an inverted capacity network and an actual network where the delay of the cache is varied from 1 to 150ms and the delay to the server is set to 200ms.

The first line is the download time of the inverted network and the other ones are the download times for a traditional network where the delay to the cache varies from 1 to 150ms.

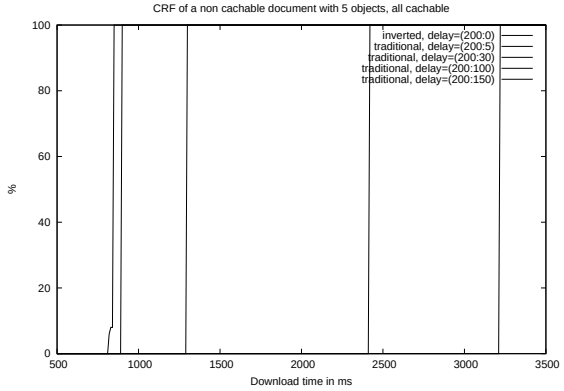

Figure 5 Cumulative relative frequency of the download time

In Figure 6, we simulated a document with the following characteristics:

iv. Non-cachable document,

v. 5 objects of same size (64B),

vi. 80% (4) of the objects are cachable,

vii. The order of those objects is reversed which means that the cachable objects are all coming first then the non cachable ones.

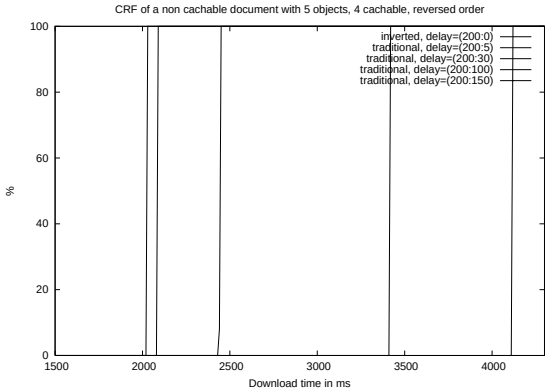

Figure 6 Cumulative relative frequency of the download time

As you can see, the download time for an inverted network is a bit shorter than in a real network. This difference occurs because the cache server is located at 1 to 150 ms for the real network case and at 0ms for the inverted network case. The conclusions are obviously that the inverted capacity network has a faster download process.

It is also interesting to view the actual time we gain from changing the network from traditional to inverted. Figure 7 and Figure 8 show the percentage of increase of time for the same document as in Figure 5 and Figure 6 respectively. For example in Figure 7, the download time is speed up by a factor of three on an inverted

network, compared to a traditional network with a delay to the cache server set to 100ms.

Note that when the client loads an object, it first needs to make a connection to the server, then a request, then finally downloads the page. Thus the download time and the delay are not a one to one correspondence.

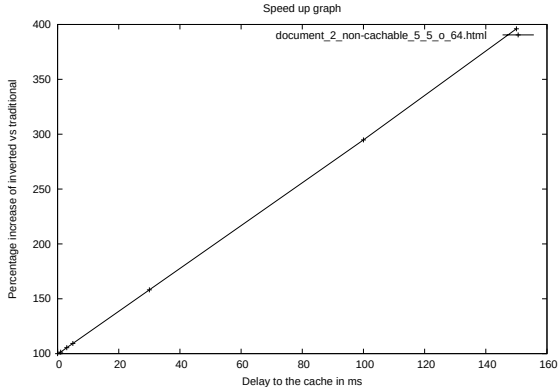

Figure 7 Percentage increase of inverted speed up versus traditionalfor a non-cachable document with 5 objects, all cachable

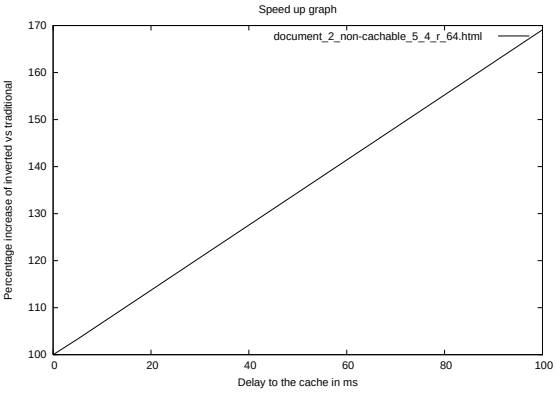

Figure 8 Percentage increase for a non-cachable document with 5 objects, 4 cachable, reverse order

Figure 9 shows roughly the total download time computation for the document simulated on Figure 5.

