NATRON: Overlay Routing to Oblivious Destinations

by

Alexander Siumann Yip

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Masters of Engineering in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

August 2002

Author .....
Department of Electrical Engineering and Computer Science
August 22, 2002

Certified by .....
Robert T. Morris
Assistant Professor
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

NATRON: Overlay Routing to Oblivious Destinations

by

Alexander Siumann Yip

Submitted to the Department of Electrical Engineering and Computer Science
on August 22, 2002, in partial fulfillment of the
requirements for the degree of
Masters of Engineering in Computer Science and Engineering

Abstract

This thesis presents NATRON, a system in which an overlay network of nodes cooperates to improve unicast routing to non-participating hosts. Previous overlay systems have used overlay routing to improve communication between participating hosts; they were unable to exploit overlay routing to non-participating hosts. NATRON uses a combination of IP tunneling and network address translation to allow members of the overlay to communicate with hosts outside the overlay network via other overlay members.

In order to estimate the potential performance improvement a system like NATRON might provide, we performed an exhaustive test on a multi-site Internet testbed. Our results show that a system that always guesses the best intermediate node could reduce the average HTTP transfer time by 18% and reduce the number of downloads lasting longer than 30 seconds by 16%. We implemented a working NATRON and a heuristic for choosing intermediate overlay nodes, but we find that our heuristic can only exploit 22% of the potential performance gains. We conclude that overlay routing to oblivious hosts has good potential for performance enhancement but further work is needed to develop a path choice heuristic.

Thesis Supervisor: Robert T. Morris
Title: Assistant Professor

Acknowledgments

This thesis could not have been completed without the support and assistance of many friends and colleagues.

I am particularly indebted to my advisor, Robert Morris, for his persistent guidance and help; somehow, he kept my efforts focused. Without him, this work would have impossible.

The remaining members of PDOS have been a great help whenever I needed it. When I had questions about Click, Benjie and Eddie had the answers. When I was faced with heaps of data, Chuck supplied statistical expertise and finally, when my desktop was overloaded with windows, Thomer supplied `tbiff`.

I would like to thank the many volunteers who host the nodes which make up the RON test-bed and in particular, David Andersen. David maintained the RON test-bed and provided access to the nodes so that I could collect data. In doing so, he endured my inadvertent bandwidth abuse of his personal cable modem.

I'm extraordinarily grateful for my family and friends who gave their loving support along with more than a few healthy distractions from this thesis work.

Finally, I'd like to thank Seanna for her patient understanding and endless supply of hiccups.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis presents NATRON, a technique for improving Internet communication performance between members of an overlay network and hosts outside the overlay. In NATRON, members of an overlay forward data via their overlay neighbors to alter the underlying Internet path used by the data. The goal of NATRON is to exploit situations where the Internet chosen path between two hosts can be improved upon by forwarding data along an alternate path.

Previous work [19, 3] showed that forwarding data through intermediate hosts can improve network performance. Overlay networks were built to exploit the potential but were limited to improving intra-overlay communication. NATRON differs because it allows overlay members to use intermediate nodes when contacting *oblivious* hosts which are outside the overlay network.

1.1 Motivation

Recent work has shown that Internet routing is less than ideal. In [13], Labovitz et al. find that the Internet routing tables may be unstable causing routing failures on the order of minutes while the Internet recovers from a link failure. In [15], Paxson found that 3.3% of the time, routing pathologies such as routing loops and erroneous routing adversely affected end to end path quality.

Routing oscillations and pathologies are typically confined to small regions of the

network because Internet service providers administer their networks independently. The underlying network topology is a mesh of ISP's, so there is path redundancy. Failures can be avoided by using alternate paths through the Internet mesh. Detour[19] and RON[3] were two studies exploring this property of the Internet.

In the Detour study, Savage et al. used traceroute data to show that routing packets through intermediate hosts has the potential to have higher performance than a direct Internet path. They argue that such alternate paths are likely to be beneficial 30-80% of the time.

RON exploits alternate paths and shows that overlay routing can improve unicast between overlay members. RON achieves these gains by forwarding network traffic via intermediate nodes thus changing the underlying Internet route used by the data.

Unfortunately, RON's usefulness is limited because it only improves communication between overlay participants. There are two reasons why non-participating hosts cannot take advantage of the RON. First, non-participating hosts cannot utilize a RON because they are not configured to perform the packet tunneling required by RON. The second reason is that using RON requires members to participate in a thorough link measurement protocol. Every member of the RON must measure round trip time and loss rate to its peers in the RON. The setup cost and overhead of these measurements makes RON impractical for deployment on large scale systems such as the world wide web.

For example, if three hosts, A, B and C make up an overlay network, A can communicate with B via host C. However, A cannot effectively communicate with an outside host such as *www.cnn.com* via either B or C. One solution is to add *www.cnn.com* to the RON, but then it would have to monitor route quality to its three peers in the overlay. This poses a serious scalability problem if the remaining web clients want add *www.cnn.com* to their RONs as well.

1.2 System Architecture

The goal of NATRON is to achieve the performance and reliability benefits of RON when communicating with hosts *outside* the overlay network. A NATRON overlay consists of a few Internet hosts scattered around the Internet. When members contact non-member servers, they have the option of sending their traffic directly to the server or directing their traffic through an intermediate node in the overlay.

NATRON controls packet routing by tunneling packets from their source to an intermediate NATRON node. The intermediate node uses network address translation (NAT) [8] to replace the source address with its own, and forwards packets to their target destination. The use of NAT causes the destination to send reply packets back through the same intermediate node, so that NATRON is able to control the routing of packets in both directions. NATRON's general architecture is inspired by a system described by Collins [5].

1.3 Contributions

In this thesis we describe a system architecture which applies overlay routing to oblivious destinations. We present exhaustive measurements of an Internet test-bed to show the potential performance improvements a system like NATRON might provide. We describe a working NATRON system and a policy for choosing intermediate network nodes. We evaluate our policy and compare it to the potential performance improvements. We conclude that overlay routing to non-participating hosts has the potential to improve download performance, but future work is needed to design path choosing policies.

This thesis is organized as follows: Chapter 2 introduces background information and Chapter 3 motivates NATRON. Chapter 4 describes the overall architecture of the system. Chapter 5 explores the potential for improvement over direct Internet routes and Chapter 6 describes the design, implementation and performance of the routing policy. We describe related work in Chapter 7 and conclude in Chapter 8.

Chapter 2

Background

This chapter explains background information important to the NATRON system. Internet organization and BGP are described to show where NATRON derives performance benefits, followed by an overview of network address translation. Finally, Click, a modular software router, is described to justify its use in the NATRON implementation.

2.1 Border Gateway Protocol

The Internet is a mesh composed of many separately managed autonomous systems (ASes). The largest ASes are Tier 1 Internet service providers (ISP's) such as AT&T, BBN and Qwest; they administer large networks which typically have national and international coverage. Tier 1 ISP's connect to one another to form the Internet backbone and carry a large portion of Internet traffic. The smaller Tier 2 ISP's such as Comcast and AOL typically purchase connectivity from a Tier 1 ISP and provide Internet connectivity to end hosts.

Internet routing is organized as a two-level hierarchy to limit routing complexity. The upper level is BGP [17, 16] which coordinates routing among different ASes. The lower level consists of one interior routing protocol (such as OSPF or IBGP) per AS which coordinates internal routing.

Each autonomous system uses BGP to notify its peers which networks it can reach

so that ASes know how to reach foreign host addresses. The Internet is designed this way to limit the amount of routes each AS needs to calculate and know about. Each AS only needs to know direct routes to the hosts it provides access to. In order to route to host h outside an AS, the AS only needs to know which AS has a route to h; it does not need to know further details like which routers it needs to pass through.

