A New Environment for

Web Caching and Replication Study

Víctor J. Sosa, Leandro Navarro
{vjsosa,leandro}@ac.upc.es
Universidad Politécnica de Cataluña (UPC)
Centro de Investigación en Computación (CIC)
Centro Nacional de Investigación y Desarrollo Tecnológico (Cenidet)

Abstract

Web Caching and replication have received considerable attention in the past years due to effectiveness in reducing client response time and network traffic. In this paper we describe a tool to improving the Web caching and replication study. An important difference with other tools is this tool try to take advantage of two tools very useful in caching and computer networks study, Proxycizer[Gadde98] and NS[NS] respectively. Proxycizer2ns is an approach to combine automatically these two tools, and in that way to improving our large scale web caching and replication study, taking the best of both parts, in one hand the facility of simulating an infrastructure which allow large scale caching and replication analisys, and in the other hand the verification of impact in the network that this simulated infrastructure have produced, rarely presented by other tool in such a didactic and complete way.

Keywords: Caching, Replication, Internet, Network, Simulation.

1. Introduction.

Nowadays Web is an important component in the global information infrastructure because of its explosive growth and use. In that situation Web becomes the first traffic generator in the Internet. As a consequence more frequently Web's users suffer long delayes when try to get a document. This necessity has emphasized the importance of the study and the analysis of the behavior of large scale caching and replication.

Some large scale Web caching and replication strategies have been propose to reduce this problem, trying to optimizes the bandwith use in the internet [Chankhun95][Gadde98b], nevertheless it is very difficult to perceive the impact that can have in a network environment with certain characteristics until does start the proposal in a real network environment, which has the disadvantage of most of times we cannot use our network equipment in march to make those tests because the problems that these can cause to users or the technical effort that this entails. Therefore, most of the proposals of architectures of large scale caching and replication come accompanied by simulation tests that indicate the effectiveness of one or another one. However in most of the cases these tests do not contemplate the characteristics of the network in which they are made, but that they are limited to driven simulation environment with the data that demonstrate their numerical improvements, this is, how good is the global cache hit ratio, how they decrease the number of messages interchanged in the network (for control cache protocol), how efficient is the management of a global cache directory, etc.

In many tools which do simulation like described above is difficult to find a way to

see the behaviour of traffic in some links, or bandwidth consumption, and latency, in other words, an overall state in the network. This is not a scarce situation because to include network state in a simulator has a very high complexity.

Taking into account this situtation we have decided to take advantage of the technical effort done in two popular simulation tools (Proxycizer and NS) and try to combine them to produce a new simulation tool which allows to define a large scale proxy-cache and replication simulation environment and also offers more information about network details happened in the simulation.

As we described above we have created a new tool based on the combination of two simulation tools, they are Network Simulator (NS)[NS] and Proxycizer [Gadde98], a suite of C++ classes and applications that can be used in simulating and/or driving web proxies. This combination has been posible by the development of an application that we call Proxycizer2ns.

It is important to say that there is a related work to simulate proxy-caches using NS[Yu99], however that work is oriented to Web cache consistency analysis in a cache hierarchy and it does not take into account things related to different control cache protocols and global cache architectures. We considered very difficult to adapt it to different scalable caching environments, beside to add the facility to combine caching and replication that it does not consider and it is an important thing for us. Make all of these adaptations in the library Proxycizer is not a big problem. Our initial intention is to take advantage of the technical effort that we have until now, trying to take the best of the two tools NS and Proxycizer with a minor effort. However we are working in improving the Proxycizer2ns to use all facilities that [Yu99] is offering.

In section 2 the Proxycizer will be described briefly, the adaptations that were necessary to make to consider the combination of replication and caching, and also will be explained the Simulador-Proxycizer application that is a tool based on the Proxycizer library, which create a large scale caching and replication simulation infrastructure. In section 3 we will describe NS, in section 4 Proxycizer2ns will be explained, its architecture and functions. In section 5 we will comment an example, and in section 6 we present future work and final comments.

