The **Stride** Measurement Infrastructure

John Snell, Tom Anderson

May 16, 1998

Abstract

There are a number of measurement points freely provided in the Internet today. Stride takes a dynamicly evolving set of such measurement points, or sources, and performs a series of source to source and source to end-point measurements, at randomized intervals. Our goal in this research is to answer the following question: "given two points $\{A, B\}$ in the network, how often is the default route $A \rightarrow B$ of a higher cost than some alternate multi-hop route $A \rightarrow C \rightarrow ... \rightarrow D \rightarrow B$?"

1 Introduction

1.1 Exploitable Resources

The concept of tracerouting from one system to another has become a standard measure of reachability in today's wide area network. Although every major operating system includes some form of the tool, this only allows one to test the reachability of external machines from a particular host, not the reachability of a particular machine from external hosts. This dichotomy of measurement need has lead to the open provision of the traceroute tool on many external hosts, such that one can initiate measurements from them to a specific machine. Some of these tools are restricted in their selection of a target, tracing only to the requesting host; most allow for unrestricted use. In all known cases, this tool is implmented as a CGI script accessible from a web server.

1.2 Issues of Import

1.2.1 Greed vs. Altruism

Unfortunately, as networking researchers, our intents and those of the providers of the resources we desire to use are somewhat at odds; we wish to take extensive measurements, and they wish to have their server load be non-dominated by what for them is essentially useless computation. Should we take the path of pure greed, our objectives could not be achieved; the resources would disappear due to abuse. Should we ourselves take the path of altruism, only a tiny amount of data is obtained and our conclusions are suspect. Thus we attempt a compromise.

Those who have explicitly stated in advance[1] that they do not wish to have this resource used by non-humans are removed from the study; those who have not, are implicitly believed to be compliant. We define an average time bound on the number of traces each server will be asked to perform per hour; the actual requests may clustered or dispersed, depending on the result of a uniformly distributed random variable.

This balance of use extends from the servers upon whom computation will be performed, to the network at large – our measurements themselves impose load upon the network, and thus need be bounded in their simultaneous number.

1.2.2 Failure Derivatives

The goal of this research is to answer the following question:

[1] This is typically done by the inclusion of a "robots.txt" file in the root path of the web server, delimiting which resources should and should not be used by non-humans.

Given a set of n points in the network, how often does the n-gon inequality fail to hold?

The n-gon inequality is a generalization of the well-known triangle inequality: given the complete graph K_3 of 3 vertices $\{A, B, C\}$, wherein we establish some measure of cost for each edge in the graph, the path $A \rightarrow B$ is less in total cost than the path $A \rightarrow C \rightarrow B$. More generally, given the complete graph K_n of n vertices $\{A_1, A_2, ...A_n\}$, the direct path $A_i \rightarrow A_j$ should be of lesser weight than some alternate path $A_i \rightarrow ... \rightarrow A_k ... \rightarrow A_j$. We define path latency as our cost metric.

Let us assume for a moment that we could ignore both server load and network load. Then the obvious method of measuring the inequality would be, for all n, to test all n-gon inequalities simultaneously. Such a measurement would allow us to determine the instantaneous truth value of the inequality on a case by case basis. But load can't be ignored.

Given a list of t traceroute servers and q additional points in the network, there are a total of $t(t+q-1)$ possible directed edges in the complete interconnection graph.

Thus we put forth a hypothesis to be tested:

The average path-cost for paths of the form $A \rightarrow B$ is calculated from a large number of sparse, randomly sampled measurements. For each pair of nodes $\{A, B\}$, if this sparse aggregate data shows evidence that there exists an alternate path $A \rightarrow C \rightarrow ... \rightarrow D \rightarrow B$ of lower cost, then further analysis of that path structure will, in the common case, only concretize this result.

The measurement process used in this research attempts to validate this hypothesis by performing the large, sparse set of measurements, and subsequently verifying the validity of the inequality failures derived therefrom. This validation is done by simultaneously measuring all links contained in the failing paths, to generate an instantaneous boolean value delimiting the satisfaction of the inequality at the time of measurement. This boolean is then itself the subject of an average to determine how often the failure truly holds.

If validated, the implication of the truth of this hypothesis is as follows: continuation of the less refined measurements will generate no false positives in a search for lower cost non-default routes.

2 Infrastructure Design

2.1 Parameters

In order to quantify various issues that arise in the measurement process, we define the following parameters:

- Let T be the set of traceroute servers available for use at any one point in time. This set may vary in content and size during the experiment due to the attrition rate of disgruntled servers or the addition of new servers by the Harvester. Further, let $t = |T|$
- Let Q be an arbitrary set of non-server nodes in the network, where $Q \cap T = \phi$ Further, let $q = |Q|$.
- Let l be the maximum number of simultaneous traces that we wish to have running over the network as a whole at any instantaneous moment in time. This will often be referred to as the *instantaneous network load.*
- Let m be the maximum time allocated for a traceroute to complete. This (hopefully) bounds the execution time of the trace process on a remote server.
- Let Δ be a uniformly distributed random variable over the interval $[\Delta_{min}, \Delta_{max}]$. Δ is then the explicit waiting time between uses of a particular server. The actual delta time in two uses of a given server may be larger, but may be no smaller.