BGP is a path vector protocol meaning that each AS announces which networks it can route to and what route it uses to get there. When an AS receives a route announcement, it adds the destination to its routing table (being careful to remove loops) and sends another announcement to its remaining peers. This way, the routing announcement propagates through the network.

Figure 2-1 illustrates a simplified BGP announcement propagating through the network. Autonomous system A needs to advertise its route to 18.26.4.0/24 so it sends an announcement message to its peer B. B then advertises its connection to 18.26.4.0/24 by sending announcements to C and D. Autonomous system D notifies E and C that it has a route to 18.26.4.0/24 via B, A. C notifies D of its new route via B, A.

Since the modern Internet is so large, BGP must control the frequency and propagation of update messages to avoid flooding the network. BGP version 4 controls updates by delaying and aggregating multiple update messages and by preventing propagation of fast changing routes [4]. These techniques combined with the distributed nature of BGP allow it to scale up but sacrifices update speed and convergence time.

2.2 Network Address Translation

NATRON uses network address translation (NAT) [8] to force client traffic to travel via a particular intermediate node. Performing NAT at the intermediate node makes network traffic appear to originate at the intermediate node, so a non-member host would send reply traffic directly to the intermediate node.

Typically, NAT is used to allow multiple hosts on a private network to share a

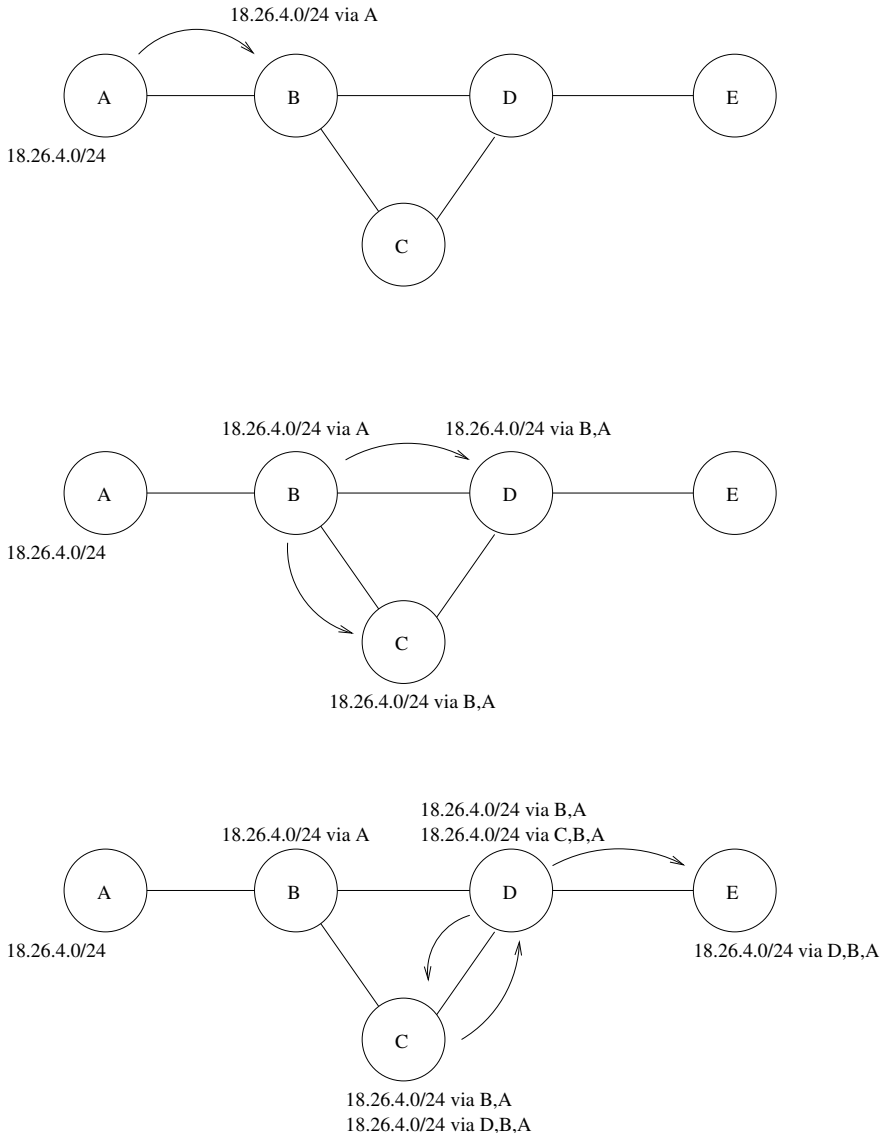

Figure 2-1: Example BGP announcement. Autonomous system A announces its route to 18.26.4.0/24. Updates start at A and propagate through the network until the rest of the ASes hear about the route.

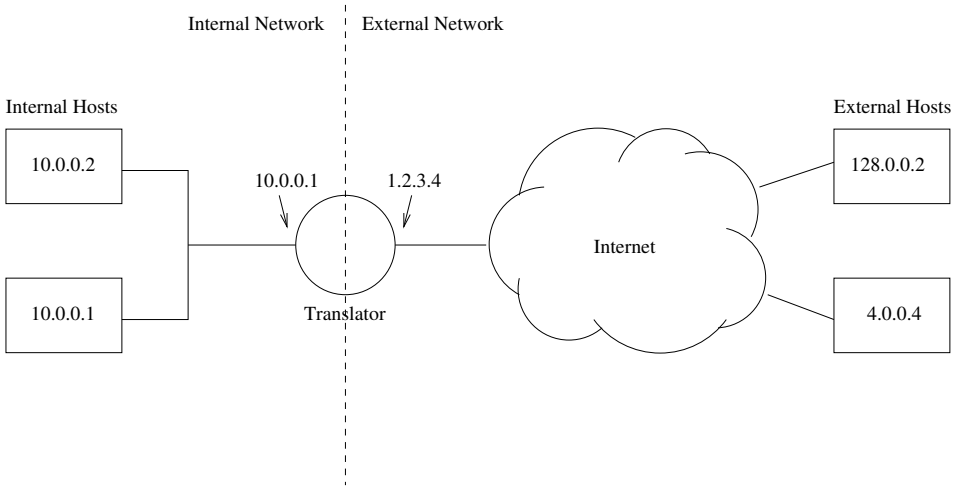

Figure 2-2: A typical configuration for network address translation. In this case, two internal nodes, 10.0.0.2 and 10.0.0.3, share a single external IP address, 1.2.3.4. Packets leaving the internal network are all translated so that their source address is 1.2.3.4. Reply packets are un-translated before they enter the internal network.

common external IP address. NAT is commonly used by home Internet users to share a single broadband Internet connection among multiple internal hosts. NAT makes it possible for many Internet clients to share a single external IP address by exploiting the flow based nature of TCP.

In a typical NAT configuration, internal hosts are assigned local IP addresses which are not accessible by the Internet. Figure 2-2 shows a sample configuration with two internal hosts: 10.0.0.2 and 10.0.0.3 in addition to an address translator acting as the gateway between the internal network and the Internet. The external IP address being shared is 1.2.3.4.

The translator changes the source address and possibly the source port of packets leaving the internal network. For example, if host 10.0.0.2 wants to contact host 128.0.0.2, it emits a packet destined for 128.0.0.2. The translator changes the source address to 1.2.3.4 before sending it out to the Internet. Reply packets from 128.0.0.2 arrive at 1.2.3.4; their destination addresses are un-translated to be 10.0.0.2 and are emitted on the internal network. To un-translate packets, the translator must maintain a translation table mapping internal flows to external flows. Table 2.1 illustrates an example translation table.

