Web10G: Stack Metrics for the Rest of Us

JET Meeting
October 21st, 2014
Chris Rapier
rapier@psc.edu
Why Stack Metrics Matter

• All performance problems look the same
  – Which complicates diagnosis considerably
    • This imparts a significant impediment to workflow
  – However, the TCP stack ‘knows’ quite a lot
    • It has to in order to respond to events properly
  – Getting what the stack knows to the user can help identify the cause of poor performance
    • Sadly, the stack isn’t instrumented. All we really have is a ‘check engine’ light.
Why Stack Metrics Matter 2

• Poor performance increases costs and decreases productivity
  – Resolution of performance problems takes time
    • Often because each problem has to be addressed from a position of no information
    • The interactive cycle with the user can take days if not weeks to resolve the problem

• Stack metrics can give engineers real time real world insight.
  – This can reduce user downtime and support staff effort
How To Get The Metrics

- Instrument the stack
- Bring the metrics out of the kernel
- Provide an API
- Build tools
- Simple!
Web10G: Making it Happen

• **Instrumentation**
  – RFC 4898 provides the basis of the KIS
    • 127+ different metrics based on known and inferred events in the TCP stack.
    • Duplicate acks, spurious retransmissions, timeouts, congestion window, sack blocks, congestion events, etc.
  – Currently supports Reno, BIC, CUBIC, & HTCP.
  – Each connection is maintained in a persistent yet stateless struct in kernel memory
    • Each instrument contains the *current* value and nothing more. No lifespan data in the kernel.
  – Relatively lightweight
    • Can support millions of connections
Web10G: Getting to the Data

• Need to move the data out of the kernel
  – Normally kernel memory is siloed from userland
    • However, there are methods to access some data
  – Proc is slow. Netlink (nl) is much faster and very well supported in the Linux kernel

• Web10g binary interface developed as DLKM.
  – Provides wrappers and entry points into KIS memory structs via a generic netlink (nl) family

• Other access methods can be built around nl using the Web10G nl family.
Web10g: Using the Data

- User side API developed to interact with netlink and process results
  - Relatively simple with a small number of calls.
    - Example code
      ```c
      Estats_nl_client_init(&client_list);
      Estats_val_data_new(&tcp_data);
      Estats_read_vars(tcp_data, cid, client_list);
      {...do stuff with tcp_data...}
      Estats_val_data_free(&tcp_data);
      Estats_nl_client_destroy(&client_list);
      ```
  - Can be incorporated into almost any existing application or build new tools easily
  - User only has access to their own connections
Web10g in Pictures

Application

API (Userland)

ABI (Kernel Module)

Linux Kernel

TCP Stack

KIS

RFC 4894

Web10G
Why Bother?

• More information
• Better tools
• Deeper insight into usage
The Insight Tool

• Three different types of network users
  – Those who know, those who expect too much, those who expect too little
  – Underutilization is a *real* problem

• How do we help those who don’t expect enough?
  – Give them a tool to visualize their flows
  – Point out poorly performing flows
    • Which is a *hard* problem
  – Let them easily report problems to the NOC
  – Teach them what to expect
The Insight UI
The Insight client

- Simple websocket server that monitors flows
- Accepts commands in JSON format
  - Stacked filters allow for fine tuning of the returned data
    - Filter on destination IP/mask, ports, and applications
    - Metric mask to limit results to specific data points
- Reports returned in JSON format
- Not tied to a specific UI.
  - Can be used as a base for other monitoring projects
- Can return reports directly to a RDBMS
The Insight NOC Tool

• Give NOCs an easy entry point into reported flow data
• Still barebones at the moment
  – Currently can find a view data in a table format
  – Hoping to add:
    • Reporting
    • Advanced search
    • Some level of data visualization
      – Particularly to show change over life of flow
Insight in Action

• Your patience please while we load the demo
Next Steps for Insight

• Insight tool is still too limited for most users
  – Dependencies create large barriers to entry
  – Cannot monitor 3rd party transfers
  – Still requires users to *watch* the flow

• Solution: Install in DMZs to monitor scheduled transfers
  – Provide a UI to the NOC with better alarming
    • This would include reporting and analysis to find trends
  – Give users access to real time visualization or post transfer reports in plain language
    • Authorization is going to be hard but doable.
Status of Web10g

• Core code is stable and deployable
• Working with teams at Google to prepare for submission to Linux kernel
• Tool development is on going

• http://www.web10g.org
• http://github.com/rapier1/web10g
• rapier@psc.edu