Technical Report

Department of Computer Science
and Engineering
University of Minnesota
4-192 EECS Building
200 Union Street SE
Minneapolis, MN 55455-0159 USA

TR 10-012

Standing on the Shoulders of Others: Using Proxies to Opportunistically Boost Distributed Applications

Siddharth Ramakrishnan, Robert Reutiman, Abhishek Chandra, and Jon Weissman

May 25, 2010

Standing on the Shoulders of Others: Using Proxies to Opportunistically Boost Distributed Applications*

Siddharth Ramakrishnan, Robert Reutiman, Abhishek Chandra, and Jon Weissman

Computer Science and Engineering, University of Minnesota,
200 Union Street SE, Minneapolis, MN 55455
{ramak,reutiman,chandra,jon}@cs.umn.edu.com

Abstract. Distributed applications are increasingly relying on and integrating remote resources including community data sources, services, and computational platforms. The coupling of distributed resources for data-intensive applications can expose bottlenecks across highly shared pathways, e.g. the network. In this paper, we consider applications that access remote clouds for data, computation, or services, and focus on alleviating the bottlenecks that they may encounter by using middleware deployed on a proxy network. We propose a proxy network that sits between the cloud and the end-user application offering resources to mitigate bottlenecks. In particular, we show how proxies can eliminate network bottlenecks by smart routing and perform in-network computations to boost application performance. We build on existing literature to create a unique synthesis of ideas that is the proxy network.

Our results obtained through experiments on PlanetLab are promising, showing substantial improvement in latency, bandwidth, and even jitter, which can benefit a wide class of data-intensive applications. Our microbenchmarks show that routing data through select proxies can accelerate network transfer by at least 25% in almost half the cases considered. Experiments using a distributed Montage[1] workflow mapped on a WAN showed 13% end-to-end performance improvement when data was routed through a proxy network. In addition, proxies also reduced the montage output data delivery time to users on mobile devices by 65-80% through compression and data transformation.

* The authors would like to acknowledge grant NSF/IIS-0916425 that supported this research

[1] This research made use of Montage, funded by the National Aeronautics and Space Administration's Earth Science Technology Office, Computation Technologies Project, under Cooperative Agreement Number NCC5-626 between NASA and the California Institute of Technology. Montage is maintained by the NASA/IPAC Infrared Science Archive.

1 Introduction

Distributed applications are increasingly relying on external resources for data sources, services, and computations. Platforms supporting this model of computation span peer-to-peer, Grid, and now, Cloud systems. Many distributed high performance computing applications routinely access large remotely-located community datasets, exploit remote services, and outsource computations to a variety of platforms including supercomputers, voluntary P2P systems, Grids, and Clouds. A common thread across high performance distributed applications is the interconnection of components and resources across shared pathways, e.g. the network, third-party services, virtualized servers, and so on. Such shared pathways are inherently capacity-limited and present bottlenecks to HPDC applications that live on the upper-end of data and compute requirements. Even beyond the application execution and computation, the diverse capabilities of end-user hosts, ranging from desktop machines to resource- and energy-constrained mobile devices, create bottlenecks for output delivery to users of these applications. Such end-point bottlenecks are particularly critical in applications requiring visualization or real-time human input for execution.

In this paper, we consider applications that access remote clouds for data, computation, or services, and focus on alleviating the bottlenecks that they may encounter. Clouds currently represent a diverse landscape containing resource providers, application hosting platforms, and service providers. In this paper, we use the term cloud broadly to refer to a large-scale out-sourced data, compute, or service resource, made available to end-users and applications. We propose to utilize middleware deployed on a *proxy network* that sits between the cloud and the end-user application offering resources that mitigate the bottlenecks. We have developed a simple tool to identify such bottlenecks, and use the proxy network to route around bottlenecks. This network also boosts application performance by performing in-network computation close to both endpoints. The proxies combine the power of network optimization coupled with in-network computing to benefit end-user machines that are poorly provisioned either in networking or compute resources.

To demonstrate the potential of proxies, we have performed experiments on PlanetLab, exploiting its resource and network diversity. We show that such proxies offer numerous optimizations for network performance as well as data transformation that can benefit data-intensive HPDC applications specifically, and distributed applications in general. We present both microbenchmarks and application results using Montage as an illustration of the potential benefits of our approach. Our microbenchmark results show that our approach has great promise as communication can be accelerated significantly, by as much as 56% for TCP bandwidth. Further investigations show that proxies improve the network performance by at least 25% for half the cases we investigated. Results for Montage show that proxies can help reduce the data download bandwidth from data sources and within the application stages via smart routing. The end-to-end acceleration for the Montage workflow is approximately 13%. Proxies also

reduce the montage output data delivery time to users on mobile devices by 65-80% through compression and data transformation.

2 System Architecture

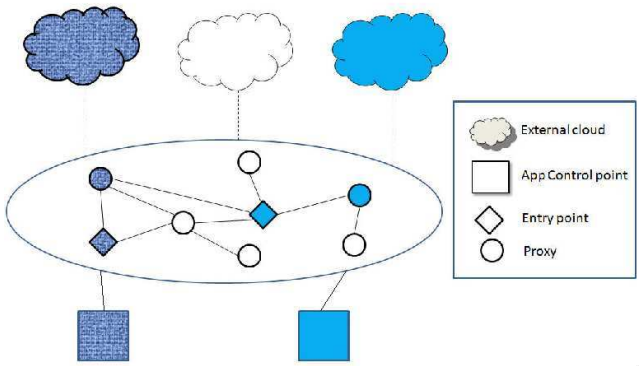


Fig. 1: System Architecture: Components include clouds, proxies, and application nodes

The system architecture is designed to integrate user applications that interact with external clouds. The core of the architecture is a proxy network that contains a configurable collection of nodes offering varying amounts of resources including bandwidth, storage, and computation. One strong appeal of the proxy network is that it offers nodes at many network locations that may provide improved performance with respect to external clouds and/or the end-user application. The proxy network has a number of entry point nodes that can be discovered by applications. We envision that such a network could be realized in a variety of ways (e.g. volunteers, CDNs, and so on), and we do not assume a specific deployment model in this paper. The system architecture contains four primary components (Figure 1):

- entry point
- external clouds
- application control
- proxy node

The *entry points* are a restricted set of "super proxies" known to applications. The entry points have greater resources than a typical proxy and are more likely to be "up". These proxies also store historical resource information about all other proxies and run proxy selection algorithms on behalf of an application.