Internal SRC IP	Port	DST IP	Port	External SRC IP	Port	DST IP	Port
10.0.0.2	1676	128.0.0.2	80	1.2.3.4	4000	128.0.0.2	80
10.0.0.2	1677	128.0.0.2	80	1.2.3.4	4001	128.0.0.2	80
10.0.0.3	2231	128.0.0.2	80	1.2.3.4	4002	128.0.0.2	80
10.0.0.3	2232	4.0.0.4	80	1.2.3.4	4003	4.0.0.4	80

Table 2.1: A sample NAT translation table for Figure 2-2. In this example, the two internal hosts each have two open sessions.

2.3 Click

Click [14] is an easily configurable software router which runs on commodity hardware. Complex Click routers are assembled by linking primitive packet processing *elements* together; many generic elements are provided with the Click distribution making it a flexible base platform. Additional functionality can be added by writing custom elements in C++. Click can run as a userlevel application to ease development or as a kernel module for improved performance.

Click provides all of the infrastructure needed for packet processing. Additionally, most of the packet processing tasks needed by NATRON such as packet classification, encapsulation and network address translation [12] are already implemented as Click elements. The only additional functionally NATRON required was an element to control path selection. The vast functionality provided by Click and its ease of development made it an easy choice for the NATRON implementation.

Chapter 3

Motivation

This chapter describes the motivation for NATRON including some underlying problems with Internet routing, a study of alternate path routing and a system built to exploit overlay routes.

3.1 Problems with BGP

As described in the previous chapter, BGP must suppress routing updates to prevent route flapping and excessive update messages. The cost of such damping is reduced convergence time and slower route updating.

One problem with BGP is its slow convergence after routing changes. The distributed path-vector protocol results temporary oscillations in routing tables. Labovitz et al. describe these oscillations and argue that the Internet may take tens of minutes to reconverge after a link failure[13]. Although they note that particular implementations are the cause of some delays, they argue that the distributed path-vector protocol has fundamental oscillation and delay problems.

In a separate study, Paxson[15] found in 3.3% of the time, routing pathologies such as outages and route loops stifled end to end communication. Many of those outages lasted longer than 30 seconds.

3.2 Detour

Savage et al. noted that inefficiencies in Internet routing might likely be localized, and that alternate routes may be available, even if routing pathologies disabled certain routes. In their Detour study [18] they analyzed data from 15-39 public traceroute servers to explore the existence of alternate routes that performed better than the default Internet routes. Using the reported loss rates and round trip times averaged over long time scales, they concluded that 30-80% of the time, there was an alternate route which performed better than the default Internet route.

3.3 Resilient Overlay Networks

Andersen et al. showed how to take advantage of alternate routes described above by building Resilient Overlay Networks [3]. They constructed a multi-host overlay network which utilized overlay routing to improve network performance. By continuously monitoring the quality of overlay links, RON is able to optimize for latency, bandwidth or loss rate. RON proved that overlay routing can be used improve network performance between members of an overlay network.

3.4 NATRON

NATRON aims to exploit alternate routes as RON does, but loosens the restriction on the destination host. In RON, both communicating parties must be participating in the same overlay network for the proper packet forwarding to function. RON works well in a peer to peer architecture, but does not lend itself to client-server architectures such as HTTP where web-servers may be unwilling or unable to participate in RONs.

In contrast, NATRON is designed to work in client-server architectures. NATRON makes it possible for overlay members to use overlay routing when contacting non-member hosts.

Chapter 4

System Architecture

The overlay consists of a small number of nodes which are scattered around the Internet in diverse locations, and the goal is to use alternate routes through other overlay members to improve unicast to non-member hosts. Figure 4-1 shows a sample configuration where $N_0 ... N_2$ are members of the overlay and S is a non-participating server. We call intermediate nodes neighbors; in this example, N_1 and N_2 are neighbors of N_0 because N_0 can use both N_1 and N_2 as intermediate nodes.

In one scenario, N_0 needs to open a session to the server S, but the direct path A between them is congested. Paths B and C are uncongested, so N_0 redirects its traffic to N_1 where the traffic exits the overlay, and travels along path C to the server S.

We will define the entry point as the node where a flow enters the overlay and the exit point where the flow leaves. In this example, N_0 is the entry point and N_1 is the exit point. Note that the exit node can be the same as the entry node if the entry node chooses to use the direct route to the server host.

Using network address translation (NAT) [8] at the exit point causes S to send return traffic via N_1, instead of directly to N_0. This gives NATRON a chance to route around slow paths in either direction.

NATRON's use of NAT means that it shares NAT's limitations. First, if a NATRON node fails, the connections using it as an exit point will fail. Second, only participating nodes can initiate connections that are routed through NATRON.

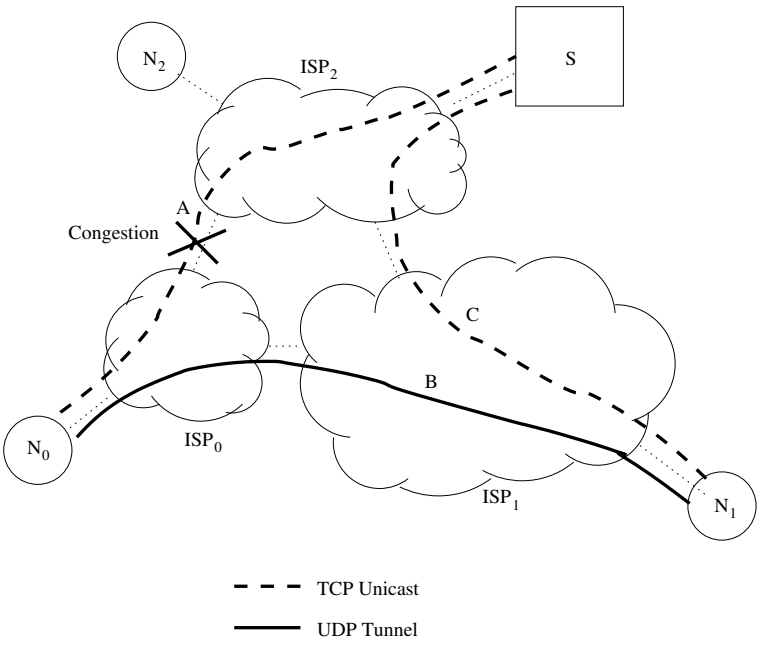

Figure 4-1: Overall NATRON architecture. The direct path from N_0 to S is congested. Client N_0 tunnels to N_1. N_1 performs network address translation, and then forwards traffic to S, bypassing the congestion. Return traffic travels from S to N_1 and then back to N_0.

The procedure used to choose the intermediate node is called the *policy*. The entry node is responsible for executing the policy and choosing which intermediate node to use for each flow. The most difficult part of building NATRON is designing a policy that performs well. The biggest difficulty in designing a routing policy for our system is that the overlay members do not have prior knowledge of what server will be contacted next. Although the network may have a few servers that it contacts often, there will be situations when a new and unexpected server will be contacted. We will be focusing on the scenario with no prior knowledge.

The problem faced by the policy is that an entry point has no historical information about a server if it has never been contacted before. The first time it learns of the server is through a client's request. Since the client is waiting for the connection to complete, the entrance point cannot spend too much time collecting information. Furthermore, NATRON connections are likely to be short HTTP transfers, making a lengthy network measurement phase unattractive.

To evaluate a NATRON policy, we compare it to an ideal policy which always

chooses the correct intermediate node. The next chapter explains how we simulated an ideal policy and summarizes how well it performs.

Chapter 5

Alternate Path Performance

The goal of this section is to find an upper bound on the performance gains available to NATRON and to characterize the relationship between round trip time and download time when using intermediate nodes.