A = remote server , B = cache server

connection to A= c(A),

request for any file to A = r(A),

download of file from A = d(A,file),

d_time ~= c(A) + r(A) + d(A,main page) *[download main doc]*

+ c(B) + r(B) +r(A) *[get robots.txt]*

+ c(B) +5[r(B) + d(B,object)]* *[get objects]*

= c(A) + 2r(A) + d(A,main page) + 2c(B) + 5d(B,object) +6r(B)

Figure 9 Download time computation

The difference between the inverted network and the real network is thus:

B1 =real network cache server ,

B2 = inverted network cache server,

difference = [2c(B1) + 6r(B1) + 5d(B1,object)]

- [2c(B2) + 6r(B2) +5d(B2,object)]

Figure 10 Difference in the download time computation

The connection and request of an object depends linearly on the delay. The download time of an object depends of the delay but also from the file size.

C. The web content impact

As you can see in Figure 4, the order of the objects is relevant to the download time. The process between the ordered and mixed version is not the same, even if the number of objects is the same. This is due to the fact that the client has to connect to every new server it encounters and it doesn't keep the old connections. Thus, if the objects are not grouped per location, the client will have to reconnect twice to the same server.

Figure 11 shows the cumulative relative frequency of the download time of the same document as in Figure 6 but where we tested each possible order and at a delay of 200 ms to the remote server and 100ms to the cache server (0 ms if it is an inverted network):

4. ordered: the non-cachable objects first then the cachable ones
5. reversed: the cachable ones first then the non-cachable
6. mixed: no particular order, cachable and non-cachable are mixed.

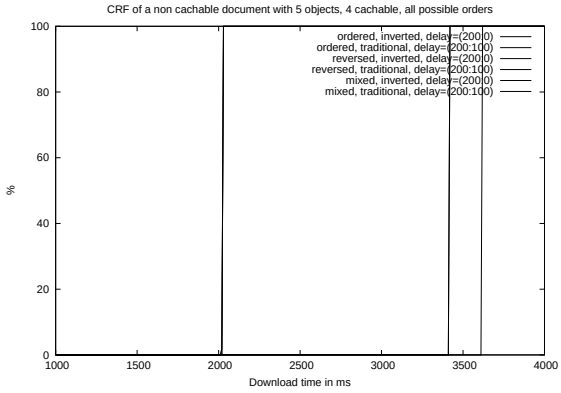

Figure 11 Difference in the download time computation

The ordered, reversed and mixed version of the inverted network simulation are all pretty close to each other (first line). The ordered and reversed version of a real network or also close (second line) and the mixed version has a longer download time (third line).

	Inverted network	Traditional network
Ordered	2021553.38	3417794.25
Reversed	2025243.88	3417845.25
Mixed	2020838.75	3616336.5

Table 1 Average download time in us

Table 1 shows the average download time in us for the same test. As you can see the ordered and reversed have almost the same time, the difference is of the order of 3-4ms for an inverted network and 50us for an traditional network. Why? Well if you have a look at

Figure 4, the reversed order will have exactly the same procedure but in a different order than in the ordered version. The number of connections to remote and cache are the same, and the requests and downloads too.

For the mixed case, in Figure 4 you can see that 2 cachable objects from the cache server are loaded, then one from the remote (non-cachable object) then the two last cachable objects from the cache. Thus you have one more connection to the cache than in the ordered version. In an inverted network, the connection to the cache is almost null since the delay to it is 0ms, but in the traditional network, the connection time to the cache is relevant since its delay is high (100ms). That is why the average download time of a mixed sequence document in an inverted network is almost the same than in an ordered sequence (4 ms difference which is irrelevant when the delay to the server is 200ms), and in a traditional network, the difference from ordered to mixed version is high (around 200 ms).

Figure 12 shows the speed-up gained from changing the network from traditional to inverted. It is interesting to note that the mixed version has a faster speed up than in other orders. From Figure 4, it is clear that the download of the mixed order page needs to do one more connection to the cache than in the other orders. Since in an inverted network this connection is almost null, the difference of download time is bigger, since in a traditional network, we will loose some time to connect to the cache.