The *external clouds* represent an external network resource that may provide data, computation, or available services to be integrated within an application.

For example, Amazon EC2 is a computation cloud, SkySurvey is a data cloud, and NCBI Blast is an external service. External clouds provide an interface to enable application integration, e.g. via http or Web services.

The *application control* node taps into the proxy network to exploit its additional resources. It contacts an entry point to locate proxy resources that can accelerate the application with respect to the access and use of external clouds. Once proxy resources are identified, the control node initiates the application, and when finished, terminates the application and releases proxy resources.

The heart of the proxy architecture is the *proxy node*. Proxies provide "bottleneck relief" for applications accessing external clouds. The proxy network consists of a large number of logically connected edge nodes that may assume a rich set of roles to boost the performance of distributed applications, including:
• *Cloud service interaction:* A proxy may act as a client to an external cloud service. This role allows a proxy with better network connectivity to access one or more cloud services. For example, a proxy may have much higher bandwidth to/from a cloud service relative to the end-user.
• *Data Transformation:* A proxy may carry out computations on data via a set of data operators. This role allows a proxy to filter, compress, merge, and mine data.
• *Caching:* A proxy may efficiently store and serve data to other nearby proxies that may consume the data later on. Proxies can also cache intermediate results from a cloud interaction that may be reused again.
• *Routing:* A proxy may route data to another party as part of an application workflow. This role allows a proxy to efficiently send data to another proxy or application component for additional processing, caching, or cloud service interactions. This role is particularly important if the application is interacting with multiple clouds which are all widely distributed, and there may be no single proxy that can efficiently orchestrate all of these interactions.

There is nothing that prevents an application control node from also participating as a proxy as well. Indeed, this might well be a reasonable incentive mechanism to construct a volunteer-based proxy network. To enable proxy selection, proxies run a configurable resource monitoring stack that provides timestamped local and non-local resource availability. The latter includes network probes that provide various metrics including UDP/TCP latency, bandwidth, and jitter, from each proxy to a configurable set of external clouds. This information is periodically sent to the proxy entry points to enable proxy selection.

A distributed application logically consists of some number of each component type: one or more external cloud services, one or more proxies, an entry point, and application control. The application may also contain other components running locally or remotely that do not directly interact with the proxies or clouds. In Figure 1 we depict two applications (shaded and patterned) and the selected components respectively.

Conceptually, the application workflow is the following:

1. Application control: locate and contact an entry point
2. Entry point: select proxies and enable roles

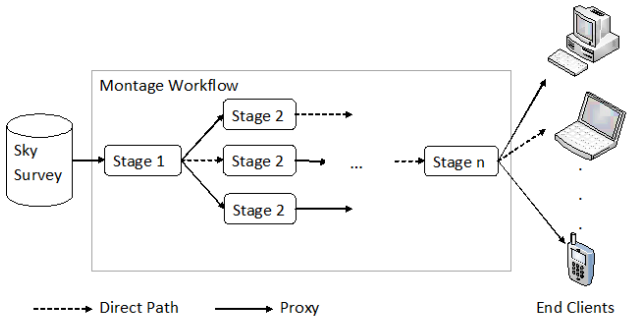


Fig. 2: Application Acceleration - the Montage workflow

3. Application control/Entry point: wire the application components
4. Application control: initiate application
5. Proxy network: accelerate application
6. Application control: terminate application

In Figure 2, we illustrate how an application can benefit from a proxy network. A simple example from image processing could be an application that retrieves images from a data cloud (e.g. SkySurvey), performs in-network processing (e.g. Montage mosaic construction), and then returns the output to the end-user. Proxies could accelerate this application in the following ways: (1) they can provide a better network path to retrieve input data from the cloud to the computation node(s), (2) they can provide a better network path to deliver output data from the computation node(s) to the end-user, (3) they can distill or compress output if the end-user device has reduced display capacity or a poor network path, and (4) they can perform in-network computations when proxy resources offer greater power than at the application control site. Proxies (in 3 and 4) allow customized processing on behalf of the application. We empirically illustrate all of these opportunities with Montage in Section 3.2. Proxies are not a new idea by themselves, but it is the synthesis of the diverse roles of proxies that is novel.

In this paper, we focus on evaluating the potential performance benefits offered by proxies. Programming proxies is outside the scope of this paper. In prior work, we have developed a generic programming framework for enabling proxies to act as a client for any http-based Web services [8]. The computing role can be confined to a predetermined set of trusted operators downloaded by the proxy, similar to the trust expressed by clients in cycle-harvesting systems such as Condor [14] or @Home networks with respect to executing non-local code. Arbitrary execution of non-local code can be enabled by various sandboxing technologies such as virtualization [7], language restriction (e.g. Java), and others (e.g. Google Native Client [53]). We also do not focus on the setup and

maintenance of the proxy network, for which existing mechanisms, such as those used in peer-to-peer networks [47], [10] and cycle-sharing systems [35], [6] can be used.

3 Evaluation of Proxy Benefits

In this section, we evaluate the benefits of using a proxy network as described above, both in terms of the potential performance opportunities as well as through the impact on Montage, a high-performance computing application. Since the pure computation benefits of external compute nodes is well established by the Grid and other cycle-harvesting systems such as Condor and @Home, we primarily focus on the network benefits of proxies, in terms of routing and cloud service interaction. At the same time, we also evaluate the benefit of using proxies in the role of data transformation, particularly for resource-constrained clients. Our experiments are conducted on PlanetLab [12], however, potential network optimization opportunities would likely extend to the Internet at large as noted by other research groups [50], [44].

3.1 Performance Improvement Opportunities

We first present results using microbenchmarks to identify the potential opportunities for performance improvement using proxies.

Proxies in networking role We first examine the opportunity by using proxies in a networking role, either to achieve a better path for cloud service interaction, or for routing data more efficiently to other application components. Note that the use of a proxy between the client and the server/cloud introduces an additional hop at the TCP/IP layer of the networking stack. Despite this, certain proxies exist that accelerate network communication by a substantial factor. [36], [55] explain the occurrence of conditions that the proxies may exploit. To evaluate this opportunity, we have constructed a network dashboard tool[2] for PlanetLab that provides us with the networking statistics for potential proxy nodes in PlanetLab.

Each proxy executes a resource monitor that collects and transmits monitoring data to the entry points and the dashboard control. The resource monitor periodically probes all proxies to measure the following entities - bandwidth (for TCP streams and UDP datagrams), the delay in the arrival of successive datagrams, and the variation of this delay, also known as jitter. Next, we present some interesting results collected by the tool.