To find the optimal potential gain we performed exhaustive downloads using every node in our twelve node test-bed to download documents from over 600 unique webservers using each of the alternate paths. We collected these measurements from each of the nodes, and inferred from the data what performance benefits an ideal routing policy might achieve.

Round trip times were collected while performing the downloads, and were used to find the relations between round trip time and download time.

5.1 Experimental Setup

We used 12 nodes from the RON test-bed [3] as our experimental network. Five of the nodes were located at academic institutions in the United States, six were scattered around the US on commercial networks, and one was located in Korea. The node locations and connection information are listed in Table 5.1.

Every node in our test network downloaded via every other node, except in the case of Internet2 nodes. Internet2 nodes were not permitted to use each other as intermediate hosts because our test network has a disproportionately large number

Name	Location	Access Link
mit	Cambridge, MA	Ethernet
nyu	New York, NY	Ethernet
cmu	Pittsburgh, PA	Ethernet
utah	Salt Lake City, UT	Ethernet
cornell	Ithaca, NY	Ethernet
ccicom	Salt Lake City, UT	Ethernet
mediaone-ma	Cambridge, MA	Cable Modem
aros	Salt Lake City, UT	Fractional T3
mazu1	Cambridge, MA	T1
pdi	Silicon Valley, CA	Ethernet
msanders	Foster City, CA	T1
kr	Korea	Ethernet

Table 5.1: Names, locations and connection characteristics of all nodes in the test network.

of nodes in academic institutions. Using the high performance Internet2 network would inflate the potential benefits of this system, and would not reveal the true performance characteristics of the commercial Internet.

To evaluate the real and potential performance benefits of NATRON, we fetched documents from a large set of Internet web servers. To create a large set of web servers, we started with a set of URLs collected on January 21, 2002 by an IRCache[1] [11] web proxy in Boulder, CO. To avoid over-representing popular servers, we limited the set of URLs to contain only one URL per server.

The URL domain names were replaced with static IP addresses to ensure that a single server was used for all fetches of each URL. This prevents different fetches for the same URL from using servers at different IP addresses.

Each URL in the server set was downloaded once to check if the server was active and responding. Unresponsive servers were removed from the set. The remaining URLs were used in the measurements.

To measure the performance of a path from a test client to a webserver, a client simply downloaded a document from that webserver using the path in question. In order to compare the available paths from a client to a web server, the client did

[1]These traces were made available through funding provided by National Science Foundation grants NCR-9616602 and NCR-9521745, and the Nation Laboratory for Applied Network Research.

consecutive downloads of the same document while using the direct path or a different neighbor each time. We call this a download set. Within a download set, the order in which paths were tried was chosen randomly because a web server may react differently to several identical requests depending on the order they get serviced; randomizing the order should reduce such a bias.

The downloads in each set were started at 500ms intervals regardless of how long each download took. Forcing the downloads to begin at regular intervals ensures that all downloads begin within a specified time-span and experience comparable network conditions.

Successive sets were started at 30 second intervals to avoid overloading the overlay network. Five sets were taken for a URL and then the client moved on to the next URL. All of the clients ran this concurrently, but they started 15 minutes apart so that they would not flood the web servers or the links with many concurrent requests.

If any server returned a *Connection Refused* error, that set was rejected because the response time would not signify anything about connection quality.

Download times were capped at 60 seconds. Any download that had not finished in 60 seconds was assigned a time of 60 seconds. Each set was also checked to make sure all the downloaded documents were the same size. If the size of the HTTP response was not equal, the download times could not be compared, so the set was rejected. The exceptions to this rule were requests that timed out and returned zero bytes; they were included in the evaluation.

Each client collected from 3,074 to 3,185 download sets, consisting of 625 to 662 unique URLs. In aggregate, the clients collected a total of 37,712 sets taken over an 80 hour period from Jun 19 to Jun 23, 2002.

5.2 Ideal Performance

In examining each set, we compared the download times recorded for each of the paths used, and chose the smallest as the ideal download time. The intuition is that the ideal routing policy would choose the path with the shortest download time.

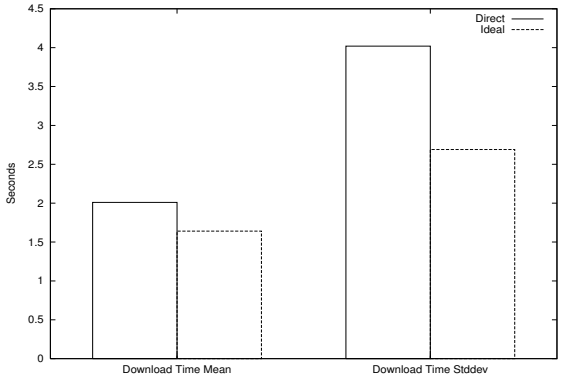

Figure 5-1: Mean and standard deviation of download times in seconds, using the direct Internet path and the fastest of the available paths. Using the fastest of the available paths would reduce the average download time and standard deviation considerably.

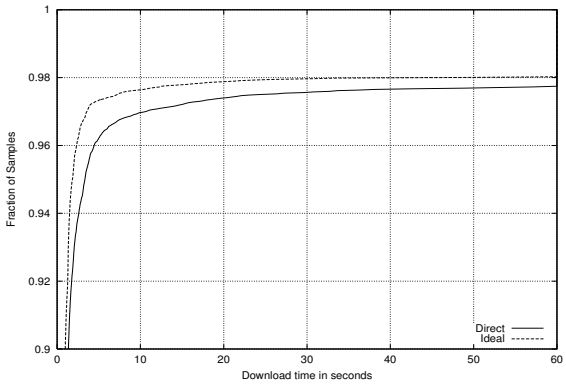

Figure 5-2: Upper 10% of CDF comparing download times using the direct Internet path and path route for 37,712 samples over all clients. Overall, the ideal policy performs well at reducing the number of slow download times and reducing the size of the distribution's tail.

Table 5-1 shows the mean and standard deviations of the direct and ideal download times of all sets. Using the ideal policy reduced the average download time by 18% and the standard deviation by 33%. The drops in download time mean and standard deviation suggests that overlay routing can significantly improve download times.

5.2.1 Distribution of Download Times

Figure 5-2 shows the upper 10% of the CDFs of the direct and ideal download times. 8% of the sets took longer than two seconds using the direct path, but only 5% needed that much time when using the best alternate path. Figure 5-2 also shows that 2.4%

of the samples took longer than 30 seconds or did not complete at all using the direct path. Using the best available path, that fraction was reduced to 2%[2]. In the lower portion of the CDFs which is not shown, the ideal plot is consistently above the direct plot until the smallest 2% where the CDFs become indiscernible.

The reduction in slow samples is promising from a performance perspective, but this figure hides details regarding the individual download sets and does not account for the different document sizes in each download set. The format of the plots does not show pairwise comparisons for each download set, and since the document sizes for each of the download sets may be different, the time values for each set are not directly comparable.

5.2.2 Distribution of Download Time Ratios

To compare the direct and ideal download times within individual sets, we evaluate the ratio of the direct download time to the ideal download time. The ratio is appropriate to compare downloads from the same set because each document in a download set is the same size. Documents from different sets are different sizes, so the ratio normalizes across sets with different document sizes.

The ratio of the direct to ideal download times is plotted as the dashed CDF in Figure 5-3. A ratio greater than one means that the direct path took longer than the ideal path, therefore points on the right of graph represent samples that would have gained by using an alternate path.

One of the complications that arise when comparing pairs of download times is that pairs of downloads using the same path, spaced seconds apart from one another, typically exhibit differences in completion time. The variation may be mistaken for gains and losses when a pair of download times is compared even though the differences are due to variation.