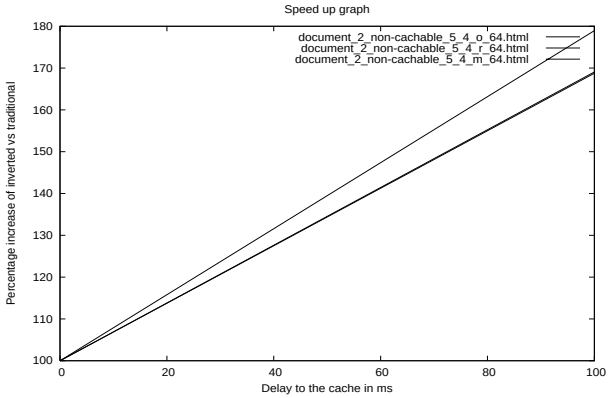

Figure 12 Difference in the download time computation

Thus, what have we learned from all this? First a good web authors and web masters should be aware of cachability. Caches can help your Web site to load faster, and gets rid of some traffic load on your server and Internet link. But from our results on the download tool (here we used wget), the order is also relevant. But this depends whether or not your tool keeps the connections alive (multiple connection alive) or not. We found that a mixed order looses a lot of time in the connection process, but if we implement an inverted network, this time lost will be irrelevant.

As a final plot, Figure 13 shows the speed up of the download time of a non-cachable document with 30 objects, and where the objects either are all cachable, all non-cachable or 80% cachable. The simulation is conducted with a delay of 50ms to the remote server and from 1 to 30ms to the cache server on the real network and of 0ms for the cache server on the inverted network.

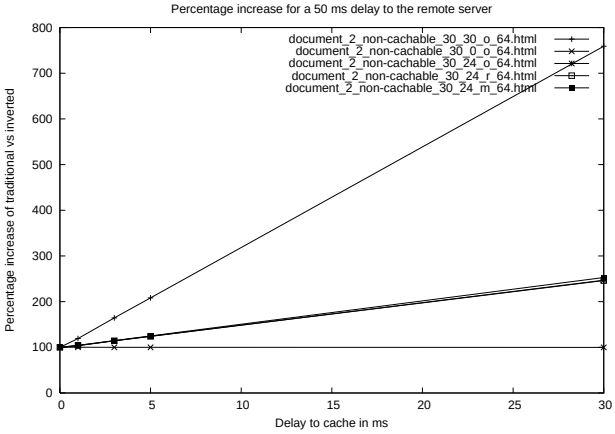

Figure 13 Cumulative relative frequency of the download time

This shows exactly what are the parameters who will be the most influenced by a change of network architecture. As you can see, a document with all its objects cachable will be more advantaged by an inverted capacity network (higher increase in speed) because all the objects will be closer to the client in an inverted network than in a traditional network. The document which will the less be influenced by a change will be a document with no cachable objects at all. All these objects would be stored on the remote server and a change of bandwidth from the client to the caches would not speed up the download at all.

IV. Download

The SWISS tool can be downloaded from the CAIA server [6].

V. Conclusion

We build a tool that simulates any kind of Internet architecture. The main goal of this tool is to simulate an inverted capacity network and compare it to the real network. The main concern here is the download time of a web page requested by a virtual client to a virtual server. The simulation also used the concept of caches and can simulate the caching process.

This report shows some preliminary results on several simulation, and mainly shows the capacities of the tool. Several different simulations can be done that are not shown in this report, but this is left to future users who want to simulate specific tasks.

An inverted capacity network has two main advantages: the first one is the decrease in download time which we can compute with the tool, and the second one is the decrease in traffic load because most of the web pages would be accessed directly from home' caches with a high storage capacity and thus a minimum of traffic will actually go through the outside network.

In a future work, the goal will be to add more specifications to the tool, so that we can also analyse the traffic load and simulate traffic burst and so on.

Acknowledgments

Special thanks to Lawrence Stewart without whom this report wouldn't have such wonderful plots.

References

[1] L.Rizzo "Dummynet", July 2002, http://info.iet.unipi.it/~luigi/ip_dummynet

[2] "GNU Wget", October 2003, http://www.gnu.org/software/wget/wget.html

[3] G.Armitage, "Maximising Student Exposure to Unix Networking using FreeBSD Virtual Hosts", Center for Advanced Internet Architecture, Swinburne University of Technology, March 2003, http://caia.swin.edu.au/reports/030320A/CAIA-TR-030320A.pdf

[4] W.Vanhonacker, "Evaluation of the FreeBSD dummynet network performance simulation tool on a Pentium 4-based Ethernet Bridge", Center for Advanced Internet Architecture, Swinburne University of Technology, December 2003, http://caia.swin.edu.au/reports/031202A/CAIA-TR-031202A.pdf

[5] W.Vanhonacker, "Characterizing Web Content", Center for Advanced Internet Architecture, Swinburne University of Technology, February 2004, http://caia.swin.edu.au/reports/040227A/CAIA-TR-040227A.pdf

[6] "Simple web inverted system simultion tool", Center for Advanced Internet Architecture, Swinburne University of Technology, February 2004, http://caia.swin.edu.au/ice/tools/swiss