We notice that proxies could improve the network characteristics of a number of network endpoints. Approximately 1600 pairs of network endpoints were monitored. We calculated the number of alternate paths, formed by routing data through a single proxy, that are superior to the direct path. This analysis was

[2] Network Dashboard Tool URL: http://proxy.cs.umn.edu

carried out for TCP streams only, as it is the primary protocol used for data and file transfer. The aggregate summary is presented in Figure 3. On average, there exist a large number of alternate paths that may benefit a given pair of network endpoints. Furthermore, the benefit of these paths remains constant over a long duration, suggesting that these opportunities - the benefit of alternate paths - are long lived. Looking at these aggregate values, one notes that about 80% of the alternate paths are faster than the direct ones by at least 10%. In the following, we show that specific paths can be accelerated far more.

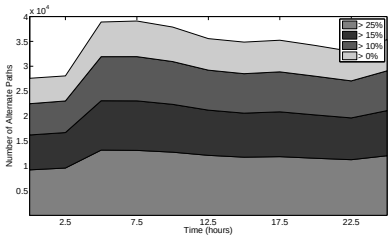
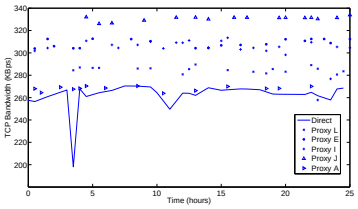


Fig. 3: For a set of 1600 pairs of network endpoints, we plot the number of alternate paths that could improve the TCP bandwidth. The data plotted was collected over a day, and is discretized at intervals of 2.5 hours. For each interval and each pair of network endpoints, approximately 40 paths were analyzed, leading to a total of 64k paths analyzed per interval. Each curve indicates the average TCP acceleration as a percentage.

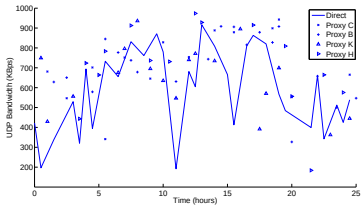
Next, we drill down and analyze the benefits of introducing a proxy for a pair of network endpoints communicating using a TCP stream. Our observations in Figure 4a show that it is possible to accelerate TCP streams using proxies, ranging from 5-25%. Numerous proxies also provide a stable and sustained improvement over time. The performance benefit observed if one uses such proxies could be easily computed from historical data with a high degree of confidence.

Initially, these benefits could be immediately exploited by any distributed application executing over a WAN, where the time to transfer data to and from compute nodes is a non-trivial part of the total execution time. Indeed, as aggressive parallelization continues to reduce the computation time of applications at all levels from multi-core to wide-area Grids, communication becomes increasingly the bottleneck, particularly for data-intensive applications.

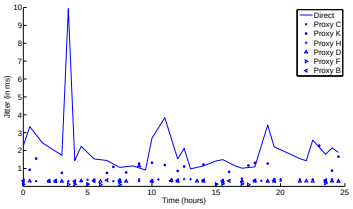
Next, we examine the benefits of using proxies for the UDP protocol. Many multimedia streaming and remote visualization applications utilize the UDP protocol for data transfer. The use of a proxy can amplify the observed UDP bandwidth between two end-points. We present an example of one such experiment (of many examples) in Figure 4b, where the alternate paths accelerate data transfer by 6-50%. Continuing with the benefits seen for the UDP protocol, one may reduce the network jitter and delay (Figures 4c and 4d) between the endpoints by using a proxy, to under 10% of the original values. This would benefit human-in-the-loop applications by reducing network delay and jitter for data transmissions. Thus, proxies can be used to reduce the sensitivity of the



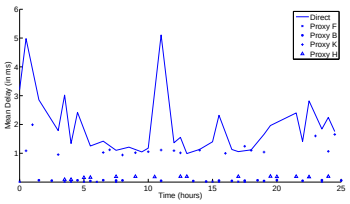
(a) TCP bandwidth (KBps) (Higher is better)



(b) UDP bandwidth (KBps) (Higher is better)



(c) UDP delay (ms) (Lower is better)



(d) UDP jitter (ms) (Lower is better)

Fig. 4: These plots show the networking benefit of using proxies to route data between a particular pair of nodes, for both the TCP and the UDP protocol. The solid line plots values for a direct network path between the endpoints. Other points on each plot indicates values that may be realized by routing data through certain proxies. The plot incorporates data sampled at different times over a day, with more recent values appearing on the right.

end user to the vagaries of the network. These observations mirror those seen for the TCP protocol - proxies could be used to improve the network performance and network QoS by a substantial amount that is sustained over time.

The next question is whether the communication performance is predictive. That is, can we exploit prior measurements to select proxies for future optimizations? For this experiment, three pairs of network endpoints are chosen at random. A fixed amount of data (2 MB) is transferred between these pairs using TCP. We measure the bandwidths for both a direct transfer, and for data routed through a proxy. These 'instantaneous' values of bandwidth are compared with observed measurements over the last 2.5 hours. We employ a simplistic predictor, the prediction is based on the average of values recorded in the last few hours. The difference between the predicted and instantaneous value is the error in our predictor. The speedup obtained is the difference between the instantaneous values observed with and without proxies. Since it is notoriously difficult to accurately predict bandwidth due to network volatility, we use the historical values as a hint. A subset of the results - experiments conducted for three pairs of network endpoints - are summarized in Figure 5. The figure shows that the best proxy is able to achieve a 56% improvement, while the prediction error is in the range of 1.6% to 54%. A large error in some predictions can be attributed to the simplistic prediction model employed by our technique. More sophisticated models have been developed [15], [39], and can be used to improve the predictor. Nevertheless, the introduction of proxies leads to an improvement in the bandwidth for most cases.

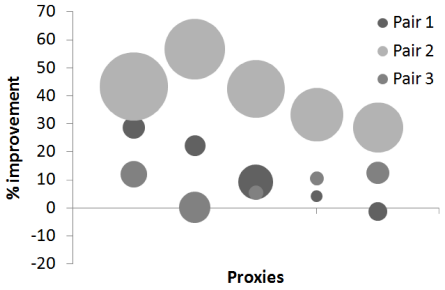


Fig. 5: This plot shows the benefit of routing data through proxies. 6 different proxies were chosen for each of the 3 network endpoints, and are shown on the x-axis. The percentage improvement of using these proxies form the y-axis. The size of each blob on the chart indicates the error in the values predicted.

