# Techniques for Speeding Web Access in the Developing World

December 6, 2001

## Abstract

This paper explores techniques for building special purpose proxies to speed web access to the developing world. In particular, we present a design for Tandem Proxies (TP), a system of two proxies that utilize a combination of caching, diffs, and compression to minimize data sent across low-bandwidth, low-quality, high-latency links. Using an experimental testbed simulating the network topology of the developing world, we show that TPs can reduce traffic across poor links by as much as 50% over traditional caching.

## 1 Introduction

### 1.1 Project Goals

Not all the world is blessed with fast connections to the internet. In much of the developing world, for example, users must cope with low-quality, low-bandwidth, high-latency links to the web. This paper explores a variety of caching techniques that could potentially improve delivery speeds of web pages without sacrificing data freshness.

Given the wealth of existing methods researchers have previously devised to address speeding content delivery [6], our project's major goal is to analyze which techniques are the most valuable in the context of the developing world. Hence our project breaks down into several phases:

- Research existing and new methods of accelerating content delivery.
- Incorporate the best methods into a deliverable software package.
- Analyze the relative merits of each technique via laboratory simulation.

### 1.2 Design Criteria

While dozens of known techniques could be applied to this problem, the solutions we explore aim to satisfy the following design constraints:

- No modification of existing web browsers or web servers.
- Compliance with HTTP 1.0 and HTTP 1.1 protocols.
- Downloaded content must not be stale, unless allowed by HTTP RFCs.

We wanted to devise a solution that did not only function on custom-configured systems in the laboratory, but one that would also be easily deployable in practice out in the field. Hence, we excluded from consideration designs that would require special purpose modification of existing web browsers or web servers. Similarly, few people would adopt a solution that was non-compliant with the existing HTTP standards [1] due to a fear of introducing unforeseen incompatibilities into their browsing experiences. Finally, we excluded solutions that forced users to accept stale content at the price of improved delivery speed. Such solutions hint of geographic discrimination and further reinforce the digital divide.

