Web100 update

Matt Mathis
Google

Joint Techs, 7/14/2010

Outline

- Disclaimer
  - Everything here predates me "going Google"
- Web100
  - Motivation
  - Deliverables
  - Kernel philosophy
  - Library philosophy
  - Collateral impact to the network

Google™

Web 100 back story

- Wizard Gap (Miami Joint Techs, Dec 1999)
  - Ordinary users lag way behind the experts
- The TCP/IP hourglass
  - Hides the network and apps from each other
    - Critical for the growth of the Internet
  - Also hides all bugs
    - *Everything* is broken
    - TCP covers up the bugs by slowing down
- The only symptom is to slow down

Google™

Web100 approach

- Kernel Instrumentation
  - Use TCP's ideal vantage point
    - Passive observation of network events
    - Observe application behavior
  - Directly "fix" TCP's part of the problem
    - Sender side autotuning (predates Web100)
    - Receiver side autotuning
    - Negotiate appropriate SYN options

Google™

Web100 primary deliverables

- Standard for TCP Extended Statistics
  - RFC4898, May 2007
- Deploy autotuning
  - Linux, Aug 2004
  - Followed by Vista, OS X, Windows 7
  - Manual TCP tuning pages are (mostly) obsolete!

Google™

Additional results

- Running code (prototypes)
  - Kernel
  - Portable library
  - Example tools
- Widely deployed 3rd party tools (e.g. NDT)
- User community
  - Pro-active support during the project
  - Widely used research infrastructure
  - Downstream redistribution

Google™

Prototype kernel implementation

- Linux kernel patch (GPL)
  - Still maintained
- Predates the standard [RFC 4898]
  - Many name changes (e.g. Bytes->Octets)
  - Dropped instruments (8)
  - New instruments (42)
- Prototype is not production
  - Assumed it would be reimplemented
  - /proc performance and scale limits
    - ~31k maximum concurrent connections

Google™

Key Web100 feature

- Automated releases are critical
  - Repository HEAD -> patch/tarball
    - Internal release strings are timecodes
      head -1 /proc/web100/header
  - All testers work in parallel
    - And see the same things
    - Minimizes "it works for me" failures
  - Enabled "on the margin" unfunded support
    - Current relative to the Linux mainline
    - 6+ years after end of funding

Google™

Future proofed library

- Designed to be portable
  - Carefully avoided unnecessary dependencies
  - BSD style license to encourage broad adoption
    - So tools can be portable too!
- Layered modular design
  - API function
  - Instrument data types
  - Instrument names
    - Hooks for migration support
- API Function mimics SNMP
  - Attach to an agent (local, remote, or recorded)
  - Scan for interesting connections
  - Support for fast polling

Google™

Watch this space #1

- Web100 is overdue for an update

Google™

Changes to the network itself

- We expected autotuning to change traffic dynamics
  - Joking: "to cause the deployment of QoS"
- Due to autotuning:
  - Every long running TCP connection either
    - raises the loss rate
    - raises queuing delay
    - or both
- "TCP friendly" implies "instantaneous window fair", which is not an acceptable fairness paradigm

How TCP friendly really works

- Without autotuning most flows are buffer limited
  - Rate determined by RTT and buffer size
  - Mostly predictable performance
- With autotuning: flows tend to equalize window
  - Rate proportional to RTT at shared bottleneck
  - All long RTT flows suffer
- **Any** one short RTT flow squashes **all** long RTT flows
  - Very unpredictable performance

We need a new capacity sharing paradigm
We need IETF CONEX (CONgestion EXposure)
NB: All correct solutions resemble shaping

Google™

Watch this space #2

- The IETF is thinking about capacity allocation, fairness and congestion

Google™