Proxies in data transformation role We now present the potential benefit of using proxies in a computing role for data transformation (compression, filtering, etc.). To show this, we use a resource-constrained client, an Android phone. The major functions that drain power on a mobile phone are 3G, Wifi, Bluetooth, and GPS transmission. By reducing the amount of data transmitted we can save both space and battery power on the mobile device.

The purpose of this experiment was to test the general opportunity of using proxies to perform compression on behalf of mobile applications, which are in-

creasingly being used by end-users to perform a large number of client-side functions. We developed proxies to compress files and send them to a mobile phone. We tested a series of jpg images, Ebooks, and Binaries for their compressibility. We ran the mobile application on an emulator to get more consistent network results. The T-Mobile G1 phone was emulated. We set the network speed to UTMS/3G.

Figure 6 shows the amount of compression we were able to achieve for these sets of files. The top of each column shows the original file size. Within each column is a black subcolumn that shows the size of the file after compression. The jpgs were resized by the proxy to better match the resolution of the mobile phone. This operation was lossy and produced a large size reduction, 95-99%. The ebooks (text files) and binary files were compressed losslessly with the gzip algorithm. They had a size reduction of 61-64% for the ebooks and 54-70% for the binaries.

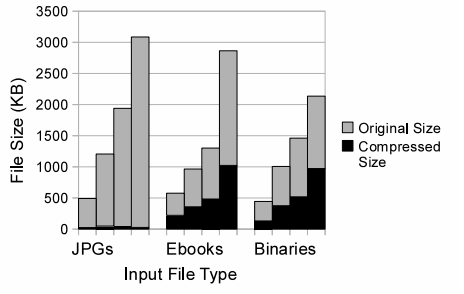
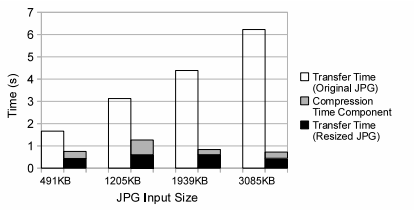


Fig. 6: This plot shows the reduction in size when a proxy is used to compress jpg images, Ebooks, and binary files

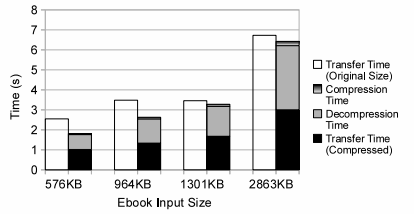
Figure 7 shows the download times of the compressed files vs the uncompressed files. In most cases we were able to see a significant reduction in download time. The jpg files saw a 121-761% speedup in download time. The ebooks saw a 5-41% speedup. The binaries ranged from 6% slower to 52% faster. In the case where the downloading was slower the decompression time seemed to be the most significant factor. The mobile phone has less cpu and memory resources to decompress than a typical desktop computer. In the case of the T-Mobile G1, the heap space for an application is limited to 16MB. The cpu runs at 528Mhz [26]. In all cases the compression time was low because the proxy had enough resources to compress quickly.

3.2 Application Performance Improvement

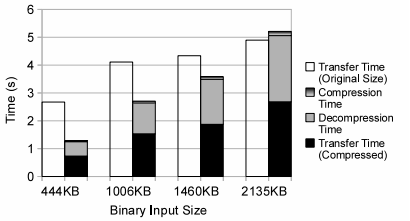
In this section, we demonstrate how proxies can be used to benefit a real-world distributed application. We present a set of experiments using Montage, a software suite for creating image mosaics. We aim to create a mosaic of the sky by sourcing images of different quadrants from the SkySurvey servers [45]. Montage illustrates all of the potential optimizations illustrated earlier (Figure 2). We show that proxies can speed up the runtime of the Montage application by



(a) This plot shows the download and compression times when a proxy is used to compress (lossy) and resize jpg files



(b) This plot shows the download, compression, and decompression times when a proxy is used to losslessly compress Ebook files before uploading them to the mobile client.



(c) This plot shows the download, compression, and decompression times when a proxy is used to losslessly compress binary files before uploading them to the mobile client.

Fig. 7: Mobile Compression Opportunity

reducing the data transfer times from the SkySurvey servers to the compute nodes. In addition, we show how proxies can benefit the delivery of output data to a user either for archival or for real-time visualization, where the user may be either on a desktop machine or on a resource-constrained mobile client. Finally, we show how proxies can accelerate the "internals" of the application (not just the input and output phases) by speeding up the data transfer between different stages of a distributed Montage workflow.

We believe that our scheme is not limited to Montage alone, and can be used to accelerate other data intensive distributed applications, such as MapReduce on a WAN, etc. The remainder of the section thoroughly examines the benefits of using proxies to accelerate the Montage workflow (Figure 2).

Input Stage Optimization Here, we show how proxies can be used to reduce the time to retrieve data from the SkySurvey servers to the nodes that compute a mosaic using Montage. For the sake of simplicity, we initially assume that each compute node performs all stages of the image mosaic operation. Thus, we concentrate on acceleration of the input delivery phase from SkySurvey servers. This type of acceleration is applicable to any application that needs to retrieve data from a number of geographically disperse locations, like blog analysis, distributed data mining, etc.

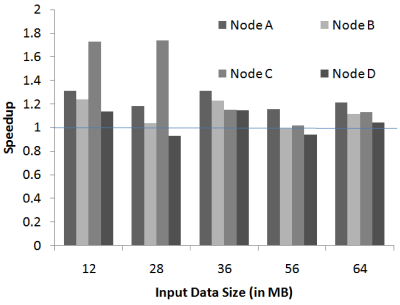
We performed experiments over PlanetLab nodes located across multiple continents. We observed the following:

- The bandwidth observed between PlanetLab nodes in North America to the SkySurvey servers are magnitudes larger than the same for nodes outside the continent.
- The bandwidth observed between PlanetLab nodes outside North America to those within are greater than their bandwidth to the SkySurvey servers.

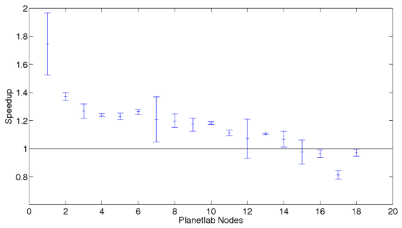
Thus, for each PlanetLab site outside the continent, our proxy is a site in the continent that has the best connectivity to that site. Our experiments with such a setup are described in the following paragraphs.