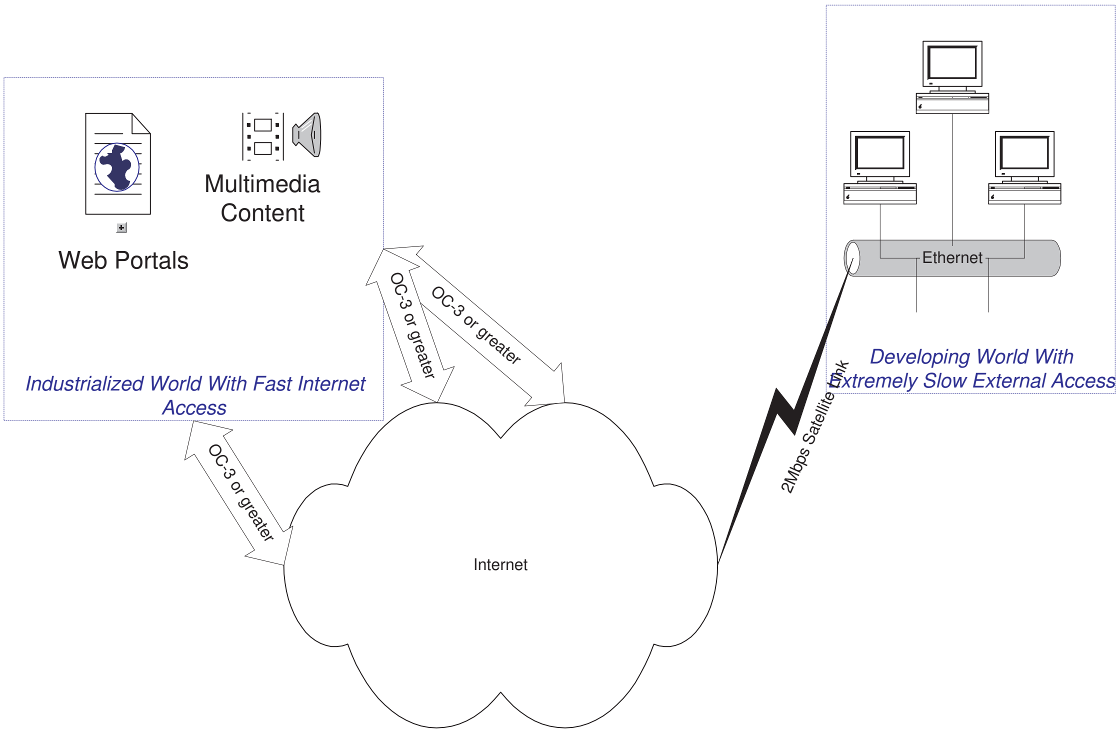


Figure 1: Strathmore College Network Topology

### 1.3 Network Topology of the Developing World

Strathmore College in Nairobi, Kenya has a network topology typical of many institutions in the developing world. The college has approximately two hundred Pentium class PCs linked together on a 10-BASE-T ethernet LAN. Due to the prohibitive cost of high-bandwidth connections, the entire school shares a single 128kbps link to Africa Online, a local ISP. Africa Online, in turn, is forced for political reasons to send all of their data out to the internet through a single 2Mbps satellite link managed by the Kenyan telecom monopoly.

Most of the websites commonly visited by students at Strathmore are located in the United States and Europe. In particular, portal sites such as Yahoo! and web-based email sites such as Hotmail are exceedingly popular. As a consequence, most of Strathmore's HTTP requests require transit through Kenyan Telecom's 2Mbps satellite link, where their bits march side by side along with all of the data from the rest of the nation. This excessive multiplexing produces a bottleneck that nearly paralyzes Strathmore's ability to access popular websites.

Our on-site experiences at Strathmore revealed that the quality of the link was poor not only in terms of bandwidth, but also reliability and latency. Web transfer speeds from portal sites based in the United States averaged 300 bytes per second, when not stalled. Typically, nearly sixty percent of these packets were dropped completely. Those that were not dropped required on the order of 300ms to make the journey.

Presuming that Strathmore is representative of many entities in the developing world, we extract the following key observations about the developing world's network topologies and surfing habits:

- Each institution houses a fast LAN such as 10B-T ethernet.
- Users popularly access portal websites in the USA or Europe.
- Data exchange with the USA or Europe is high-latency, low-bandwidth, and unreliable.

## 1.4 Existing Techniques

The most effective way to speed content delivery is to reduce the amount of data transmitted across bottleneck links. All of the techniques described below attempt to curtail such traffic.

### 1.4.1 Compression

Many existing systems utilize compression to curb network traffic. The originating web server compresses the content and sends it to the client, who then decompresses the data to recover the original content. The best known text compression algorithms have a 2:1 savings. Images, such as JPEGs, are hardly compressible at all because the image format is already encoded in a fashion to minimize file size. Since most web content is an amalgam of text and images, the bandwidth savings from compression is significant, though not spectacular.

### 1.4.2 Caching

The most widely adopted technique used to relieve network congestion is caching. When a file is originally downloaded from a server, the client stores the file locally in memory. In the future if the file is requested again, the client fetches the file from memory rather than re-requesting it from the original server. Caches are frequently incorporated into HTTP proxies so that multiple clients can share the same pool of pages [5].

Traditional caches and proxies can only achieve this savings if the cached file and the remote file are exactly the same. As the web becomes populated with increasingly dynamic content, the likelihood of a page being cachable is plummeting, particularly with portal sites. When loading the front page of CNN twice in a row, for example, the text is not cachable because CNN's webserver dynamically stamps the page with the current time. Hence, "all-or-nothing" caches are becoming ineffective.

### 1.4.3 Dynamic Tags

Website designers are gradually becoming aware of the non-cachable nature of dynamic pages. Some have proposed extensions to HTML that would allow a content creator to tag certain regions of a webpage as having dynamic content [7]. To return to the CNN example, the area of the page with the time would be encased in special tags denoting dynamic content. A smart browser could then recognize the dynamic tags and use them to improve the efficiency of its cache. For the foreseeable future, however, dynamic tags are not a realistic method of reducing network traffic because they would require massive adoption by website designers.