2. Proxycizer

2.1 Description.

Proxycizer is a suite of C++ classes and applications that can be used in simulating and/or driving web proxy-caches. The Proxycizer library will compile with any recent version of g++ and has been tested on platforms running FreeBSD, Digital, Unix, and Solaris. The Proxycizer library has functions which allow read different log formats like: Squid[Wessels98], Crispy Squid [Gadde99], Harvest[Chankhun95], UC Berkeley Home IP [Gribble97], DEC [Kroeger99] and some propietary logs. This suite of classes/appications allow make simulations of large scale hierachy caches (using ICP protocol), or a large scale collective internet caches using different directory structures (replicated directory, partitioned directory, replicated partial directory) with a little effort in implementation.

Proxycizer was developed in the University of Duke by Syam Gade[Gadde98], which has given support us for the accomplishment of this work. Proxycizer has the inconvenience of not offering support for the definition of replication within a structure of distributed caching, which is in our interest analyze, because many people have found that to

combine caching and replication is a good strategy to improving internet traffic bandwith. As a consequence we have done to Proxycizer library some changes to allow this combination.

2.2 Why Proxycizer?

There are different packages which address different aspects of Internet server simulation and benchmarking. For instance, Wisconsin Proxy Benchmark(WPB) [Almeida99] uses syntetich workloads modeled to emulate typical temporal locality patterns to test proxies under load. WebCASE[Zhang99], its objetives are to provide a framework for caching algorithm constructions and caching statistics collection, and to allow the users to observe algorithm behaviors (hit ratios). The Web Polygraph tool[Polygraph] is a web proxy benchmarking tool which can be used to study proxy performance under various stress conditions. Hbench:Web[Manley99] is a tool which preprocess Web server logs and automatically generates and appropiate traffic model for driving a Web server. S-Clients[Banga97] proposes a scalable mechanism for a driving Web servers to overload conditions. Squid Proxy Analysis (SPA)[Duska97] provides trace-driven simulations tailored to emulate the replacement behavior of Squid caches.

However the packages described above do not create trace-driven simulations which allow to analyze the behavior of different architectures of distributed proxy-caches and replication. It is well known the importance of a good infrastructure of distributed caching and replication to decrease internet traffic and latencies, as a consequence in this work we want to develop a simulation environment which allow us to analyze different proposals of distributed caching and replication (hierarchy, mesh, using replicated directory, partitioned directory, etc.) taking into account the impact in the network (bandwidth consumption, latencies, etc.).

Given these circumstances, the library of classes that offered Proxycizer to us was the one that more approached our expectations.

2.3 Proxycizer limitations.

As we have said the Proxycizer library do not have functions or methods to include replication in a possible simulation. Taking into account the well C++ classes structure in Proxycizer and the support of its developer (S. Gadde) it was no very complicated to adapt the Proxycizer library to include replication. However the Proxycizer library like many others simulators do not consider some network aspects (links bandwidth consumption, latencies, etc). These aspects are very complicated to be added in Proxycizer, that is why We have looked for some tool which offers to us a reliable way to incorporate the analysis of these network aspects. An alternative for this was incorporate the results produced by a Simulator developed using the Proxycizer library to a simulator of networks like NS. NS is a simulator widely used for the analysis and study of protocols and computer networks. As a consequence of this idea We have developed Proxycizer2ns. Proxycizer2ns is a tool which allows to transport resulting data from a simulator done using the Proxycizer library to the Network Simulator (NS). Sections 3 and 4 describe thorough more these tools.

2.4 Simulator-Proxycizer

Simulator-Proxycizer is an application that we have developed and that allows us to simulate different large scale caching and replication infrastructures.

This application is based on the library of classes of Proxycizer, thus we can create proxy-cache and replicas hierarchies using the ICP protocol (as the Squid does [Wessels97]), or we can define a global cache (no

hierarchies) using directory structures, as CrispySquid [Gadde98b] does. All of this infrastructures can be done by a combination of proxy-caches and replicas (we call proxy-mirrors).