We observed that a good proxy is usually insensitive to the size of data being routed through it. The speedup obtained is constant for different data sizes. This is illustrated in Figure 8a. We selected a set of 4 PlanetLab nodes, and downloaded datasets of different sizes from the SkySurvey servers. Speedups of 18 - 31% are achieved by the more stable proxies.

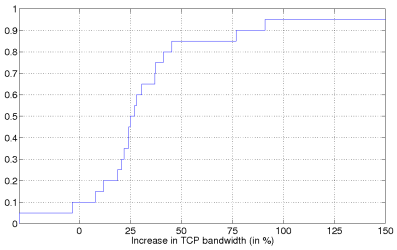
We proceed to investigate the stability of proxies over time, i.e. are the speedups repeatable over a long period? For this experiment, we download a dataset of a fixed size on to a set of PlanetLab nodes, a number of times over 24 hours. We compare the times to download data with and without proxies to accelerate the transfer. The results are plotted in 8b. We see that a number of proxies exhibit a sustained positive speedup over the duration of a day. Also, by the size of the confidence intervals, we see that the variation in this speedup is fairly small for a number of proxies, indicating that these speedups are fairly stable over time.



(a) Speedup observed when proxies route the transfer of different input image-sets for the Montage application



(b) The speedup observed over time, when proxies accelerate communication between PlanetLab nodes (y-axis) and the SkySurvey web-servers. This plot shows the average values (with 95% confidence intervals) by which the data transfer can be accelerated using proxies, observed over a 24 hour period.



(c) This plot shows the cumulative probability distribution (CDF) of the % increase in TCP bandwidth when proxies accelerate data transfer.

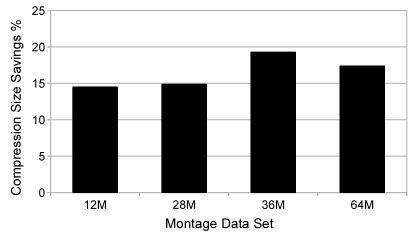
Fig. 8: Speedup and consistency seen when proxies are used to accelerate the transfer of data sets from the SkySurvey servers to different PlanetLab nodes outside the continental US.

We observe that proxies improve the network performance for numerous compute nodes. The improved performance is sustained over time. To see if this benefit is realized for many nodes, we experimented by downloading a dataset of fixed size to a large number of PlanetLab nodes. We compare the increase in bandwidth that is made possible by routing data through proxies, and summarize the results in Figure 8c. We note that the transfer times for approximately half the cases are improved by at least 25% when proxies are used to accelerate the data transfer.

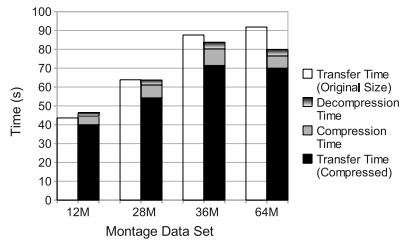
To summarize, we have shown that proxies can be used to accelerate data transfer for a large number of nodes. The speedup obtained is substantial and is stable over the period of a day.

Output Stage Optimization We now present results for optimizing the application's output stage, namely, delivering the output data to the user. Here, we consider two scenarios: (i) where the user is on a wired desktop machine, and (ii) where the user is on a mobile phone. Montage produces large image files, and so the critical requirement in both scenarios is to reduce the network overhead for data transmission. In addition, for the mobile client scenario, we would like to present the output image to the user in a size and resolution appropriate to the mobile device. We show how proxies can be used to achieve these goals. For these experiments the proxies were colocated with the data on the montage output node.

Desktop Client:



(a) This plot shows the % reduction in size when a proxy is used to losslessly compress the output *fits* files before uploading them to the desktop client.



(b) This plot shows the download, compression, and decompression times when a proxy is used to losslessly compress the output *fits* files before uploading them to the desktop client.

Fig. 9: Desktop Client

The Montage process outputs large image files with the *fits*[3][25] extension. These are the same types of files used as input to the Montage process. *Fits* files

[3] Flexible Image Transport System

can be converted into jpg files by a utility that comes with the Montage software suite. It may also be desirable to keep the output images in *fits* format to prevent image data loss and to reuse the images as input in a Montage process later. A proxy could cache this data for later optimization, e.g. it might be useful to another Montage execution avoiding some recomputation if regions of interest overlap. The drawback to keeping the images in *fits* format is their large size. Therefore it may be desirable to compress the *fits* images losslessly prior to transport to the desktop end client for storage.

For this experiment we created a proxy that compressed the images with a gzip algorithm and transfered them to a desktop client. Figure 9a shows the size savings ranged from 14.5-17.38% for the output images generated earlier (associated with the input images in Figure 8a). The *fits* images used as input for compression were 23.57MB, 34.85MB, 47.81MB, and 45.72MB in size.

Figure 9b shows the download times for the images with and without compression. The downloading was done over a 5 Mbps link. There is a tradeoff between space savings and download/compression time. On average it took 6.55 seconds to compress an image and 2.8 seconds to decompress. This overhead made compression more useful for the larger images.

Mobile Client:

It may also be desirable to view the image outputs remotely via a mobile device if the user is on the go. To this end we created a mobile application written for the Google Android platform.

Android is an open source software environment for mobile phones. It was developed by Google in conjunction with the Open Handset Alliance [19]. Most applications for Android are written in Java. We chose to develop our mobile applications on Android because of the ease of development. Android also has the benefit that the development tools are completely free.

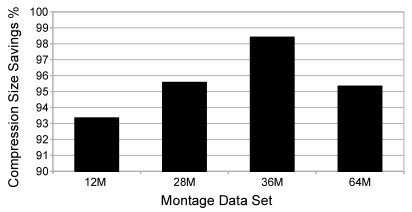
The proxy reduces the resolution of the images to 225 pixels width, while preserving the aspect ratio of the image. This results in significant space savings while still producing an output image with a viewable resolution for a phone. The phone displays the image below the transfer button.

The images were converted from *fits* format to jpg format using the utility that comes with Montage. They were converted using the highest quality settings initially. The resulting sizes were 392KB, 1136KB, 1272KB, and 565KB. These images were at full resolution and were used as input to the proxy. The resolutions ranged from 1504x2054 pixels to 2735x2191 pixels. These resolutions are too large for the typical desktop computer let alone a mobile phone.