### 1.4.4 Diff

One of the most promising methods of exploiting the redundancy of content within portal websites is to use diffs [2]. Under this scheme, if the client has a page already cached, it could simply query the web server for a description of the changes that have since been made to the webpage. Since only the differences are transmitted over the network, the traffic is greatly reduced for dynamic pages that retain common elements. This technique, however, places a burden upon web servers by requiring them to retain a history of versions for each webpage and to compute diffs between pairs of versions.

# 2 Design

## 2.1 Design Overview

The solution we propose utilizes a pair of proxies that straddle the low-bandwidth low-reliability high-latency link. By using a combination of caching, diffs, and compression, the tandem proxies are able to reduce the traffic across the slow link. Adopting terminology introduced by the Rproxy [6] project, the proxy on the LAN in the developing country is referred to as the decoding proxy (DP). The proxy on a LAN in the United States or Europe is called the encoding proxy (EP).

## 2.2 Tandem Operations

To better understand the mechanics of the EP/DP tandem operations, it is necessary to examine what messages the proxies exchange with each other in the context of cached pages, non-cached pages, and "nearly-cached" pages.

### 2.2.1 Cached Pages

When a client requests a webpage from the DP, the DP first checks to see if the content is cached. If the file is cached and not expired, then the DP returns the page to the client as a normal proxy would do. Since the DP is on the client's LAN, this connection should be fast and will not require any traffic across the poor quality link.

### 2.2.2 Non-Cached Pages

If the page is not cached at all, then the DP queries the EP for the webpage. The EP then fetches the page from the internet if the EP does not already have it cached. Before sending the file to the DP in a compressed format, the EP caches the file itself. Once the DP receives the compressed content, it decompresses the file. The DP then caches it, even if it has a "Pragma-nocache" header, and finally sends the content to the original client. In this fashion, the DP caches every precious bit of data sent across the low bandwidth link. Though it may seem a violation of the HTTP specification, it is not because the DP will never send it to the client unless it is guaranteed to be fresh (see 2.2.3).

When compared to a standard single proxy scheme where data travels directly from the server to the proxy, this method has the potential of introducing more latency because the data must travel a longer route first from the server to the EP and then from the EP to the DP. However, given that the EP has a much faster connection to the original web server than it has to the DP, this extra latency is not significant; the poor quality link between the EP and DP is still the bottleneck. Furthermore, since the content across the poor link is compressed, it reduces the traffic and potentially increases delivery speed.

### 2.2.3 Nearly-Cached Pages

The most interesting case is when a file is "nearly cached" on the DP. By nearly cached, we mean that the file is currently cached but for some reason the DP must verify the content of the file. This can happen, for example, if either the client explicitly reloads a page or if the cached file was not allowed to be cached.

Now the task at hand for the DP is to verify that the cached file is fresh. It sends a message to the EP asking for updates to the file. The DP identifies to the EP the version of the file it has by computing a checksum. The EP fetches the file from the webserver and analyzes differences between the fresh file and the version that the DP is known to have. Once the differences are calculated, which can be done on the fly even for streaming content, the EP compresses the description of the differences and sends it to the DP. When the DP receives the response, it decompresses the message and brings its version of the file up to date if necessary. Finally, it passes the data to the client.

If it turns out that the nearly-cached file is equivalent to the fresh file, then only a few packets of data are transferred across the poor link. The request essentially becomes a variation on the HTTP 1.1 "Conditional GET" request. Its performance can be no worse than the conditional GET.