2.2 Component Descriptions

2.2.1 The Nexus

The Nexus functions as a centralized data collection and storage point, managing all requests for traceroutes to be directed to individual servers, and disbursing the stored trace data to any interested clients. It services requests on a FIFO basis, while attempting to perform traces and data retrieval with a high degree of parallelism.

Due to the initial distributed nature of the project, measurements are time stamped from an external time source; here, this is according to the Time Protocol described in [1], providing second resolution on stamping of retrieved results. This is more than sufficient for the utilized infrastructure, as the delay bound on retrieving a traceroute measurement from an external source is measured not in milliseconds, but due to various aspects of the connection and the source machine, in seconds to minutes.[2]

Within the Nexus resides the actual request mechanism for requesting the execution of traces and the translation of their output into a standard format, termed *SlaveTraceRoute*. Due to both the simplicity and commonality of the mechanism, only those trace servers using HTTP GET as their communication mechanism are supported.

2.2.2 Stride

Stride contains the core measurement logic of the experiment. Essentially a simple client to the Nexus, Stride initiates a sequence of requests to the Nexus for measurements, relying on the Nexus' storage facilities for later data retrieval. Stride works in the following manner:

The Core Loop. Upon startup, the Nexus is queried for the list T of currently available servers; a copy of this list is then randomized and deposited in the ready queue of source nodes. A second copy is merged with a local list Q of "interesting nodes" to form a large list of targets; a copy of this list is then assigned to each source in the queue.

One by one, sources are now popped from the ready queue and activated. Activation causes the source node to randomly select from its list of remaining targets and request that the Nexus direct that source to perform a trace to that target. The target is then removed from that source's target list. A maximum of l source nodes are allowed to be contained within the active queue; as each source node is limited to one trace per activation, this provides a bound on the instantaneous network load.

Upon completion of activity, the now inactive source node is removed from the active queue, and a waiting period Δ is generated. This Δ is added to the current time, with the resultant sum used as the key for insertion into a wait queue, implemented as a standard min-priority queue.

Before insertion of a new node into the active queue, all nodes whose key values are greater than or equal to the current time are removed from the wait queue and appended to the end of the ready queue. For all free spaces in the active queue, nodes are now popped off of the ready queue and activated.

Execution continues as above for precisely $(t+q-1) \times t$ traces, exhausting all possible targets for each source. Once this iteration of the loop has completed, the listing of available servers is refreshed from the Nexus, and the cycle begins anew.

Effect of the Loop It is here that the name *Stride* was derived: requests are strided across multiple trace sources in parallel, in random fashion. This randomness is acomplished in three ways:

1. The initial selection of a source is randomized, having randomized the source vector on retrieval.

2. Upon activation, each source picks a new target at random from the remainder of its

[2] The most common of these factors is the DNS resolution performed by nearly all such traceroute servers on each node between the source and the target. In most cases, it is not possible to instruct them to avoid such resolutions. It is in part due to this and an oddly encountered tendency for some servers to "hang" in their response, that m was defined above.

copy of the target set Q.

3. Once selected, activated, and deactivated, a uniform random variable Δ is generated to bound the time that this source must wait before being used next; this is a lower bound, as once this wait has expired, the source is appended to the end of the ready queue behind some number of other ready sources.

2.2.3 The Harvester

The Harvester is a component in the wings whose purpose is to query selected Web search engines for the tell-tale signs of free traceroute servers. Upon finding a promising URL, the Harvester directs it to perform some basic measurements, comparing the resultant output against what one would expect to find from a useable server. Should it be usable, it is added to the current list of servers.

3 The Measurements

3.1 Dividing the World

The Stride architecture allows the analysis of arbitrary network regions, dependent only on the existence of traceroute servers in the area of interest.

There is a natural partition among the nodes of our graph provided by the Atlantic ocean, generating two essentially disjoint regions: North America and Europe. We choose to measure the two regions separately, due in no small part to the congested trans-atlantic transmission lines and interchange points to which those lines connect; it seems likely that our cost measurements would be dominated by these factors, eliminating much of the utility of a cross-Atlantic experiment.

Once the separate continental studies have been performed, we plan to verify the above assumption by merging the two sets of servers in a larger study.

3.2 Concretized Parameter Values

The method of determination of the proper values for Δ_{max} and Δ_{min} is necessarily imprecise; Δ is required primarily as a method of fending off over-use-based server attrition. In discussions of possible experimental enviroments, it was decided that the average time between traces should be fifteen minutes. More precisely, given the random variable $\Delta \sim \mathrm{U}(\Delta_{min}, \Delta_{max})$, $E(\Delta) = \frac{\Delta_{max} - \Delta_{min}}{2} = 15$ minutes. Let $\Delta_{min} = 0$. Then $\Delta_{max} = 30$. The instantaneous network load l and the maximum duration of the trace, m, are similarly derived: current heuristics suggest that $l = 60$ and $m = 60$ are appropriate, although this may change in the future. Q will consist of a set of high traffic websites, as yet not fully specified.