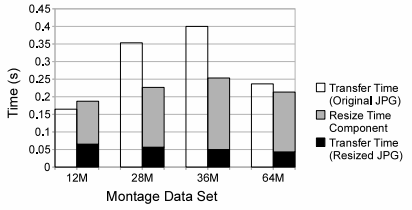
Figure 10a shows the compression savings from resizing the images. The compression savings were extremely high, ranging from 93.36% to 98.42%. This results in a significant space savings if multiple images are stored on the phone for later viewing.

Figure 10b compares the download times for images with and without compression. For these downloads the link was set at optimal 3G/UTMS download speeds and latency. This was controlled through the settings for the Android emulator. The bandwidth a mobile phone receives typically isn't optimal and

can vary from location to location. The speedup varied from 11% slower for the smallest file to 60% faster for the largest file, indicating that compression is still desirable for the larger images. However, when we look at the breakup of the download time, we see that compression can take up a significant portion (65-80%) of the total time. Since compression occurs at the proxy, the actual network transmission time for the image is much smaller (13-40% of that for uncompressed image). Reducing the network transmission time is critical for a mobile client as it also saves precious bandwidth and power, showing the benefit of using proxies across all file sizes.



(a) This plot shows the % reduction in size when a proxy is used to compress (slightly lossy) and resize the output jpg files.



(b) This plot shows the download and compression times when a proxy is used to compress (slightly lossy) and resize the output jpg files.

Fig. 10: Mobile Client

Distributed Workflow Optimization Finally, we show how proxies can be used to accelerate the data transfer between different stages of a distributed application. Again, we consider the Montage application. The Montage application comprises of a number of stages that are executed serially one after another. Montage has a few compute intensive operations - *mProjExec*, *mDiffExec*, *mFitExec*, which take several seconds to minutes to complete. Parallelism in Montage can be obtained by running multiple instances of these operations on a partitioned input set, on different machines. Thus, a typical parallel execution of the Montage application will have a high fan-out and fan-in for certain stages. Planners such as Pegasus [18] are used to create workflows that can be utilize the parallelism available in grid environments.

For this experiment, we execute each compute intensive stage of the Montage application on a different machine. The other stages of the Montage application are distributed over these machines. The output of each stage is sent to the next stage. Proxies are used to accelerate data transfer between stages of Montage that are executed on different physical machines.

This setup illustrates the most conservative improvement one can achieve by deploying the Montage application over a WAN. A deeper parallel execution of the stages that comprise Montage will further reduce the computation time,

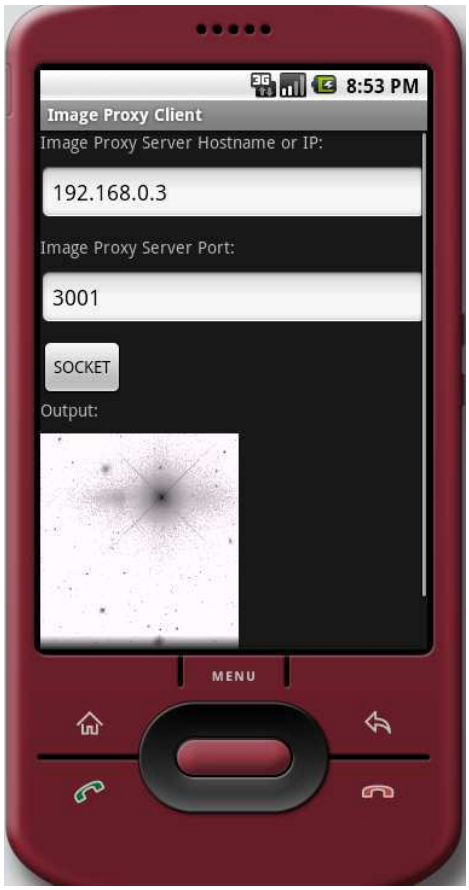


Fig. 11: Screenshot of T-Mobile G1 displaying Montage output.

and thus, the networking benefits brought about by proxies will be even more pronounced.

The results are shown in Figure 12. The input set, of size 36MB, is present on the machine that executes the first compute intensive stage of Montage. This stage generates a set of intermediate files, totaling around 118MB. This data is copied over the network to another machine that executes the next few stages of the Montage workflow. The copying phase is accelerated by 42% if data is routed through a proxy. The intermediate files generated by next few stages, totaling 369MB, are then copied to another machine that executes the final few stages. When data is routed through a proxy, we see an improvement of 24% in the completion time of this phase. The final stage of Montage generates a single output *fits* image, of size 24MB. Thus, by using proxies to accelerate the network-intensive tasks of the workflow, we are able to reduce the total execution time of the workflow by 13%. Again, the benefits seen are likely to increase as the internal stages are parallelized, leading to a reduction in the computation time.

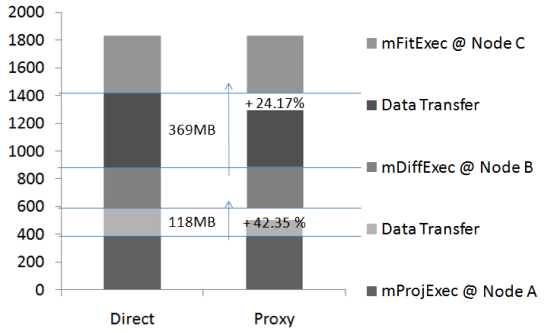


Fig. 12: This plot shows the % improvement when proxies are used to accelerate communication between different stages of Montage running on different nodes. The y-axis shows the total execution time, in seconds.

4 Related Work

Our work is a synthesis of multiple ideas, spanning different disciplines in computer science – data intensive programming systems, cloud computing, network computing, Web 2.0, volunteer systems and proxy architectures.

Many emerging cloud systems such as Amazon EC2/S3 [17] and Google Apps [23] provide virtual resources to third-party applications. There are many data clouds such as Google Earth [24] and Sloan Digital Sky Survey [45] that serve useful data to end-users. These systems are optimized for data-intensive computing within a single cloud or data center, however, a challenge is transferring the underlying data and the results to or from these clouds.

A number of papers and scholarly articles, most notably [49], [30], advocate an intelligent network with computations on data streams in the network fabric itself. This scheme has been adapted for protocol conversion [49], caching [9],

transcoding data [2], remote execution [32] and even routing optimization [38], [29]. Although our model is logically similar, we operate at the application layer of the protocol stack. We make no assumptions about the network, and treat it like a black box. We seek to address the different set of problems at the application layer of the protocol stack by exploiting the location and diversity of a proxy network comprised of volunteered resources.