If the nearly-cached file has changed slightly, as we've previously discussed pages on portal sites often do, then this scheme offers a dramatic advantage over any other technique. Since only the differences are transmitted across the poor link, the overall bandwidth savings could be a significant fraction. Most dynamic content, which under any other scheme would be uncachable without the sacrifice of either

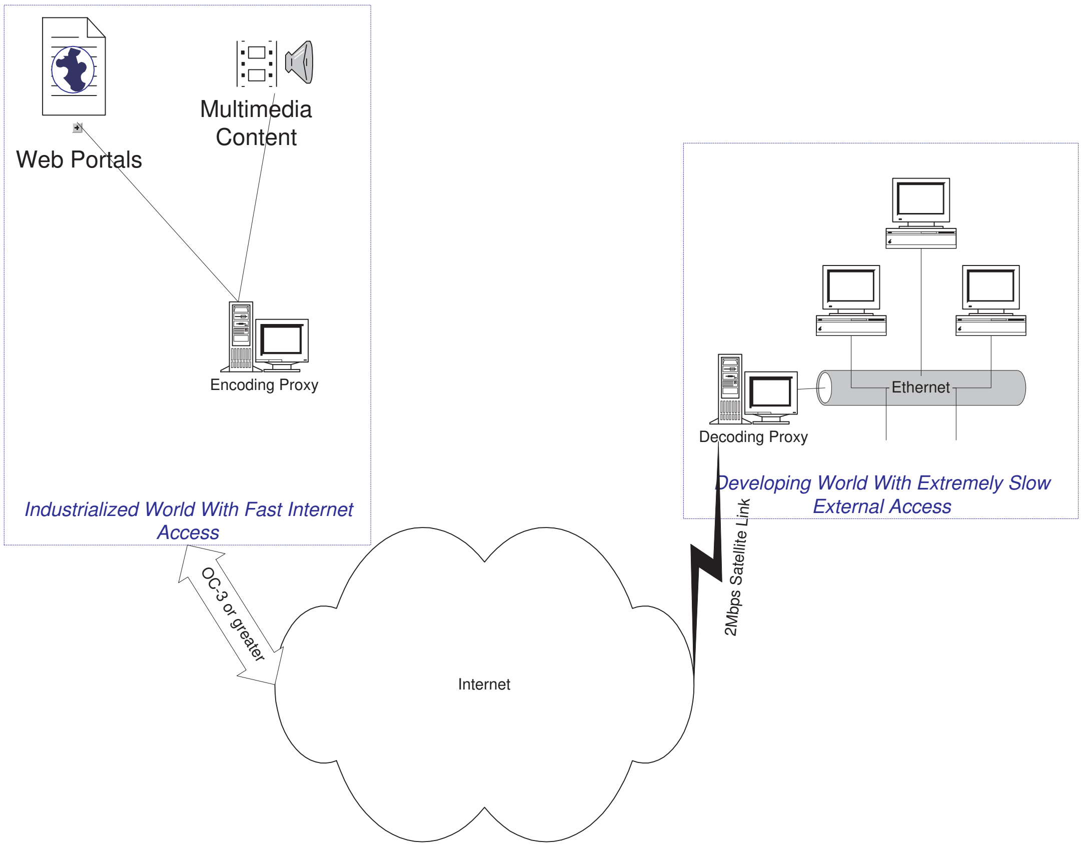


Figure 2: Tandem Proxy Architecture Overview

transparency or consistency, will benefit from these savings.

Though rare, it is possible that the file has changed entirely. In terms of the amount of data transferred over the poor link, it will be as if the file had never been cached on the DP in the first place, so Section 2.2.2 applies.

No matter if the file has changed or not, the DP is now guaranteed to have a fresh version of the file, so the HTTP specification is not violated even if the DP had originally cached the file despite seeing a "Pragma-nocache" header.

## 2.3 Summary of Features

In terms of theoretical performance, the tandem proxies (TPs) either meet or far exceed the content delivery speeds of traditional proxies:

- For cached pages, the TPs behave like traditional proxies.
- For non-cached pages, the TPs utilize compression for performance no worse than traditional proxies.
- For nearly-cached pages, the TPs utilize diffs and compression for dramatically superior performance.

The tandem proxies also implement a fully transparent solution:

- Clients see DP as a regular proxy. No special configuration required.
- Servers see EP as a regular proxy. No special configuration required.
- Rsync cache verification ensures HTTP RFC compliance.

