MIT 18.996: Topic in TCS: Internet Research Problems **Spring 2002**

Lecture 9 — April 3, 2002

Lecturer: Gary Miller *Scribe: Noshirwan Petigara and Rui Fan nosh,rfan@mit.edu*

The lecture introduces the concept of overlay networks. Overlay networks are application level routing mechanism that allows hosts to take advantage of routes that are not advertised by traditional internet routing protocols. By taking advantage of these paths, applications can improve their end-to-end reliability and performance. The concepts are explained by discussing the motivation, design, implementation, and performace of the akarouting protocol, developed at Akamai.

9.1 Why Traditional Routing is not always the best method

The internet is comprised of multiple autonomous systems (AS). Each AS is generally controlled by an independent entity, most often commerical. The connectivity between AS's is generally based upon two methods: transit and peering. If ISP's A and B have a transit relationship, then packets destined for B can be routed through A and vice versa. These routes are publicly advertised and these globally advertised transit relationships usually include some finacial compensation. If A and B have peering relationships, then customers of A and B can reach each other though these links. However, these links are not publicly advertised - they stay a secret between A and B, and so other AS's cannot route packets to A through B and to B though A. As only customers of A and B can send packets through

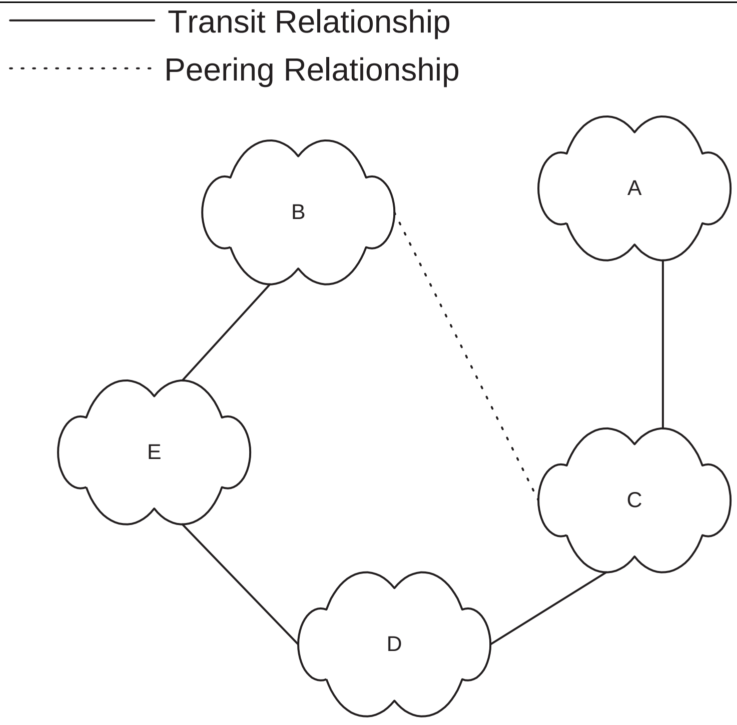

Figure 9.1. Effect of Transit and Peering Relationship on How Packets are Routed Through ASes

the peering link, these relationships are often free.

Transit routes are advertised by the Border Gateway Protocol (BGP), which allows ISPs to exhange route information. Consequently, an ISP can find the next hop for a packet that is destined for an ISP that is does not have peering or transit relationships with. However, BGP cannot guarantee the optimal route. As seen in figure 9.1, a packet that needs to go from A to B will have to traverse through C, D, and E if it follows the BGP advertised route. However, if the packet was first sent from A to C, and then it could be sent from C to B by taking advantage of the peering relationship between C and B. This gives an indication that using application level intermediate nodes could provide a transmission with a more efficient path.

However, the motivation behind overlay networks is not simply circumventing the financial relationships of ISPs. The BGP protocol was built from the ground up to be highly scalable. However, to achieve this scalability the BGP protocol is very simple. It uses simple hop-counting metrics for picking "best" routes, which are often non optimal. These are documented in the Detour Study [2]. Additionally, the BGP routing protocol does not probe paths for performance so it is unable to route around congested links. Recent studies by Chandra et al [1] and Paxson [4] have documented these characteristics. More disturbingly, Labovitz discovered that due to implementation issues and ambiguity with the protocol, the convergence time for BGP routing tables to stabilize after a link or other failure was on the order of many minutes [3]. In addition to the problems with BGP, Paxson documented the rising incidence of major routing pathologies and path outages [4]. Consequently, end-to-end routing mechanisms neither guarantee the best path between end hosts, nor do they recover quickly from link failures.

9.2 Akarouting

9.2.1 Vision

An examination of current internet routing mechanism and the analysis of the above studies leads one to believe that one might be able to provide higher performance and better failover properites if applications were able to take advantage of multiple paths through different AS's. This could be achieved by placing nodes within multiple AS's, creating an overlay network. These nodes would probe links to the other nodes and export the paths that have the best performance. Consequently, if a path is found that is better than the BGP chosen path, packets can be routed by this path using the cooperation of intermediate nodes in the overlay network.