Figure 5-3 contains a plot labeled Direct/Direct showing the CDF of the ratio

[2]Unresponsive servers cause the two percent of failed downloads. We confirmed that all servers in our URL set were responsive before we began our experiments, but some servers may have become unavailable prior to our download measurements.

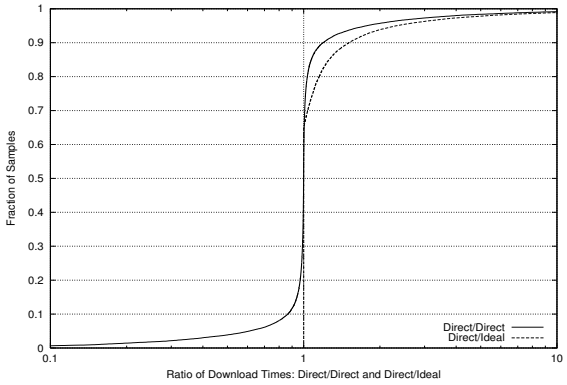

Figure 5-3: CDF of the ratio of two direct route download times and the ratio of the direct route time divided by the ideal route time. The upper 30% of Direct/Ideal reveals that 30% of the download sets could have reduced their download time considerably using an alternate route.

between pairs of downloads in the same set which both used the direct Internet path. Points where the ratio is near one represent download pairs that had very similar download times. Points to the left and right signify pairs where one download completed faster than the other. The Direct/Direct distribution represents the typical ratios one would observe even without using any alternate path routing.

For each set, one of the direct download times was collected by explicitly using the direct path. The second download time used to calculate the ratio was measured in the same download set by an experimental policy which chose the direct path to perform a download. In 7.7% of the download sets, the experimental policy did not choose the direct path; these sets are not included in the reference plot. From previous experiments, we do not expect the removed sets from altering the plot significantly. However, in the worst case, these removed sets could either compress or expand the the reference plot by 3.4% at the top and bottom; they would not change the overall shape or symmetry of the plot.

The significance of the lower portion of the Direct/Ideal plot is that 70% of the sets showed no benefit from using an alternate path route. In contrast, the upper portion of the plot reveals that in 30% of the download sets, a download using an alternate path completed the download in less time than the download which used the direct path.

Figure 5-3 also shows that the direct path download took at least 10% longer than the ideal path in 24% of sets. In comparison, only 13% of the Direct/Direct ratios used at least 10% more time than the previous direct download. In 6% of the sets, the direct path download took more than twice as long than the ideal path download. On the reference plot, only 4% of the Direct/Direct sets have a ratio of 2 or greater.

The conclusions from Figure 5-3 are that 30% of the download sets could have benefited by alternate path routing and that those benefits can be distinguished from the natural variation in the direct path download times.

5.3 Policy Considerations

Data presented in the previous section suggests that using intermediate Internet hops has the potential to improve download times for cooperating clients. This section describes a naive policy for choosing intermediate nodes called SYN-Probe and two reasons why SYN-Probe is impractical for general use.

Previous work in server selection techniques [7, 6] suggests that probing the available servers and choosing the server with the shortest round trip time is a good method for deciding which server to download from. It seems reasonable for NATRON to use a similar approach in which it picks the intermediate node that gives the lowest round trip time to the server. SYN-Probe, a basic version of this idea, chooses an intermediate node for each new TCP connection. SYN-Probe sends concurrent SYN (TCP connection setup) packets via all available intermediate nodes. It selects the intermediate node through which it first hears a reply from the server.

There are two problems with SYN-Probe. Round trip time is not always a good predictor for download time, and it is impractical to send a duplicate SYN for via all of the nodes in the overlay if the overlay has more than a few members.

5.3.1 Round Trip Time Prediction

The first problem with SYN-Probe is that round trip time is not always strongly correlated with download time. Figure 5-4 presents the relationship between round

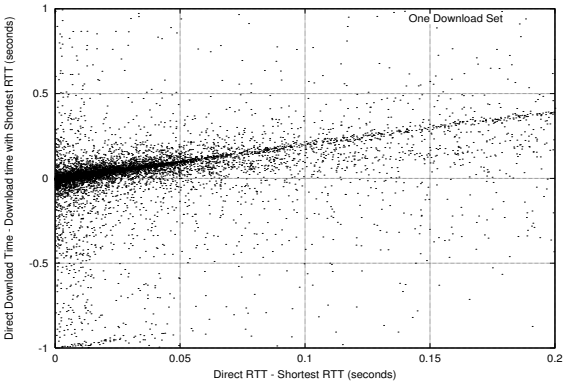

Figure 5-4: Difference in Round Trip Time vs. Difference in Download Time. Each point represents a download set. If an alternate path round trip time is shorter than the direct round trip time by 20ms or more, the alternate path download time is likely to be shorter as well. If the difference in round trip time is less than 20ms, then the download time is difficult to predict.

trip time differences and their corresponding download time differences from the above dataset. Points above the $y = 0$ line represent sets where the direct download time was greater than the download performed on the path with the quickest round trip time. Points below the $y = 0$ line represent sets where the opposite was true: the direct path download completed faster than the path with the quickest round trip time.

The trend of interest in Figure 5-4 is the line with slope two formed by the dense points. This line begins to drop below the $y = 0$ line when the difference between the shortest round trip and the direct round trip drops below 20ms. This drop suggests that an alternate path round trip time should be at least 20ms lower than the direct round trip time to justify choosing the alternate path over the direct path.

5.3.2 SYN Flood

An additional problem with the SYN-Probe policy is that it probes all of the available paths. A practical routing policy cannot probe every path because the server may experience SYN flooding when many concurrent SYNs are sent at once. Ideally, a policy should limit the rate at which it sends probes and/or limit the number of neighbors that it probes.

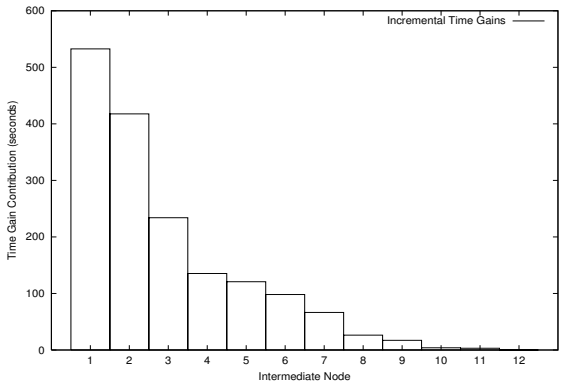

Figure 5-5: Incremental time gains for adding intermediate nodes to client Aros. The best intermediate nodes 1, 2 and 3 account for 71% of the gains made possible by alternate paths.

To consider the practicality of limiting the number of SYN probes, we examined the clients and calculated how much each additional intermediate node would add to the total time savings of each particular client. Figure 5-5 shows the incremental time savings each intermediate node would provide for one of our test clients in Utah named Aros. The left most bar shows the potential time savings provided by Aros' best performing intermediate node. Each additional bar to the right shows how much additional savings neighbor i would provide if used in conjunction with neighbors $1...i-1$. For Aros, the three best intermediate nodes contributed 71% of the potential time gains. Further analysis of Aros' download sets showed that the direct path would have been ideal in 55% of the download sets, meaning that Aros could have covered 88% of the ideal paths with only the direct path and its best three neighbors. This analysis only considers the Aros client, but the results are similar for the remaining clients.

The high contribution of so few intermediate nodes suggests that a client should not need to probe all its neighbors when deciding on a intermediate node. Better performing neighbors should be given more priority when probing alternate paths. A client's best three neighbors can be found by running the experiment in 5.1 and choosing its three highest contributing neighbors.