Volunteer edge computing and data sharing systems are best exemplified by Grid and peer-to-peer systems including, Kazaa [42], Bittorrent [13], Globus [21], BOINC [5], and @home projects [37]. These systems provide the ability to tap into idle donated resources such as CPU capacity or aggregate network bandwidth, but they are not designed to exploit the characteristics of *specific* nodes on behalf of applications. We aim to integrate the best of these techniques in the overlay network we setup between the actors in our system – end clients, proxies, data sources and clouds.

Overlay networks can provide greater reliability and improve performance of the networking components. [3] details a scheme allowing distributed Internet applications to detect and recover from path outages, and re-route around failures. They illustrate the benefits of moving control of routing to the application layer by providing a more resilient system. [20] support this view and provide evidence of improved robustness and scalability of the network. [4], [33] improve client availability by using multi-homing and cooperative overlay networks to find and use a larger number of paths to the server. The proxy network would encompass these techniques, thus providing a highly resilient network to all end users. Similar systems are described in [41] and [54].

Performance improvements using overlay networks are brought about by exploiting triangle inequality in networks and selecting better paths. [36] provide an excellent study of interactions occurring in the routing layer, and their effects on applications. Triangle inequalities in networks are a common feature of the current routing landscape [55] and the projects like Detour [43] exploit these inequalities to provide superior network paths. The results we observe in this paper are consistent with existing literature. Through our work, we demonstrate the usefulness of proxies in such a setting. [48] improves performance in grid environments by using store and cooperating forwarding techniques. They demonstrate how "logical depot" can be used to accelerate network components of a distributed grid application. Similarly, we show how the network components of a HPC application – a Montage workflow – can be accelerated using volunteer donated resources. We differ from the existing implementations in the sense that our architecture is highly dynamic and comprised of donated resources.

Estimating network paths and forecasting future network conditions are addressed by [51]. We have used simple active probing techniques and network heuristics for prototyping and evaluation of network paths in our experiments. Existing tools [40], [16], [31] would give us a more accurate view of the network as a whole. Direct probing in a large network isn't scalable, and we advocate the use of passive or secondhand measurements [28]. [27] shows that it is possible to infer network conditions based on CDN [1] redirections and [11] is an imple-

mentation of such a scheme. Such techniques can be easily used by end clients to identify proxies in close network proximity, without any point-to-point probing. We aim to integrate a number of these techniques into the proxy network.

Many Internet programming paradigms and systems have been emerging. Relevant to our project are systems that support content mashup and state sharing, such as Yahoo Pipes [52], GME [22], and LiveMesh [34], which can be used to connect applications and/or distributed data. While most of these systems provide abstractions to build distributed applications, they typically rely on underlying support from the network, service provider, or end hosts for their communication and execution needs.

Finally, Web proxies [46] and CDNs [1] have been widely deployed for Web caching and Web content distribution. However, these systems deal mostly with data caching and replication, and the amount of data itself (per request) is fairly small and originates in a central location. Moreover, there is no further processing or distribution/merging requirements on the data, so that there are no major computation or massive bandwidth needs for such systems.

Our approach is novel in that proxies may assume a diverse set of roles unlike other systems in which network nodes either compute, route, serve data, or invoke services. Enabling proxies to assume multiple roles is key to the performance of distributed data-intensive applications.

5 Conclusion

In this paper, we explored the potential of middleware deployed on a *proxy network* for boosting the performance of distributed HPC applications. The proxy network sits between the cloud and the end-user application offering resources that mitigate performance bottlenecks due to resource sharing and poor network connectivity. Our approach is a synthesis of network optimization, routing, an in-network computing to accelerate complex distributed workflows. We created a tool that helps us identify and eliminate network bottlenecks by smart routing and perform in-network computations to boost application performance. While we focused on HPC applications that access external clouds, the proxy network can be beneficial for any distributed applications that contain a geographically-dispersed set of components.

To demonstrate the potential of proxies, we performed experiments on PlanetLab, exploiting its resource and network diversity. The results revealed that proxies can improve many aspects of network performance including latency, throughput, and jitter. Using Montage as an application exemplar, we showed how input, output, and internal data transmission can be optimized in a client-specific way. Future work will focus on the development of automated proxy selection algorithms for applications given the roles that they were chosen to play.

References

1. Akamai. `http://www.akamai.com`.

2. E. Amir, S. Mccanne, and H. Zhang. An application level video gateway. pages 255–265, 1995.
3. D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris. Resilient overlay networks. Technical report, 2001.
4. D. G. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. N. Rao. Improving web availability for clients with monet. In *In Proc. 2nd Symposium on Networked Systems Design and Implementation (NSDI*, 2005.
5. D. P. Anderson. BOINC: A System for Public-Resource Compting and Storage. In *Proceedings of the 5th ACM/IEEE International Workshop on Grid Computing*, 2004.
6. A. Awan, R. Ferreira, S. Jagannathan, and A. Grama. Unstructured Peer-to-Peer Networks for Sharing Processor Cycles. *Journal of Parallel Computing*, 2005.
7. P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. H. an d Rolf Neugebauer, I. Pratt, and A. Warfield. Xen and the Art of Virtualization. In *Proceedings of the Nineteenth ACM Symposium on Operating systems Principles*, October 2003.
8. A. Barker, J. B. Weissman, and J. van Hemert. Orchestrating Data-centric Workflows. In *IEEE/ACM CCGrid International Symposium on Cluster Computing and the Grid*, 2008.
9. C. M. Bowman, U. Manber, P. B. Danzig, M. F. Schwartz, D. R. Hardy, and D. P. Wessels. Harvest: A scalable, customizable discovery and access system, 1995.
10. Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker. Making Gnutella-like P2P Systems Scalable. In *Proceedings of ACM SIGCOMM*, Aug. 2003.
11. D. Choffnes and F. Bustamante. On the effectiveness of measurement reuse for performance-based detouring. In *INFOCOM 2009, IEEE*, pages 693 –701, april 2009.
12. B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman. PlanetLab: an overlay testbed for broad-coverage services. *ACM SIGCOMM Computer Communication Review*, 33(3):3–12, July 2003.
13. B. Cohen. Incentives build robustness in BitTorrent. In *Proceedings of the First Workshop on the Economics of Peer-to-Peer Systems*, June 2003.
14. Condor project. `http://www.cs.wisc.edu/condor/`.
15. F. Dabek, R. Cox, F. Kaashoek, and R. Morris. Vivaldi: a decentralized network coordinate system. In *SIGCOMM'04*, pages 15–26, 2004.
16. A. Downey. Using pathchar to estimate internet link characteristics. In *In Proceedings of ACM SIGCOMM*, pages 241–250, 1999.
17. EC2/S3. `http://aws.amazon.com`.
18. E. D. et. al. Pegasus: A Framework for Mapping Complex Scientific Workflows onto Distributed Systems. *Scientific Programming Journal*, 13(3):219–237, July 2005.
19. F. Ableson, C. Collins, and R. Sen. *Unlocking Android: A Developer's Guide*. Manning, CT, USA, 2009.
20. N. Feamster, H. Balakrishnan, J. Rexford, A. Shaikh, and J. V. D. Merwe. The case for separating routing from routers. In *In ACM SIGCOMM Workshop on Future Directions in Network Architecture*. ACM Press, 2004.
21. I. Foster, C. Kesselman, J. Nick, and S. Tuecke. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. In *Proceedings of the Global Grid Forum*, June 2002.
22. Google mashup editor. `http://code.google.com/gme/`.
23. Google apps. `http://www.google.com/apps`.