The classes strcture of the application Simulator-Proxycizer is showed in the figure 1.

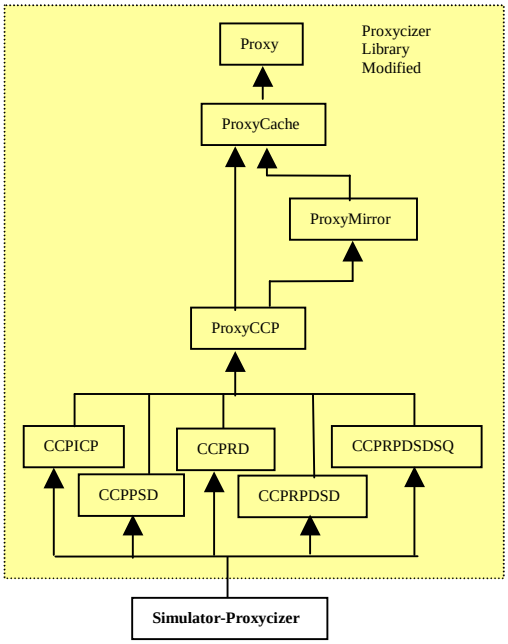

Figure 1. Simulator-Proxycizer Class Hierarchy.

Simulator-Proxycizer receives as input the characteristics of the infrastructure of proxy-cache and replicas which are desired to simulate their behavior through indicating it the wished protocol, that is to say, if we indicated protocol ICP, would assume that a hierarchic infrastructure is desired, if we indicated RD, would construct a caching and replication infrastructure with a replicated directory structure, and so on we can indicate the type of infrastructure that we wished.

The other part important to indicate in the simulation is the network topology in which it is desired to create this proxy-caching and replication infrastructure. This is indicated providing as input a file that contains a graph with the specifications of the wished topology (node links, type of connections, distances).
This file is in format SGB (Stanford Graph Base)[Knuth99], which has been generated through an application program which we have developed and It's based on the Donald E. Knuth [Knuth99] programs library. We have chosen this topology definition format because it is a format widely used in academic and scientific means.

Since Simulador-Proxycizer is based on the Proxycizer library, also it can receive as input a trace file which will drive the simulation. The trace file log format is automatically senses and than Simulator-Proxycizer selects the correct filter for the following log formats: Squid, Crispy Squid, UC Berkeley Home IP, DEC and some Proxycizer logs.

Simulador-Proxycizer provides basically as output 2 files. In one of them appears in textual format the numeric data that is obtained in the simulation in which it talks about number of made requests, rate of hits, number of control messages between caches and replicas for each node (proxy-cache or proxy-mirror) that participates in the simulation. The other file contains the registry of all the events that involve an interchange of packages between nodes, that is to say, query messages, document requests, answers to requests, happened in each node that takes part in the infrastructure of Proxy-caching and replication that we decided to define.

The file that contains the events provoked during the simulation done by Simulador-Proxycizer is generated thanks to an aggregate that we have done to the Proxycizer library. Later the use of this file of events (traces) in Proxycizer2ns will be explained.

3. Network Simulator (NS).

3.1 Description.

NS is a object oriented simulator written in C++, with an Otcl interpreter as a frontend which is used as a command and configuration interface. The simulator has been used widely for the analysis of new protocols of network, routing, queues, traffic in general, multicast, wireless networks, everything at different levels, network, transport, and application. NS has a suite of " agents " (C++ and Otcl class) that allows us to define a network with characteristics similar to almost any network which we found at the present time. NS belongs to project VINT [Vint] in which are participating UC Berkeley, LBL, USC/ISI, and Xerox park researchers.