### 2.4 Summary of Weaknesses

The chief stumbling block is this proposal is that it requires someone who wants to setup a DP in the developing world to find a partner in the USA or Europe willing to setup a EP to complete the pair. While the EP would not consume much bandwidth for the host in the industrialized world, it would still require a hardware investment and occasional administration. Forging these partnerships may be a challenge and may limit scalability of the system.

From an operational standpoint, having a system so geographically separated could pose recovery problems as well. If the EP were to go offline, then the DP would be left incapacitated. Since the two proxies are likely to belong to different organizations, getting an offline proxy back online may not be a rapid process. Of course, the DP could simply revert to becoming a a traditional proxy if it cannot find its partner EP, but then the system would lose any performance gains that the tandem proxies provide.

The last significant weakness of this design is that it is not sufficiently generic to serve as a solution to improving content delivery speeds to all low bandwidth clients. Only network topologies that have a single well-defined bottleneck can benefit. This solution does not apply, for example, to speeding content delivery to clients on congested networks.

## 3 Implementation

To evaluate this design, we had to implement a prototype EP and DP.

### 3.1 WBI

To accelerate development, we utilized IBM's WBI architecture [3]. The WBI development kit allows programmers to create customized web intermediaries using Java APIs. An intermediary is an entity that is capable of processing information that flows through it. A caching proxy, therefore, is a kind of intermediary. WBI's libraries provide convenient objects for passing data between machines using HTTP requests and responses. All the developer is required to do is to customize a WBI by writing Java plugins that perform the desired information processing task.

### 3.2 Decoding Proxy

Tailoring WBI into the Decoding Proxy required the creation of the following plugins:

- Gzip decompression plugin to process data from EP
- Diff and Patch plugin to update cached files

Caching functions had to also be independently implemented to complete the DP. All of this code, including the plugins, were written in a manner consistent with the above Design section.

### 3.3 Encoding Proxy

Similarly, writing the Encoding Proxy required the creation of the following WBI plugins:

- Gzip compression plugin to send data to DP
- Diff analysis plugin to compute changes from cached files

The EP used the same caching functions as the DP. The code closely followed the specifications of the Design section.

## 4 Testing

Analyzing the performance of the TP architecture required great care. Our analysis strategy involved multiple phases. First, we determined our testing goals and devised a series of experiments to meet those goals. Then, we implemented a collection of tools that enabled us to carry out the tests in an environment that simulated the developing world's network infrastructure. Finally, we performed the tests.

## 4.1 Goals

We designed our tests to gain insight into three major questions. How well do the Tandem Proxies perform? How much does performance vary depending on the type of page being browsed? And to what degree does each technique utilized by the Tandem Proxies contribute to the system's performance?

### 4.1.1 Performance

Given the specific problem domain of this project, we focused on evaluating the following two major facets of performance:

- Minimizing the time required to load a web page (latency).
- Minimizing the amount of data transferred across the poor quality link.

These two criteria are intimately linked, but we list them separately to provide a framework for terminology.

### 4.1.2 Contexts

In particular, we designed our experiments to test performance in the context of both web pages with static content as well as those with dynamic content. In order to gauge the system's general overhead, we were also careful to measure the Tandem Proxies' performance when loading pages initially, and not just upon reload.

### 4.1.3 Techniques

Perhaps the most important piece of information we intended to extract from our tests was the relative contributions of caching, diffs, and compression to the system's overall performance. WBI's plugin architecture facilitated this performance decomposition by allowing us to selectively enable or disable different features of the TPs.

## 4.2 Environment

Of paramount importance to our evaluation effort was recreating the network topology of the developing world (Kenya, specifically). By modeling such a topology in the laboratory, we could use it as a testbed for assessing the performance of the Tandem Proxies.