9.2.2 Possible Applications for Akarouting

- **Streaming Network:** Reduce lost packets by using overlay network.
- **Voice over IP:** Reduce jitter by using overlay network.
- **VPN:** Route around link failures by using overlay network.
- **Browser Plug in:** Improve overall browsing experience.

9.2.3 Experiments

In order to quantify the effect of tunelling packets through an intermediate AS, experiments had to be run. The purpose of these experiments was to figure out the following questions:

1. How often does tunneling via a third host give better performance than a direct route?
2. How do we predict a good path? Is it enough to do a ICMP ping? If a ping does not correspond to a good transfer, then is it enough to make a TCP connection to and from the end hosts and record the time taken. Or is it necessary to actually perform a small transfer between the hosts.
3. If a path is good at time t, then is that path always the best path, or does the best path change over time.

In order to answer these questions, two sets of experiments were performed. The first set of experiments were performed using 40 hosts scattered around the world. Ping times, TCP pings (connection time) and download times were recorded between 30,000 pairs of centers. These experiments showed that an average of 15 to 30% gain was possible by choosing between multiple paths. Also, these experiments showed that the best path changed often. Howver, these experiments showed that ping time was not a good predicter for download times.

In the next set of experments, 25 pairs of centers were used. Every five minutes, three simulataneous small downloads were performed over 3 different paths. These experiments confirmed the results from the previous experiment. However, this experiment also proved that if simulataneous small downloads were performed and the best path chosen, then that path could be used to get the best download time for a file for a short period of time.

Based on the results of these experiments, the Akarouting system was constructed.

9.3 Akarouting Implementation

Akarouting consists of two components, MapMaker and Guide. MapMaker produces a global measure of “distance” on the Internet, and suggests paths for traffic. Guide provides a view from the edge of the Internet, and selects the fastest path from those suggested by MapMaker.

9.3.1 MapMaker

MapMaker produces a map of the best one-hop and two-hop paths between each Akamai data center (DC) and a content provider’s (CP) sites. There are three steps to generating this map. First, MapMaker collects ping data from around the Internet. Then, it computes the effective distance between sites based on the ping data. Finally, it uses a special metric to compute the shortest paths between sites. We will describe these steps in more detail.

Collecting Ping Data

MapMaker uses two sources for estimating the ping between Akamai DC's and CP sites: Akanote and Trace Ping Server (TPS). Akanote selects 35 Akamai DC's, and pings all other Akamai DC's from those DC's. TPS uses 90 Akamai DC's to ping other Akamai DC's, and also pings 20 CP data centers housing meta-data configurable content. The pings are conducted every 15 minutes. The resulting data can be tailored to specific content providers via *strategies*. For example, different strategies may be applied for Yahoo's images versus its texts.

Computing Effective Distance

The ping data MapMaker collects is converted to effective distances between sites using a "Magic Formula." The formula takes into account ping time, ping loss, and the age of ping data. For example, when a site is suffering 100% ping loss, then either the target machine at the pinged site is down, but otherwise the site is okay, or the Internet connection to the site is actually down. In the first case, MapMaker simply discards the ping data to that site. In the second case, 100% loss will be charged to that site, which dramatically increases its effective distance. MapMaker uses the rule that a machine is up at some T if someone has received a packet after time T from the machine.

Computing Shortest Paths

MapMaker produces either a shortest one or two-hop path between Akamai DC's and CP sites. The one-hop path goes directly from an Akamai DC to a CP. The two-hop path goes through one additional Akamai DC called the *middle* DC. The usable middle DC's are set on a per strategy basis. They come either from a standard list of middle DC's, or an explicit list. Using ghost info data, suspended DC's can be eliminated from the list. A cut-off can be set based on the DC's load, though this feature is currently not used. MapMaker tries to select diverse paths, going through different ISP's. This way, a problem with one ISP is not likely to degrade all traffic.

MapMaker uses the $L_{1.4}$ norm to compute the distance between sites. Given distance $d(A,B)$ between sites A and B, and distance $d(B,C)$ between sites B and C, the distance between A and C is computed as $d(A,C) = (d(A,B)^{1.4} + d(B,C)^{1.4})^{1/1.4}$. The $L_{1.4}$ norm was selected because it gave the best empirical results.

We still need to assess the quality of a shortest-path map produced this way before using it. The quality of the map depends on the amount of ping data in the map, and the freshenss of the data. Given a map based on N data samples with ages $a_1, \ldots, a_N$, the *quality* of the map is defined as $\Sigma_1^N \frac{1}{a_i}$. If the samples were all collected t time units ago, then the quality of the map becomes $\Sigma_1^N \frac{1}{a_i+t}$. Since the number of samples is usually large, the quality is approximated using an exponential bucketing scheme. Each map is represented by a vector $\{t, b_1, \ldots, b_{50}\}$. t represents the elapsed time since the samples were taken. $b_i, 1 \leq i \leq 50$, represents the number of samples with age between r^{i-1} and r^i, for some r.

The final output of MapMaker are three suggested routes CP, P_0 and P_1. CP is the shortest direct route from an Akamai DC to some CP. P_0 is the best two-hop path, and P_1 is the second best two-hop path. These paths are published via DNS.