3.2 Why NS?

At the present time several languages and packages exist to simulate computer networks, we can divide them in 3 types[Law94]: a general-purpose simulation language, a communications-oriented simulation languages, and a communications-oriented simulators. Examples of simulation languages are Arena, BONeS, NetDESIGNER, GPSS/H, MODSIM II, SES/workbench, SIMSCRIPT II.5. As example of the seconds we have OPNET. Some examples of communications-oriented simulators can be: BONeS PlanNet, COMNET III and NETWORK. In general all of these tools help to model network environments to be analyzed, some of them use scripts, others have graphical interfaces, nevertheless they have the disadvantage of being commercial, and their code is not free.

NS is a tool strongly used in the investigation, it offers documentation of its code, and exists availability on the part of its developers to support in projects with NS.

The main reasons to choose NS as our simulator to analyze the happened events in the network during the simulations done with Simulador-Proxycizer is the availability of their code, documentation, and the wide use that it is having within the scientific community which give a touch of confidence, as well as its transparent integration with Nam (tool that next is explained). More details about the NS can be found in the page of the Vint[Vint project].

3.3 Network Animator (Nam)

One more reason to choose NS as our tool to verify the network events produced by the simulations done with Simulador-Proxycizer is in fact to have Nam.

Nam is a Tcl/TK based animation tool for viewing network simulation traces and real world packet traces. It supports topology layout, packet level animation, and various data inspection tools. Nam began at LBL. It has evolved substantially over the past few years. The nam development effort is now an ongoing collaboration with the VINT project, the traces generated by NS simulator are totally compatible with Nam.

With Nam more information about the events that are happening in the simulation is contributed, in such a way that with this one we can obtain more information than it comes from the Simulador-Proxycizer application. We can say that Nam is a tool that also contributes in a didactic sense in the education of networks and simulation. In section 5 we will see how Nam works in the context of this project.

4. Proxycizer2ns

4.1 Description.

Proxycizer2ns is a tool developed in C that perfectly compiles with most recent gnu C. Its

main objective is to transport the results of the simulations done by Simulador-Proxycizer to means that are understood by NS, with the purpose of verifying the effects that these simulations have in the network (bandwidth consumption, latencies, etc.).

As we have said before, a form to obtain results in NS is introducing a program in Otcl that indicates the characteristics of the simulation, topology, and generates the simulation environment using objects understood by NS, so this is what Proxycizer2ns is in charge to include in a Otcl program that constructs in an automatic way.

Proxycizer2ns receives as input three files. First of them it contains the information of the network topology in format SGB, same that was used in the simulations with Simulador-Proxycizer, and it will use to define the network topology that will use NS in a format that this one understands (Otcl objects). The second file contains the traces or the actions that were registered during the simulation by Simulador-Proxycizer, and the last file, it indicates the name to give to the program in Otcl that will be generated automatically by Proxycizer2ns, and that will serve later as input to the NS, and so finally to see the simulation network effects done by Simulador-Proxycizer shown through Nam and some graphs generated by Xgraph.

Figure 2 shows the flow that follows this simulation process.

Once NS begins to run the simulation under the specifications given by the Otcl program generated by Proxycizer2ns, a proxy-cache and proxy-mirror infrastructure is created (nodes in the network), which will have clients representing the elements that send document requests, whose request sequence will be driven by the traces that were originated by Simulador-Proxycizer. Proxycizer2ns generates a file by each proxy-cache and proxy-mirror which will constitute the trace file that will receive during the simulation with NS, following the pattern simulated by Simulador-Proxycizer.