24. Google earth. http://earth.google.com.
25. R. J. Hanisch, A. Farris, E. W. Greisen, W. D. Pence, B. M. Schlesinger, P. J. Teuben, R. W. Thompson, and A. W. Iii. Raytheon ITSS.
26. Htc - products - t-mobile g1 - specification. http://www.htc.com/www/product/g1/specification.html.
27. A. jan Su, D. R. Choffnes, A. Kuzmanovic, and F. E. Bustamante. Drafting behind akamai (travelocity-based detouring. In *In Proceedings of ACM SIGCOMM*, pages 435–446, 2006.
28. J. Kim, A. Ch, J. Weissman, J. Kim, A. Ch, and J. B. Weissman. November 24, 2008open: Passive network performance estimation for data-intensive applications.
29. L. Kleinrock. Nomadic computing. In *Keynote Address: International Conf. on Mobile Computing and Networking (MOBICOM*, 1995.
30. T. Koponen and A. Ermolinskiy. A data-oriented (and beyond) network architecture. In *In submission*, 2007.
31. K. Lai and M. Baker. Measuring link bandwidths using a deterministic model of packet delay. In *in Proceedings of ACM SIGCOMM*, pages 283–294, 2000.
32. M. T. Le, S. Seshan, F. Burghardt, and J. Rabaey. Software architecture of the infopad system. In *In Proceedings of the Mobidata Workshop on Mobile and Wireless Information Systems*, pages 20–0, 1994.
33. Z. Li, P. Mohapatra, and C. nee Chuah. Virtual multi-homing: On the feasibility of combining overlay routing with bgp routing. In *In Proc. of Networking 2005*, pages 1348–1352, 2005.
34. Live mesh. http://www.mesh.com.
35. V. Lo, D. Zappala, D. Zhou, Y. Liu, and S. Zhao. Cluster Computing on the Fly: P2P Scheduling of Idle Cycles in the Internet. In *Proceedings of the IEEE Fourth International Conference on Peer-to-Peer Systems*, 2004.
36. C. Lumezanu, Y. Baden, N. Spring, and B. Bhattacharjee. Triangle inequality and routing policy violations in the internet.
37. D. Molnar. The SETI@Home problem. *ACM Crossroads*, Sept. 2000.
38. D. Mosberger and L. L. Peterson. Making paths explicit in the scout operating system. pages 153–167, 1996.
39. Network Weather Service. http://nws.cs.ucsb.edu/ewiki/.
40. A. Pasztor and D. Veitch. Active probing using packet quartets. In *In ACM SIGCOMM Internet Measurement Workshop*, pages 293–305, 2002.
41. D. A. Rohit, D. G. Andersen, and R. N. Rao. Principles for end-to-end failure masking, 2003.
42. S. Saroiu, K. P. Gummadi, R. J. Dunn, S. D. Gribble, and H. M. Levy. An Analysis of Internet Content Delivery Systems. In *Proceedings of Symposium on Operating Systems Design and Implementation*, Dec. 2002.
43. S. Savage, T. A. A. Aggarawl, T. Anderson, A. Aggarwal, D. Becker, N. Cardwell, A. Collins, E. Hoffman, J. Snell, A. Vahdat, G. Voelker, and J. Zahorjan. Detour: a case for informed internet routing and transport. *IEEE Micro*, 19:50–59, 1999.
44. S. Savage, A. Collins, E. Hoffman, J. Snell, and T. Anderson. The end-to-end effects of internet path selection. In *Proceedings of the ACM SIGCOMM Computer Communication Review*, Aug 1999.
45. Sloan digital sky survey. http://www.sdss.org/.
46. Squid proxy cache. http://www.squid-cache.org.
47. I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Pastry: Scalable, decentralized object location. In *Proceedings of the ACM SIGCOMM*, Aug 2001.

48. M. Swany. Improving throughput for grid applications with network logistics. In *In SC '04: Proceedings of the 2004 ACM/IEEE conference on Supercomputing*, page 23. IEEE Computer Society, 2004.
49. D. L. Tennenhouse, J. M. Smith, W. D. Sincoskie, D. J. Wetherall, and G. J. Minden. A survey of active network research. *IEEE Communications Magazine*, 35:80–86, 1997.
50. G. Wang, B. Zhang, and T. S. E. Ng. Towards network triangle inequality violations aware distributed systems. In *Proceedings of the ACM/USENIX Internet Measurement Conference (IMC'07)*, Oct 2007.
51. R. Wolski, N. T. Spring, and J. Hayes. The network weather service: A distributed resource performance forecasting service for metacomputing. *Journal of Future Generation Computing Systems*, 15:757–768, 1999.
52. Yahoo pipes. `http://pipes.yahoo.com/`.
53. B. Yee, D. Sehr, G. Dardyk, J. B. Chen, R. Muth, T. Ormandy, S. Okasaka, N. Narula, and N. Fullaga. Native client: a sandbox for portable, untrusted, x86 native code. In *Proceedings of IEEE Security and Privacy*, 2009.
54. A. S. Yip. Natron: Overlay routing to oblivious destinations, 2002.
55. H. Zheng, E. K. Lua, M. Pias, and T. G. Griffin. Internet routing policies and round-trip-times. In *In PAM*, 2005.