### 4.2.1 DummyNet

Recall that the key feature of Kenya's network toplogy is the presence of a poor quality bottleneck link separating a high quality LAN on the Kenyan side from the high speed internet on the American or European side. To model this poor quality link, we used a FreeBSD workstation running DummyNet [4]. DummyNet is a network simulation tool that simulates packet loss, delays, and bandwidth restriction among other features. Since DummyNet intercepts packets between layers in the networking stack, it can be used as a traffic shaping tool. We used DummyNet to shape network traffic in and out of our workstation in a manner that emulates the bottleneck link. More specifically, the workstation became a network link with 300ms one-way delay, 60 percent packet loss, and 300byte/s bandwidth.

### 4.2.2 Global Toplogy

To complete the infrastructure, we added two more machines to our testbed. One machine simulated the high-speed LAN at Strathmore and the other machine simulated the high-speed LAN at MIT. The two machines (simulated LANs) could only communicate with each other through the DummyNet-enabled machine.

This testbed allowed us to setup clients and our decoding proxy on the simulated Kenyan side of the global network and to setup servers and our encoding proxy on the simulated American side.

## 4.3 Experiments

We performed a total of four experiments and for each experiment we recorded latency and byte transfer statistics. Two of the experiments, C1 and C2, were controls, and two experiments, V1 and V2, were variable experiments. Type 1 experiments load webpages initially and Type 2 experiments reload webpages.

### 4.3.1 Controls

Any statistics assessing the performance of the Tandem Proxies would be virtually meaningless without an experimental control for reference. We chose to use Squid [5], a popular caching web proxy, as the control. Squid is a good choice not only because it is popular, but also because it provides a benchmark for standalone proxy architectures.

Therefore, in our first experiment (C1), we deploy a Squid proxy on the Kenyan end of the link. Using a simulated client (a perl script) deployed on the Kenyan side, we query Squid for a series of websites originating on the American side. Since the Squid cache is initially empty, Squid must query the web servers for content across the poor link. The statistics from this operation provide a reference figure for how much uncompressed data must be sent across the poor quality link.

In our second control experiment (C2), we reload each website once from our client. Squid returns cached content back to the client and queries the original servers to retrieve uncachable content. Hence the data transferred across the poor quality link in this experiment represents the volume of content uncachable under HTTP RFCs.

### 4.3.2 Variables

Once the control experiments have been performed, we essentially perform them again except this time we deploy our Tandem Proxies instead of Squid; the EP is on the American side and the DP is on the Kenyan side. Experiment V1 is identical to C1, except this time when the web pages are initially loaded, the data travels across the slow link in compressed form. By comparing statistics from C1 and V1, we can judge how much savings the tandem proxies achieve due to compression.

When the reload experiment (V2) is performed on the tandem proxies, the data sent across the poor link now has a special significance. It represents diffs between content that the DP had cached but was unable to return to the client due to HTTP RFCs and fresh content that the EP just fetched directly from the originating server.

## 4.4 Results

### 4.4.1 Qualitative

In the control experiments using Squid, content servers would often time out due to the ultra-slow transfer speeds. In fact, no page accessed during the controls ever loaded fully within twenty minutes (at which time we chose to terminate the test). This is, however, consistent with our experiences in Kenya, where pages only rarely loaded to completion.

In contrast, when we employed the TPs in the variable experiments, every page loaded to completion within 8 minutes. Although this is still slow by industrialized-world standards, it is a stunning and encouraging qualitative result.

### 4.4.2 Quantitative

Because of the snail's pace of data exchange, collecting statistics was especially challenging. We were only able to gather data on three websites, www.hotmail.com, www.cnn.com, and www.google.com. Table 1 shows the total aggregate number of bytes transferred across the poor link for each experiment. Latency data is not shown because all of the type C experiments failed to complete within 20 minutes.

| | |
|---|---|
| V1<br>173.0 KBytes | C1<br>332.4 KBytes |
| V2<br>89.6 KBytes | C2<br>171.1 KBytes |

Table 1: Total Bytes Transferred in Experiments

We can derive key statistics from this table. Since C1 did not complete loading, a conservative estimate of the total amount of data contained on the webpages is 332.4KBytes.

Experiment V1, which did complete loading, only transferred 173.0KBytes. Compression, therefore, seems to save at least 159.4KBytes (or 48%).