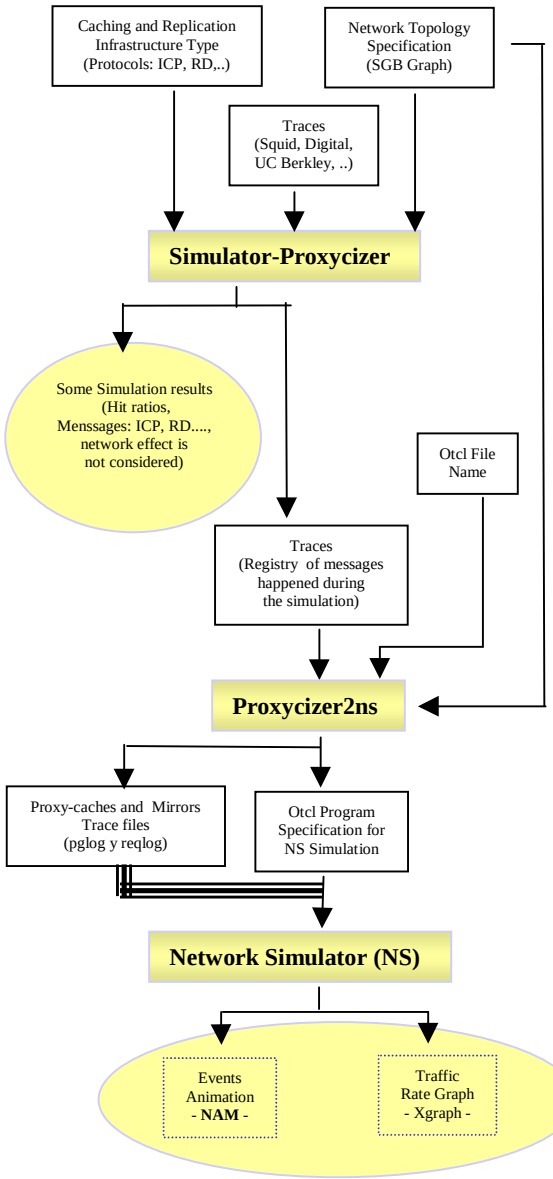

Figura 2. Proceso de Simulación.

It is important to remember that the simulations done by Simulador-Proxycizer also are trace-driven, in this case through the traces that were extracted of a real environment, as for example traces of Squid, Digital, Berkeley, etc.

The traces generated by Proxycizer2ns have a special format that next is explained.

4.2 Traces.

Proxycizer2ns generates traces from the simulations done by Simulator-Proxycizer which will drive the simulation done finally by NS. The format of these traces follows the specification for the PagePool/ProxyTrace class that are used in NS (to see NS manual [NS]).

The PagePool/ProxyTrace class uses real traces to drive simulations. Because there exist many web traces with different formats, in NS has been decided that the PagePool/ProxyTrace class uses an intermediate format so that it can be fed in the simulations done with NS. The PagePool/ProxyTrace type basically consist of 2 files: pglog and reqlog. Each line in pglog has the following format:

<serverID> <URL_ID> <PageSize> <AccessCount>

Each line, except the last line, in reqlog has the following format:

<time><clientID><serverID><URL_ID>

The last line in reqlog records the duration of the entire trace and the total number of unique URLs:

<Duration><Number_of_URL>

Proxycizer2ns creates these files based on what happened in each proxycache and proxy-mirror that took part in the simulation done by Simulador-Proxycizer (information registered in the events file by Simulador-Proxycizer). In this way and using some data structures in Proxycizer2ns we can drive the simulation in NS in such a way that it faithfully reflects the happened things in the simulation done by Simulador-Proxycizer.

5. Example of Use.

The best form than we must to show the operation of this simulation environment using Proxycizer2ns is writing up an example.

We have defined a very simple topology represented by a proxy-cache and proxy-mirror hierarchy which we can see figure 3.

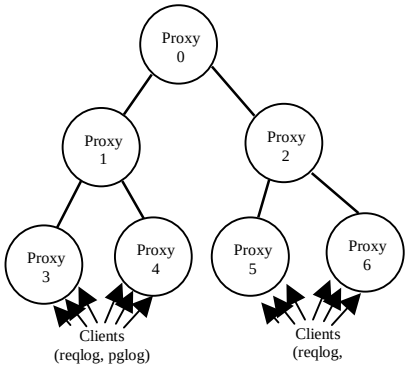

Figura 3. Una topología jerárquica simple.