Chapter 6

Policy and Performance

This section describes the policy NATRON uses to choose intermediate nodes and how well NATRON's policy improves routing performance compared to direct Internet routing and an ideal policy. We observe that the policy does reap some benefit, but the gains do not approach those achieved by the ideal policy.

6.1 The NATRON Policy

The NATRON policy is based on the SYN-Probe policy described earlier, but it addresses the two problems revealed in Section 5.3. It shares the idea of probing the available paths and picking the path with the shortest round trip time, but it improves on SYN-Probe using the data presented above.

6.1.1 Reducing SYN Probes

The NATRON policy uses information about its best neighbors to reduce the number of SYNs it sends out concurrently. NATRON only sends out three concurrent SYNs at a time, and starts with its three best performing neighbors(defined in Section 5.3.2) rather than sending a SYN via every neighbor.

One concern in probing three paths at first is that all three paths might be unavailable or under-performing due to network outages or congestion. To address this

risk, NATRON cycles through all of the available paths if no acknowledgments are received.

Another concern in reducing the number of probes is that the fastest path to a destination is not via one of the best three intermediate nodes. We feel that the risk is low from our analysis in 5.3.2 and that the impact of excessive probing would make NATRON impractical. Reducing the number of SYNs is necessary to make NATRON a viable system.

6.1.2 Delaying SYN Probes

The problem with round trip time prediction is addressed by giving preference to the direct path when choosing intermediate nodes. An intermediate node is only used if the round trip time using the intermediate node is much shorter than that of the direct path. This preference is imposed by giving the SYN on the direct path a head start. When a client initializes a TCP session, a SYN is sent along the direct path immediately, but probes are delayed by some time t milliseconds before they are sent out. The delay time t is chosen as the x intercept in Figure 5-4, marking the minimum improvement in round trip time that produced an improvement in download time. The analysis in 5.3.1 led us to use 20ms as our delay time t.

6.1.3 Algorithm

The policy is invoked when a client initiates a new TCP session by sending the first TCP SYN packet. The SYN packet is immediately forwarded along the direct Internet path to the destination host and a timeout of t milliseconds is scheduled. If at any time, a SYN-ACK returns to the client, the path it traveled on is chosen to carry out the session and all other initialized sessions are reset.

After t milliseconds have passed, three SYN probes are then sent out via the best three neighbors. The best three neighbors are calculated offline using the technique described in 5.3.2. If no acknowledgment is received and the client retransmits the SYN, the retransmitted SYN is immediately forwarded along the direct path, and

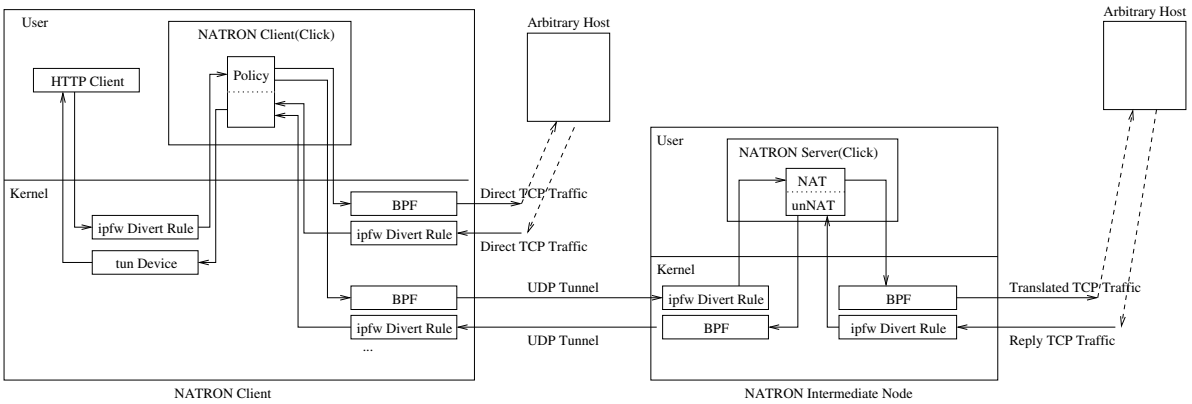

Figure 6-1: Illustration of NATRON's software implementation.

another timeout of t milliseconds is scheduled. If no acknowledgments are received when the second timeout triggers, NATRON sends three more probes via an arbitrary three of the remaining nodes. Retransmitted SYNs are handled in this way until a SYN-ACK is received or the client stops retransmitting SYNs.

6.2 Implementation

The NATRON implementation is designed to facilitate policy design. It is implemented as a custom router configuration for the Click Modular Router [14]. Figure 6-1 illustrates the traffic flow in our NATRON implementation. Both the client and the intermediate node runs as a userlevel Click processes on FreeBSD [9] systems.

The architecture is designed to simplify policy development. Implementing most alternate policies only requires changes to a Click policy element which is written in C++. The element is given control over all packet interaction between the client and the destination host and is capable of using any of the available neighbors. All forward and reverse packets can be observed by the module and the module can utilize all intermediate hosts for packet indirection. The architecture is flexible and simplifies policy implementation because the forwarding and NAT machinery is reusable.

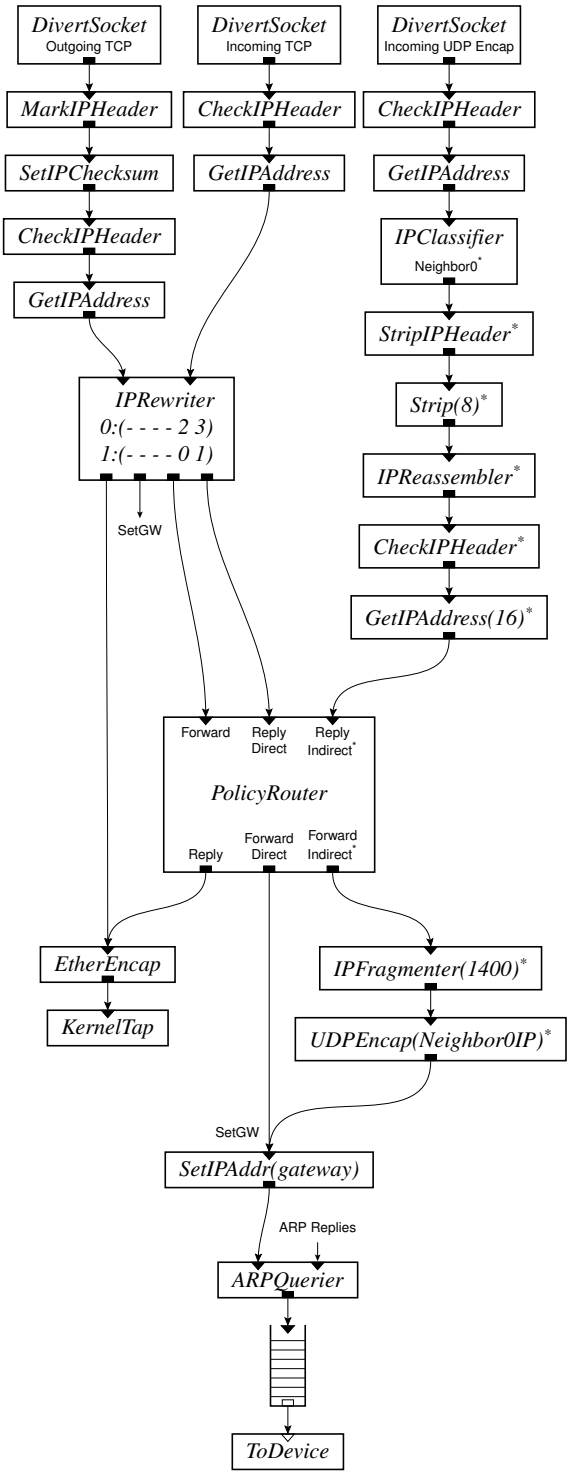

Figure 6-2: Sample NATRON client node Click configuration. This configuration supports a single neighbor, *Neighbor0*. To extend this configuration to support multiple neighbors, duplicate the starred components.

