

Discovering (and using) the State of the Net

Van Jacobson (filling in for Vern Paxson)
(van@ee.lbl.gov)

Network Research Group
Berkeley National Laboratory
Berkeley, CA 94720

DOE LSN Research Workshop
Reston, VA

January 5, 1998

The Internet has a monitor infrastructure (SNMP) but no measurement infrastructure.

As the scale increases and problems become more diffuse, users & network operators need solid measurement data to:

- diagnose performance problems
- study traffic patterns and their evolution
- assess path and “cloud” performance

There are (at least) three parts to the problem:

- acquire (good) measurements
- archive and make them available to users
- relate the measurements to application behavior

We're working on all of these.

NIMI — National Internet Measurement Infrastructure

NSF/DOE supported pilot project to develop an measurement infrastructure for the general internet.

Do this by creating inexpensive “measurement platform” that can be easily replicated and fielded.

NIMI Design goals and constraints:

Support wide range of active measurements.

Scale to \gg 1,000's of platforms (avoid success disaster).

Solid security designed in from the beginning.

Platform owners have full admin/policy control.

Platforms require minimal administration, maximal self-configuration.

Basic measurement requests:

- schedule participation in a measurement
- pick up measurement results
- cancel or suspend a schedule
- query current schedule

Security model:

Requests to a NIMI platform include the name of a *credential* and a corresponding cryptographically-secure signature.

Authentication based on access control lists (ACLs).

Rows correspond to credentials.

Columns correspond to actions.

⇒ You control access to a platform by controlling who has the corresponding credentials.

(Requests are also encrypted.)

Implemented on top of RSAREF.

Auto-configuration:

Each NIMI platform has one “master point of contact.”

In the minimially-configured state, platform just has a single ACL entry: master can modify entire table.

To boot, platform says “initialize me” to master.

NIMI Hardware/software platform:

Standard platform (< 3K\$):

- 200 MHz Pentium Pro
- 64 MB RAM
- 4 GB disk
- 10/100 Mb Ether
- modem
- optional GPS card

Standard OS: FreeBSD 2.2.2.

Administrative access: ssh.

Status:

Perl prototype running on PSC, LBNL platforms.

Includes autoconfiguration, ACL-based authentication, security, running measurements.

Does not yet include: management of results, querying and manipulation of measurement schedules.

Needed soon: an access GUI.

Canonical measurement: “poip.”

Poip (“Poisson ping”):

Sources/sinks UDP packets transmitted at Poisson intervals (or uniform or periodic).

Uses generic “wire time” library — interface to `libpcap`.

Myriad sanity checks on packet integrity.

Packet headers include: version, type, length, sequence, timestamp, MD5 checksum over payload.

Uses Anderson-Darling A^2 test to check sending times.

2,000+ lines of code (!).

Future:

Near-term: deployment at DOE sites.

Longer-term: deployment by volunteer ISPs (e.g., WorldNet).

Measurement orchestration, analysis frameworks.

Archiving results: reduction, navigation, visualization issues.

Integration with Surveyor, IPMA projects?

Meaty research issues:

Self-configuration: the “mapping problem.”

Integration with HOPS, SRM?

Going from individual results to problem diagnosis.

Aggregating results into higher-level statements about regions.

Holy grail: end-user clicks on a button to diagnose a problem; this queries distributed database; maybe causes additional measurements to be made, added to database.

Millions of platforms??

pathchar — helping users find path characteristics

`pathchar` is a tool similar to `traceroute` but designed to isolate congestion and loss problems rather than routing strangeness.