The file containing the graph with the topology we called it Graph2p2ch.gb. In order to execute Simulador-Proxycizer we would make the following:

Simulator-Proxycizer -prot icp -top Graph2p2ch.gb -trace Squidtracefile

In this way we are indicating that the proxy-caches and replication infrastructure is given by the graph that contains Graph2p2ch.gb. The SGB graphs structure contains attributes that are multipurpose in which we can indicate if a vertex is a cache or is a mirror, and in addition the characteristics to its connections (bandwidth and distance between nodes). This graph have been created by an application that we have developed and we name: create_graph.

With the parameter - prot that appears in the command line, also we are indicating to Simulador-Proxycizer that the communication control between the proxies will be using ICP protocol. The last parameter is indicating to Simulador-Proxycizer that uses a trace file called Squidtracefile, which contains Squid-cache traces which we have taken from NLANR.

When finalizing the Simulador-Proxycizer execution generates 2 output files, one which show in a textual way some data that happened in the simulation:

- Hit and Miss ratios by each Proxy.
- Number of operations that had to be done by the communication control protocol between proxies
- Percentage of bytes transferred and received by each Proxy.

These data are easily obtained thanks to the Proxycizer library has a ProxyStats class which takes accountants which obtain this type of information.

Nevertheless the interesting thing of this work is to observe the impact that these operations simulated by Simulador-Proxycizer has in the network. For it we need to execute Proxycizer2ns in the following way:

Proxycizer2ns Tracefile Graph2p2ch.gb nsfile.tcl

In this command we are indicating to Proxycizer2ns to take the *Graph2p2ch.gb* file to generate the Otcl code necessary to create a network topology with the characteristics included in *Graph2p2ch.gb* using objects which can be understood by the NS.