References

[1] J. Postel and K. Harrenstien. *Time Protocol*, RFC 868, 1983.

A Server Listing

A.1 North America

University of Washington, Seattle, WA http://weber.u.washington.edu/ geigudr/nph-traceroute.cgi?
Carnegie Mellon, Pittsburgh, PA http://www.net.cmu.EDU/bin/traceroute?
SD SuperComputing Center, San Diego, CA http://www.sdsc.edu/ hutton/cgi-bin/tracert.cgi?
Atmospheric @SUNY @Albany, Albany, NY http://www.atmos.albany.edu/cgi/trace.cgi?
Stanford, Stanford, CA http://www.slac.stanford.edu/cgi-bin/nph-traceroute.pl?
CCIT Telecom, U of AZ, Arizona http://WWW.Telcom.Arizona.EDU/cgi-bin/trace?
University of Wisconsin, Madison, WI http://www.medicine.wisc.edu/cgi-bin/traceroute?
MIT, Massachusetts http://bs.mit.edu:8001/cgi-bin/traceroute?
MIT, Massachussetts http://nic.mit.edu/bin/traceroute?
Internet Weather, Walnut Creek, CA http://www.internetweather.com/trace/nph-webtrace.cgi?
Yahoo, Santa Clara, CA http://net.yahoo.com/cgi-bin/trace.sh?
Structured.net, Portland OR http://www.structured.net/cgi-bin/traceroute.cgi?
Dimensional Communications, Denver, CO http://www.dimensional.com/cgi-bin/nph-traceroute?
Illuminati Online, Austin, TX http://www.tpoint.net/cgi-bin/trace?
Heller Inf. Services, Rocksville, MD http://his.com/cgi-bin/trace?
E-Z Net, Vancouver, WA http://www.e-z.net/cgi-bin/trace?
GetNet, Phoenix, AZ http://www.getnet.com/cgi-bin/trace?
Mag-Net BBS Systems, Prince George, BC, Canada http://www.mag-net.com/cgi-bin/nph-traceroute?
CWIX, Vienna, VA http://www.cwix.net/cgi/trace?
Lyceum Internet, Atlanta, GA http://zeus.lyceum.com/cgi-bin/nph-trace?
Peterson Data Mgmt, Vancounver, BC, Canada http://www.petersen.net/cgi-bin/nph-trace?
PICS On-line, Mount Holly, NJ http://www.pics.com/cgi-bin/trace-route?

Voice of America, Rocksville, MD http://voa.his.com/cgi-bin/trace?
AboveNet, San Jose, CA http://www.above.net/cgi-bin/trace?
NovaGate Communications, West Michigan http://www.novagate.com/cgi-bin/trace.cgi?
The Missing Link, Santa Clara, CA http://www.missing.com/trace.cgi?
Call America, San Luis Obispo, CA http://www.callamer.com/cgi-bin/cgiwrap/pbowden/trace.sh?
FMP Computer Services, Austin, TX http://www.fmp.com/cgi-bin/trace?
Home of the Bungi's, Bay Area http://www.bungi.com/cgi-bin/trace?
Techmart Sales Info, Mountain View, CA http://www.techmart.com/cgi-bin/trace.cgi?
Foothill.Net, Foresthill, CA http://www.foothill.net/cgi-bin/nph-traceroute?
Obnosis Consulting, Washougal, WA http://www.obnosis.com/cgi-bin/trace?
ATM Net , San Diego, CA http://www2.atmnet.net/cgi-bin/nph-trace?
XeniTec Consulting, Ontario, Canada http://www.xenitec.on.ca/cgi-bin/trace.cgi?
Lara ISP, Burlington, Ontario http://www.lara.on.ca/cgi1/111.cgi?
FrontierNet, Rochester, NY http://www.frontiernet.net/cgi-bin/trace-route?
Internet Global Services, Lewisville, TX http://develop.iglobal.net/cgi-bin/traceroute?
NTRNet System, Inc., Durham, NC http://www.ntrnet.net/cgi-bin/traceroute?
Interconnected Associates, Seattle, WA http://www.ixa.net/cgi-bin/trace?
Develop Company, Lewisville, TX? http://develop.iglobal.net/cgi-bin/traceroute?
Nobis, Inc., Wilmington, DE http://www.nobis.com/cgi-bin/trace-route?
NOC:Vector Internet Services, Minneapolis, MN http://noc.visi.com/cgi-bin/traceroute-you?name=
GlobalOne, Washington DC (believed) http://www.gsl.net/cgi-bin/traceroute.sh?
Red University, Mexico http://www.noc.unam.mx/cgi/trace.cgi?