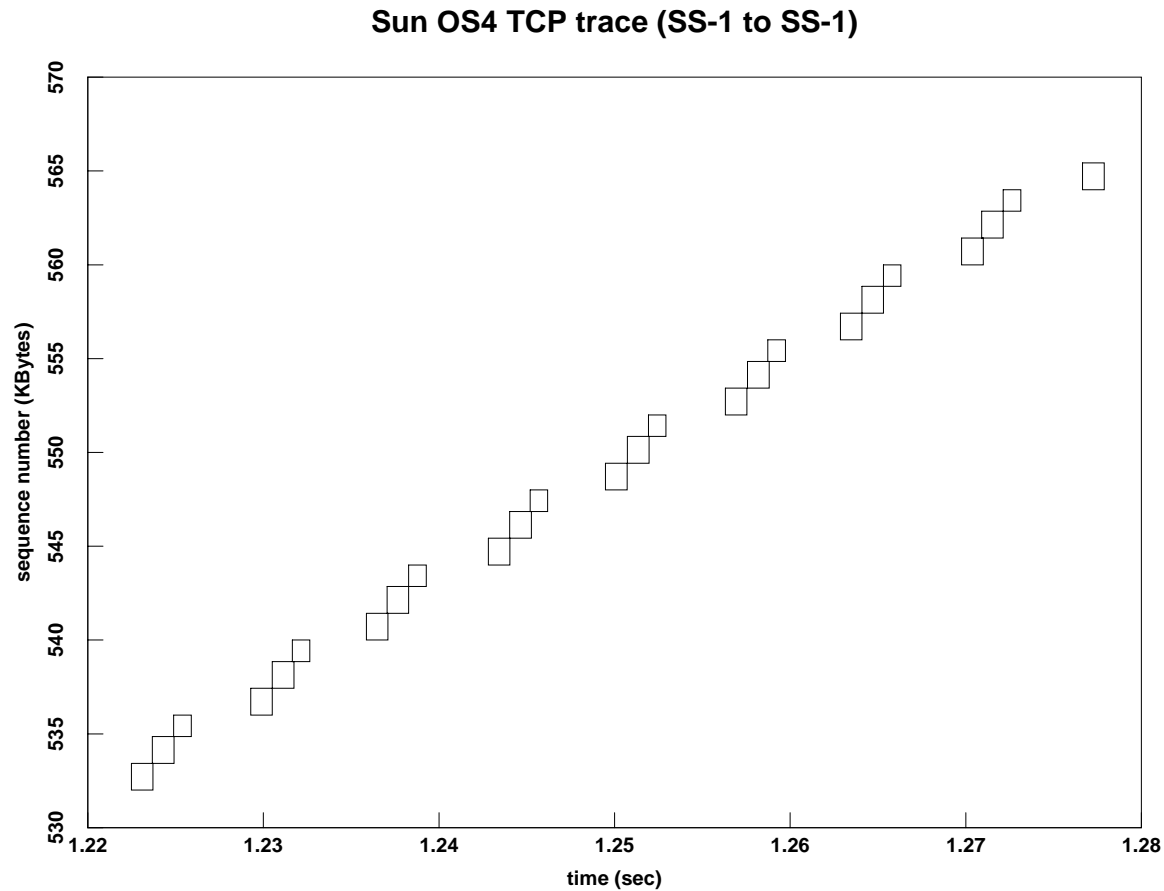
It determines the bandwidth, propagation delay, queue and loss rate for every hop on any Internet path.

It uses the same probe machinery as `traceroute` (i.e., modulate the packet time-to-live and look for ICMP “time exceeded” messages), will work wherever `traceroute` works, and can be run by anyone.

```
% pathchar www.cern.ch
mtu limited to 1500 bytes at local host
0 fry
|   30 Mb/s,   299 us
1 ir40gw.lbl.gov
|   29 Mb/s,   120 us
2 erlgw.lbl.gov
|  106 Mb/s,    -1 us
3 lbl-lc1-1.es.net
|   24 Mb/s,   34.6 ms
4 cebaf-atms.es.net
|   31 Mb/s,   2.89 ms, +q 8.08 ms
5 dccon-cebaf-mae-e.es.net
|   18 Mb/s,   1.58 ms, +q 8.10 ms
6 cern-dcconn.es.net
|   1.9 Mb/s,  43.9 ms, +q 8.05 ms
7 cernh8-s0.cern.ch
|  100 Mb/s,   91 us, +q 8.06 ms
8 cgatel.cern.ch
|   93 Mb/s,   67 us, +q 8.08 ms
9 r513-c-rci47-17-gb0.cern.ch
|   8.4 Mb/s,   25 us, +q 8.05 ms, 1% dropped
10 www.cern.ch
10 hops, rtt 167 ms, bottleneck 1.9 Mb/s, pipe 41223 bytes
```

tcpanaly — fixing application network performance problems

Observation: A lot of behavior is visible in a simple protocol trace:



Vern Paxson has developed a tool, `tcpanaly`, to analyze TCP protocol traces.

It currently knows about 10 different TCP implementations:

SunOS, Digital OSF/1, IRIX 4.0/5.1/5.2/6.2, VJ 1/2

BSDI 1.1/2.0/2.1, NetBSD 1.0, HP/UX 9.05/10.00,

Solaris 2.3/2.4, Linux 1.x/2.x

and the (mis-)features of each.

From a trace, it will automatically discover what kind of TCP each end of a conversation is using then describe all the end-node and network anomalies in the trace.

Future plans: Work with selected user community(s) to turn `tcpanaly` into a tool to help solve end-user network performance problems and tune distributed applications.