Proxycizer2ns will generate the code

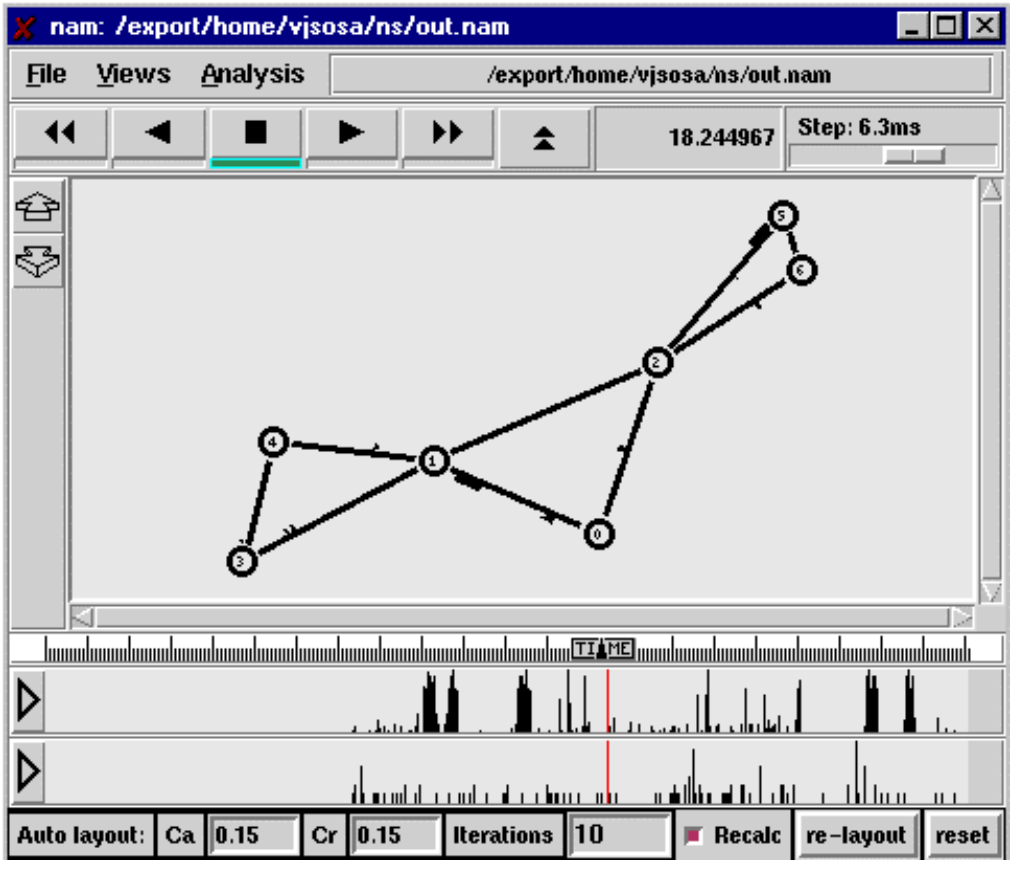

Figure 3. Events Animation in NAM

necessary to create Proxycache and Proxymirror agents that use as generating source the Pagepool traces that will be created from the obtained information of the *Tracefile* file, which was generated by Simulator-Proxycizer.

At the end Proxycizer2ns records the Otcl code which has generated in the *nsfile.tcl* file. In this point we are ready to observe the events that happened in the simulation done by the combination Simulator-Proxycizer and NS. These events will be shown in a graphical way through Nam and using Xgraph.

The last step in the execution of this simulation environment is the following:

ns nsfile.tcl

With this we indicated to NS to run a simulation using the specifications recorded in the *nsfile.tcl* file. Within the code generated in the *nsfile.tcl* file also there are commands which will execute the NAM and the Xgraph.

NAM will be in charge to show the events animation, as well as to provide information of what was happening in the links, and Xgraph will draw the traffic rates in each node which have been detected by NS.

Figure 4 shows a stage of the animation offered by NAM, can be seen the packages interchange between nodes in the hierarchy, with the possibility of extracting more information of this animation with only click the mouse button over the link. In the inferior part are some traffic graphs between links. In general are some data that reflect what is happening in the links in a period of time.

Figure 5 shows the Xgraph graph showing the total of bytes sent/recieved in a period of time determined for each involved Proxy. The observation periods of times can be a parameter in Proxycizer2ns.

It is important to mention that the type of graph used in Xgraph is a parameter predetermined in Proxycizer2ns, in such a way that we are testing several graph types whose

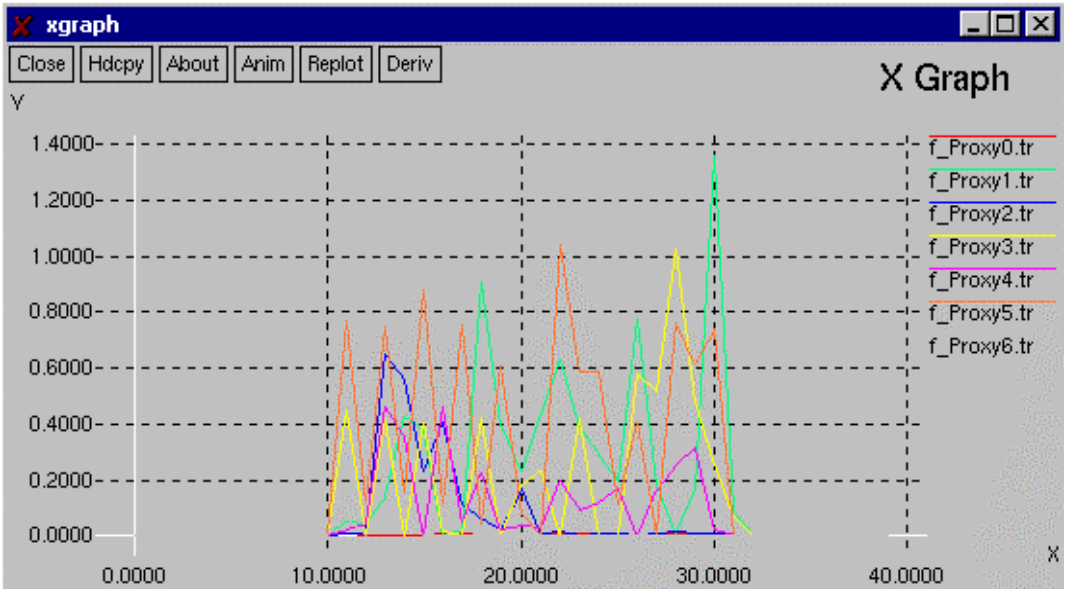

Figure 5. Traffic in Megabytes sent/received by each Proxy in a period of time.

representation is most advisable.