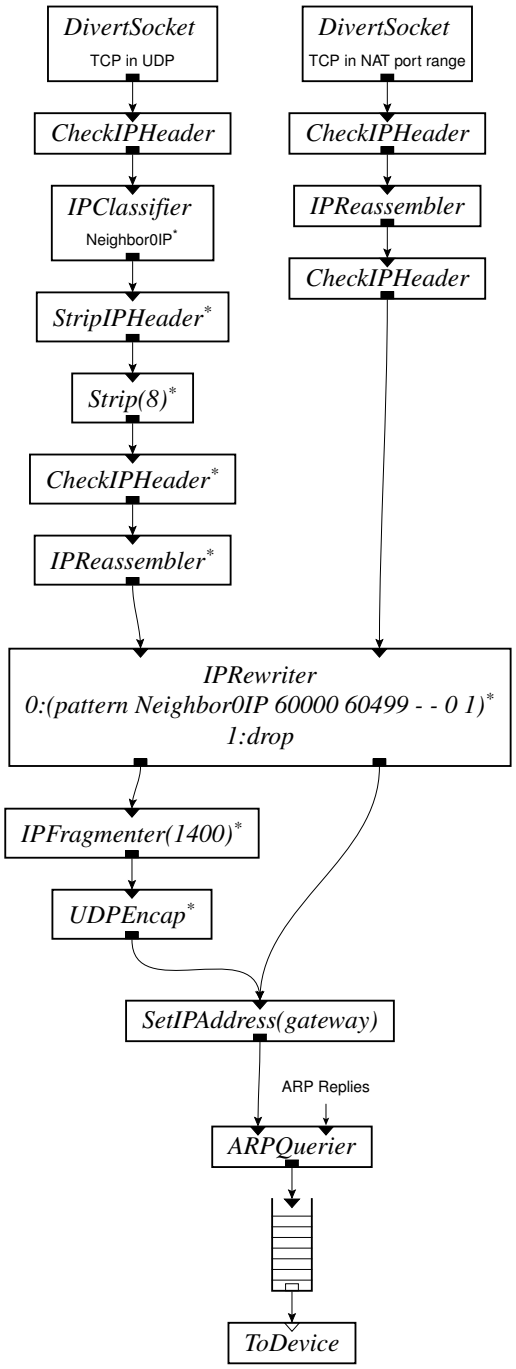

Figure 6-3: Sample NATRON intermediate node Click configuration. This configuration supports a single neihbor, *Neighbor0*. To extend this configuration to support multiple neighbors, duplicate the starred components.

6.2.1 Client Node

On the client node, a firewall divert rule diverts client TCP traffic into the Click where packet processing takes place. Figure 6-2 illustrates the Click configuration on the client node. Outgoing packets are sanity checked and then classified by an *IPRewriter*[12]. The *IPRewriter* patterns ensure that flows originating from the client are processed by NATRON and that packets originating from other hosts bypass the NATRON elements. All NATRON traffic passes through the *PolicyRouter* element. The *PolicyRouter* element implements the path choosing policy by sending SYN packets according to the NATRON policy and by observing returning SYN-ACKs. The *PolicyRouter* selects the intermediate node for each flow and either sends session packets directly to the destination through a Berkley Packet Filter device, or tunnels them to a NATRON intermediate node via UDP.

Reply packets come in two forms. For flows using the direct path, TCP packets return directly to the client. These return packets are taken up by the *Incoming TCP DivertSocket*, and enter the *PolicyRouter*. The *PolicyRouter* handles SYN-ACKs as described above; other packets are passed to the operating system through to the *KernelTap* element.

For flows using an intermediate node, TCP packets return encapsulated in UDP. The UDP packets are diverted by a firewall rule and enter Click at the *Incoming UDP Encap DivertSocket*. Click strips the IP and UDP headers off the packets and the *PolicyRouter* processes them as described in the previous paragraph.

6.2.2 Intermediate Node

The NATRON intermediate node (see Figure 6-3) has a simpler task. It receives UDP encapsulated packets, removes the UDP/IP header and performs NAT in the *IPRewriter*. Translated packets are then emitted on the local network. Reply TCP packets which match the port range used by *IPRewriter* are taken by Click via a divert socket and reverse translated by the *IPRewriter*. They are then encapsulated in UDP and sent to the originating NATRON client node.

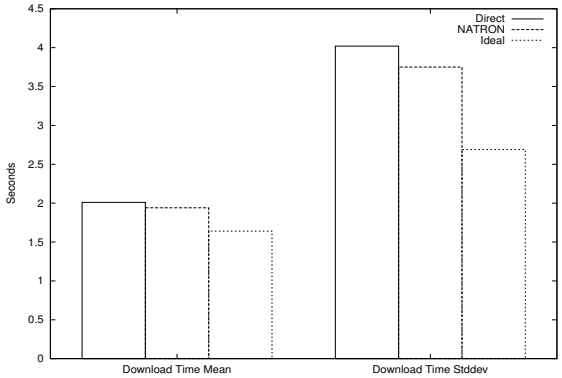

Figure 6-4: Mean and standard deviation of download time in seconds, using the direct Internet path, NATRON chosen path, and ideal alternate paths. NATRON improvements are minor compared to potential performance improvements.

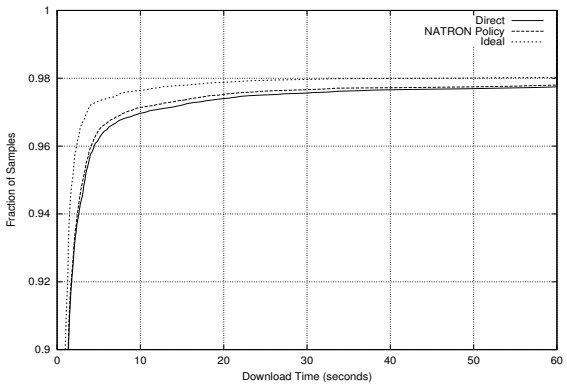

Figure 6-5: CDF comparing download times using direct Internet route, NATRON chosen route and ideal route.

6.3 Policy Performance

To evaluate the NATRON policy, we deployed it on our test network. While collecting the data presented in Section 5.1, we also collected download times using the NATRON policy router.

The mean and standard deviation for NATRON downloads is presented in Figure 6-4. The results show that the NATRON policy was able to take advantage of some performance benefits, but not as much as an ideal policy.

Figure 6-5 presents the CDF of the download times using the NATRON policy against the CDF for the direct and ideal download times. It is clear from Figure 6-5 that the NATRON policy does not exploit the potential gains from the underlying overlay network. It does, however, remove a portion of the long and failed downloads.

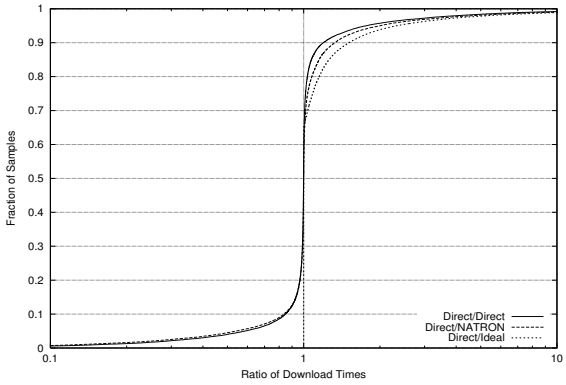

Figure 6-6: CDF of the ratio of NATRON download time to Direct download time. The NATRON policy improves download performance in the upper 25% of download sets.