Experiment C2, which also failed to load entirely, transferred 171.1KBytes. Comparing this to C1, we see that caching seems to save on the order of 161.3KBytes (48%).

Finally, experiment V2, the other successful loader, transferred 89.6KBytes of data, which in this case were compressed diffs. We expect 52% of the original compressed data to be re-requested across the link, which would be 89.9KBytes. Diffs, therefore, seem to account for only 0.3KBytes (0.3%), which is negligible.

# 5 Discussion

## 5.1 Goal Evaluation

Given that Squid was unable to load a page successfully within twenty minutes and the Tandem Proxy was able to, the TP definitely meets our first performance goal of improved latency. What drives this decrease in latency is most likely the reduced number of bits forced to make the perilous journey across the poor intercontinental link. TP, therefore, has compression to thank for its ability to meet its second performance goal of reduced bottleneck traffic.

One of the other analysis goals was to characterize the performance of the TP depending upon the type of page visited (static vs. dynamic). Unfortunately, the slow transfer speeds prohibited enough data from being collected to settle this issue.

Finally, our experiments did help us decompose which techniques seem to be the most useful in speeding web access to the developing world. Caching, the tried and true scheme, helped curtail traffic by fifty percent. For pages that must be transferred, compression can save an additional fifty percent. Diffs, which we projected would help curtail traffic significantly, had almost no effect at all. Non-cachable content such as image advertisements with unique URLs occurred so frequently relative to diff-able pages that the diff technique provides almost no performance enhancements.

## 5.2 Significance

Providing those in the developing world with more solid information infrastructures is a critical goal. Though the Tandem Proxies do not provide any Earth-shattering advances in web transfer speeds, our tests and simulations show that the TPs can open access to a class of clients that previously were digitally disenfranchised. In the next decade, other technological advances such as those in satellite availability and terrestrial fiber optics will dwarf the performance gains of the Tandem Proxies. But in the interim, a generation of students is being swallowed by the digital divide. Accordingly, in the next few months we plan on polishing our distribution and making it available to schools in the developing world.

## 5.3 Future Work

Further research may be performed in the areas of binary diffing and compression as well as in image transformation (such as reducing number of colors in images to save space) to minimize necessary bandwidth consumption. Parallelizing testing strategies should be explored in order to better characterize the proxy's performance on workloads with ultra high latency.

# 6 Conclusion

This paper explored techniques for speeding web access to the developing world. We introduced a design called Tandem Proxies (TP) in which two proxies work in concert by using a combination of compression, diffs, and caching in order to reduce traffic across bottleneck links. We showed through a laboratory simulation of the networking toplogy of the developing world that our implementation of TP can curtail bottleneck traffic by 50%. This produces significant performance gains that enable clients in the developing world to access websites that were previously beyond reach.

# 7 References

## References


[1] R. Fielding, J.Gettys, J. Mogul, H.Frystyk, T. Berners-Lee. RFC2068: Hypertext Transfer Protocol – HTTP/1.1, January 1997. http://www.w3.org/Protocols/rfc2068/rfc2068.

[2] Peter Fenwick. Differential Ziv-Lempel Text Compression, August 1995. http://www.jucs.org/jucs_1_8/differential_ziv_lempel_text/paper.pdf.

[3] R. Barrett & P.P. Maglio. Intermediaries: An approach to manipulating information streams, April 1999. In *IBM Systems Journal, 38*, 629-641.

[4] Luigi Rizzo. Dummynet: a simple approach to the evaluation of network protocols, January 1997. http://info.iet.unipi.it/ luigi/dummynet.ps.gz.

[5] Squid Web Proxy and Cache. http://www.squid-cache.org.

[6] A. Tridgell, P. Barker. rsync in http, 1999. http://rproxy.sourceforge.net/doc/calu_paper/calu_paper.ps.

[7] P. Bethke. Corral Methodology for Publishing Dynamic Web Sites, March 2001. http://www.corralmethod.org/pdf_docs/corral_v1.pdf.