6. Final Comments and Future Work

In this article we have presented a simulation environment which is based on the combination of two powerful simulation tools. They are Network Simulator (NS) and the Proxycizer library. Although it is truth that generating new classes in the NS we can obtain similar results without the necessity to use the Proxycizer library, also is certain that the level of complexity to obtain this is higher. Because the Proxycizer library was created with the purpose of simulating diverse architectures of Web caching, it was much more easy to adapt the Proxycizer library to the use of replication, and to see the effects of the simulations in the network by means of transporting its results in a simple way to the NS.

At the present time several developments exist that try to obtain the same aim that we persecuted with our simulation environment, nevertheless we considered that take like a start point to use two powerful tools in continuos developing is a very acceptable option.

At the moment we are in the process of improvements of Proxycizer2ns in order that it generates Otcl code that uses more functionalities than NS can offer to us. At the same, we are evaluating the cost/benefits that implies to continue doing improvements to the Proxycizer library or definitively to approach the complexity to generate new classes that operate in a direct way with the NS.

References.

[Almeida99] http//:www.cs.wisc.edu/~cao/wpb1.0.html.

[Banga97] G. Banga and P. Druschel. "Measuring the capacity of a Web server". In Proceedings of the USENIX Symposium on InternetTechnologies and Systems (USITS). Department of CS, Rice University, Dec. 1997.

[Chankhun95] Anawat Chankhunthod, at al. " A Hierarchical Internet Object Cache", Technical Report 95-611, Computer Science Department, University of Southern California, Los Angeles March 1995. http://www.usc.edu/dept/cs/tech.html

[Duska97] B. Duska, D. Marwood, and M. J. Feeley, "The measured access characteristics of World-Wide Webclient proxy caches. In Proceedings of the USENIX Symposium on InternetTechnologies and Systems (USITS). Dec. 1997.

[Gadde98] http://www.cs.duke.edu/ari/cisi/Proxycizer/

[Gadde98b] S. Gadde, J. Chase, and M. Rabinovich. "A Taste of Crispy Squid". Workshop on Internet Server Performance. June 1998.
http://www.cs.duke.edu/ari/cisi/crisp/crisp-wisp/crisp-wisp.html

[Gadde99] S. Gadde, J. Chase, and M. Rabinovich. Crispy Squid. Available at http://www.cs.duke.edu/ari/crisp

[Gribble97] Steven .D.Gribble. UC Berkeley Home IP HTTP traces, July 1997. Available at: http//www.acm.org/sigcomm/ITA/

[Knuth99] D. Knuth, "The Stanford Graph Base"http://www-cs-faculty.stanford.edu/~knuth/sgb.html

[Kroeger99] ftp://ftp.digital.com/pub/DEC/ traces/proxy/webtraces.html

[Law94] A. M. Law and M. G. McComas, "Simulation software for communication networks: the state of the art". IEEE Communication Magazine 1994 32:44-50.

[Manley99]http//:www.eecs.harvard.edu/~vino/web/hbench-web/.

[NS] http://www-mash.CS.Berkeley.EDU/ns/
[Polygraph] http://polygraph.ircache.net/

[Wessels97] D. Wessels and K. Claffy. "Application of Internet Cache Protocol (ICP), version 2". Request for Comments RFC-2186.
[Vint] Virtual InterNetwork Testbed Project http://netweb.usc.edu/vint/

[Yu99]http://www.acm.org/sigs/sigcomm/sigcomm99/papers/session5-1.html

[Zhang99]http://www.ircache.net/Cache/Workshop99/Papers/zhang-final.ps.gz.