2.4% of the downloads took longer than 30 seconds to complete using the direct path. NATRON reduced this fraction to 2.3% which accounts for 25% of the potential decrease.

Figure 6-6 shows the ratio of the direct path download time to the policy download time against the ideal policy, and the reference plot. Points to the left of one signify sets where the policy took more time than the direct path; points to the right show where the policy downloaded the document faster than the direct path. On the left hand side, the NATRON plot closely matches the Direct/Direct plot, meaning that the NATRON did not show more variation than the Internet at large. In the upper 25% of the graph, the NATRON plot separates from the Direct/Direct plot and remains between the Direct/Direct plot and the Direct/Ideal plot. The upper 25% signify that the NATRON policy improved download ratios in roughly 25% of the download sets, but did not achieve large improvements available to the ideal policy.

Our measurements conclude that NATRON's current policy for choosing intermediate nodes only achieves a small portion of the potential gains, but it was not successful at exploiting the full potential of overlay networking. We believe that with further work, a good policy can be developed to exploit the performance of overlay routing for oblivious hosts.

Chapter 7

Related Work

Labovitz et al. confirmed that BGP does not react to changing network conditions as quickly as desired. Routing instability and oscillations can occur for minutes at a time in response to network changes [13]. As a result, link failures, link congestion and routing anomalies can cause poor performance and low reliability in connections when using standard Internet routing techniques [15]. NATRON's goal is to reduce the symptoms of routing instability by utilizing alternate network paths.

In [18] and [19], Savage et al. examine link quality between sets of 15-39 public traceroute servers. They analyze the latency and loss rates recorded by the traceroutes and infer that 30-80% of direct paths had significantly better alternate paths. Our measurements are unique because they consider network performance from 12 overlay nodes to more than 600 unique Internet hosts. Furthermore, we have designed a system to take advantage of the opportunity Savage et al. describe.

Collins describes how to build a system implementing overlay routing in [5], and analyzes the packet processing overhead incurred by alternate path routing. He goes on to explain why network address translation should be used where traffic leaves the overlay and suggests a routing policy based on comparing round trip times. Our work furthers his. We have designed and implemented a routing policy using measurements taken on an Internet test-bed, and then evaluated its ability to choose alternate paths in a deployed system.

Andersen et al. [2, 1, 3] built RON, which exploits routing inefficiency. RON

is limited to improving communication within a small group of cooperating hosts. NATRON allows a similar group of cooperating hosts to exploit the same routing inefficiency, but is not limited to communicating between participants. NATRON provides a way for participants to use intermediate Internet hops when communicating with oblivious destination hosts.

Steinbach [20] chooses between multiple paths by sending probes along each path, but requires cooperation from the destination host. NATRON differs because it only needs cooperation from participating hosts. NATRON does not need cooperation from destination hosts to choose intermediate nodes.

Earlier studies of Internet server selection [7, 6] algorithms show that choices based on shorter round trip time perform well for choosing among replicated servers. NATRON differs because it does not select between replicated servers. NATRON selects between intermediate nodes which control the path a packet travels on to the destination host.

InterNAP [10] connects directly to backbone providers at sites called P-NAPs. At P-NAPs, InterNAP routes packets directly to the backbone which services the packet's destination host. NATRON differs because it does not necessarily have a presence in all ISPs and it does not depend on a single sites with connections to many backbone providers. NATRON also does not need to know which backbone services a particular IP address to choose intermediate nodes.

X-Bone [21] is a general architecture for deploying and managing overlay networks. NATRON could use X-Bone as an underlying substrate.

Chapter 8

Conclusion

We have described NATRON, a system where an overlay network of hosts can co-operate to improve routing to arbitrary Internet destinations. The system differs from previous overlay networks because it allows members to forward traffic through intermediate nodes on its way to hosts outside the overlay network.

We estimated the potential performance gains a system like NATRON might achieve. We used a twelve-site geographically diverse test-bed to download documents from over 600 Internet web servers and compared download times using different nodes as intermediate hops. In our dataset, we found that overlay routing could have decreased the average transfer time by 18% and reduced the number of downloads taking longer than 30 seconds by 16%.

We built a working NATRON and designed a path choosing policy. We evaluated our policy by using it on the same twelve-node test-bed and compared its results to the potential performance gains. We found that the NATRON policy achieves 22-25% of the potential, but does not exploit the maximum benefits of the alternate paths. We believe that with future work, a policy could be designed to exploit more of the potential gains revealed by our performance measurements.

Bibliography

[1] D. Andersen. Resilient overlay networks. Master's thesis, Massachusetts Institute of Technology, 2001.

[2] D. Andersen, H. Balakrishnan, M. Kaashoek, and R. Morris. The case for resilient overlay networks. In *Proceedings of the 8th Annual Workshop on Hot Topics in Operating Systems (HotOSVIII)*, 2001.

[3] David Andersen, Hari Balakrishnan, Frans Kaashoek, and Robert Morris. Resilient overlay networks. In *Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP)*, October 2001.

[4] R. Chandra C. Villamizar and R. Govindan. RFC 2439: Bgp route flap damping, November 1998.

[5] A. Collins. The detour framework for packet rerouting. Master's thesis, University of Washington, 1998.

[6] Mark Crovella and Robert Carter. Dynamic server selection in the Internet. In *Proceedings of the Third IEEE Workshop on the Architecture and Implementation of High Performance Communication Subsystems (HPCS '95)*, 1995.

[7] Sandra G. Dykes, Kay A. Robbins, and Clinton L. Jeffery. An empirical evaluation of client-side server selection algorithms. In *INFOCOM (3)*, pages 1361–1370, 2000.

[8] K. Egevang and P. Francis. RFC 1631: The ip network address translator (nat), May 1994.

[9] FreeBSD. http://www.freebsd.org.

[10] InterNAP. http://www.internap.com.

[11] IRCache. http://www.ircache.net.

[12] Eddie Kohler, Robert Morris, and Massimiliano Poletto. Modular components for network address translation. Technical report, MIT LCS Click Project, December 2000. http://www.pdos.lcs.mit.edu/papers/click-rewriter/.

[13] Craig Labovitz, Abha Ahuja, Abhijit Bose, and Farnam Jahanian. Delayed internet routing convergence. In *SIGCOMM*, pages 175–187, 2000.

[14] Robert Morris, Eddie Kohler, John Jannotti, and M. Frans Kaashoek. The click modular router. In *Symposium on Operating Systems Principles*, pages 217–231, 1999.

[15] Vern Paxson. End-to-end routing behavior in the Internet. *IEEE/ACM Transactions on Networking*, 5(5):601–615, 1997.

[16] Y. Rekhter and P. Gross. RFC 1772: Application of the Border Gateway Protocol in the Internet, March 1995.

[17] Y. Rekhter and T. Li. RFC 1771: A Border Gateway Protocol 4 (BGP-4), March 1995.

[18] S. Savage, 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(1):50–59, 1998.

[19] Stefan Savage, Andy Collins, Eric Hoffman, John Snell, and Thomas E. Anderson. The end-to-end effects of internet path selection. In *SIGCOMM*, pages 289–299, 1999.

[20] Eckehard G. Steinbach, Yi J. Liang, and Bernd Girod. Packet path diversity for tcp file transfer and media transport on the internet. http://wsl.stanford.edu/projects/Retreat/steinbach.pdf.

[21] J. Touch and S. Hotz. The x-bone. In *Globecom*, pages 75–83, 1998.