9.3.2 Guide

Guide takes the three paths suggested by MapMaker and chooses the actual fastest among them. It does this by *racing* the paths against each other. Downloads are started on all three paths, and the user can start to receive data as soon as the fastest of these three paths finishes.

However, not all content is raceable. For example, if the content is a command to debit a user's account, then only one copy of this content should be sent. When content is not raceable, test downloads are performed on the paths. The first client will use a direct route if no data from the tests is available yet. In general, races are better, as they ensure the first request also achieves better performance.

9.4 Akarouting Results

We now present some results for Akaroute. Three types of downloads were tested on the Yahoo homepage. The images for the tests were cached using FreeFlow. The first test uses only basic FreeFlow. The second test uses Edgesuite (ES). The third test uses ES and Akaroute. Figure 9.2 shows the mean download times for the three tests. The ES/Akaroute test gives the fastest download and has the lowest variance.

Figure 9.3 shows the times for the various stages of downloading Yahoo's homepage. ES/Akaroute gives the fastest download. Furthermore, if we ignore the one time cost of

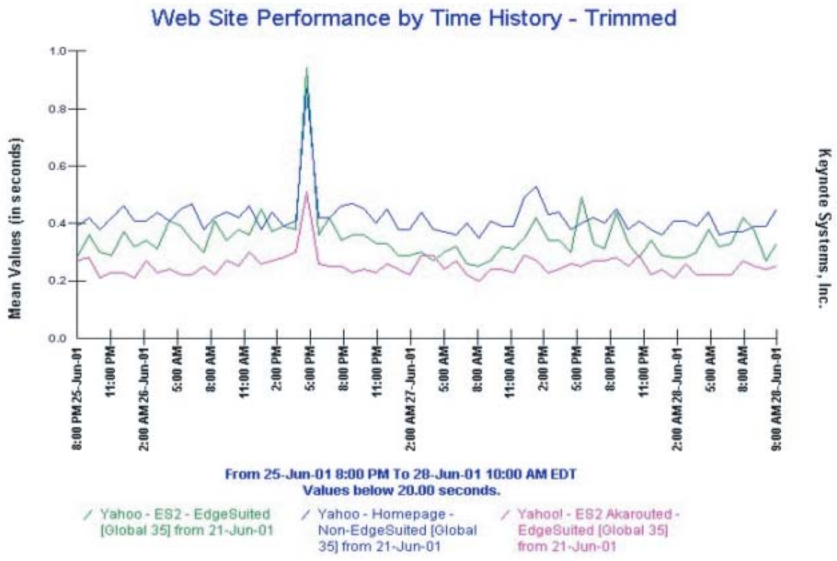

Figure 9.2. Download time time-series.

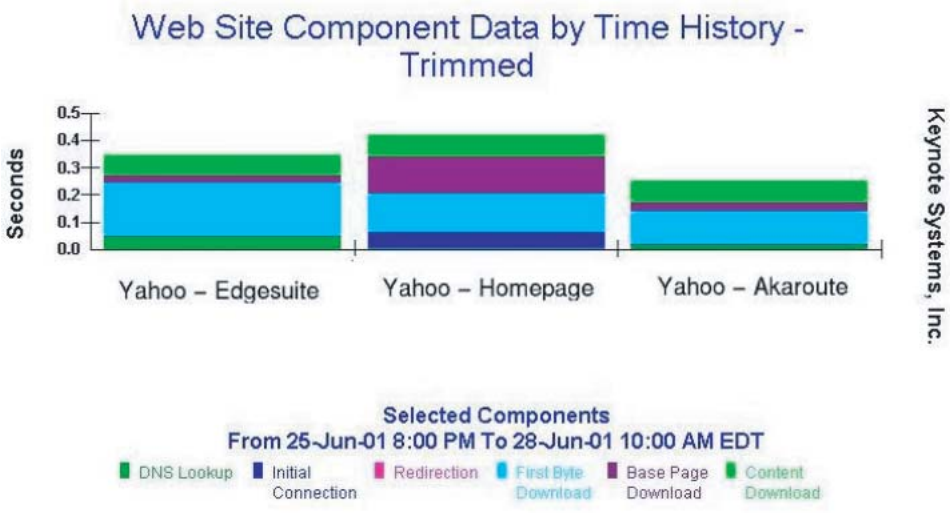

Figure 9.3. Download time history.

DNS lookup, the improvement of ES/Akaroute is even higher. Overall, it was found that EdgeSuite was 27% faster than basic FreeFlow, while Akarouting was 44% faster than basic FreeFlow. The tests also showed that Akarouting is most cost-effective when 25-30% of

Akamai DC's use two-hop routes. Also, Akarouting works well when a small amount of the content is dynamic.

Bibliography

[1] B. CHANDRA, M. DAHLIN, L. GAO, and A. NAYATE. End-to-end wan service availability, 2001.

[2] A. COLLINS. The detour framework for packet rerouting, 1998.

[3] C. Labovitz, A. Ahuja, A. Bose, and F. Jahanian. Delayed internet routing convergence. In *SIGCOMM*, pages 175–187, 2000.

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