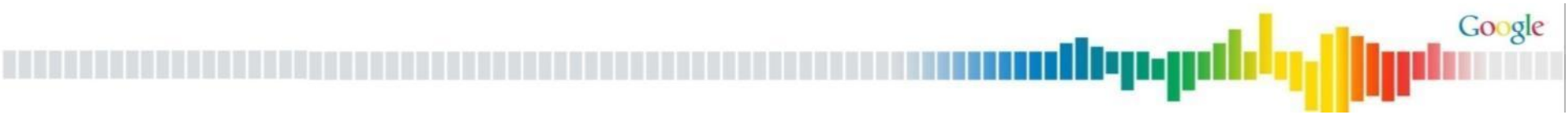
packetdrill: Scriptable Network Stack Testing, from Sockets to Packets

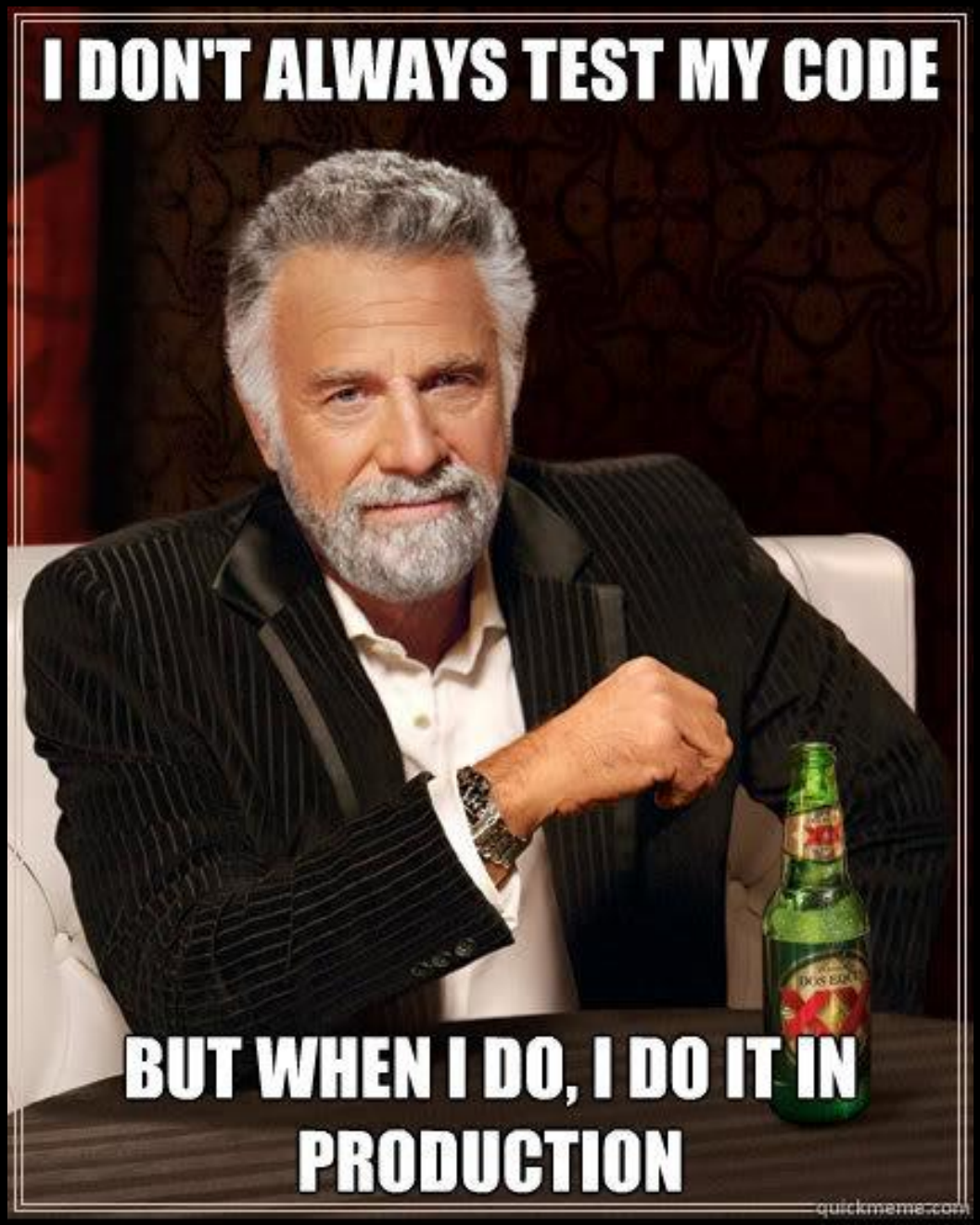
N. Cardwell, Yuchung Cheng, Lawrence Brakmo, Matt Mathis, Barath Raghavan, Nandita Dukkipati, Hsiao-keng Jerry Chu, Andreas Terzis, and Tom Herbert
Google

iccrg IETF87 - July 29, 2013

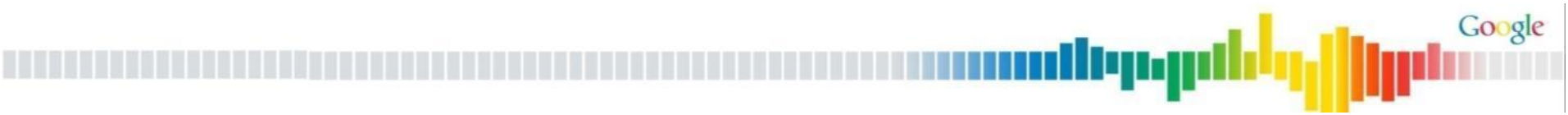
Modern networking stack



- Cornerstone of (large) distributed systems
- Complex & feature rich
  - Interact with many components
  - Regression test is critical
- Protocol / implementation bugs can crash the entire system
- How do people test and develop networking stack today?


I DON'T ALWAYS TEST MY CODE
BUT WHEN I DO, I DO IT IN PRODUCTION
quickmeme.com

Troubleshooting is another nightmare



From: angry-netops
To: net-dev
Subject: [urgent] TCP is hell slow

Here is a 20GB tcpdump sampled from 100 hosts. Figure out what's wrong.

From: pissed-netops
To: net-dev
Subject: [urgent] Kernel panic on TCP

at line tcp_input.c:4172. kernel version is "linux-3.x-super-stable". Fix it NOW.

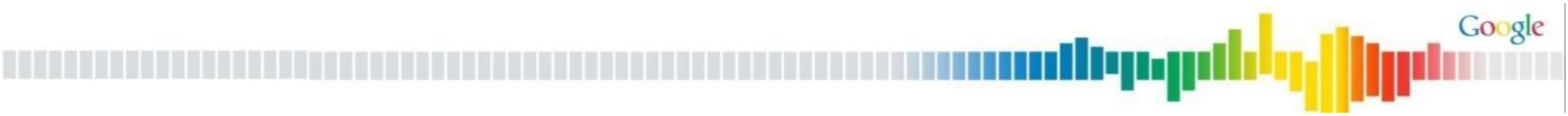
Packetdrill: a tool for scriptable network testing

- New features development
  - Black box unit tests
- Regression testing
  - netperf / load tests aren't enough
- Troubleshooting
  - Replay traces to reproduce production issues

... and for testing several aspects of network stacks:

- correctness - state machine handling corner cases
- performance - why did things get slower
- security - malicious attacks

The packetdrill Scripting Language: Design



- packets: tcpdump-like syntax
  - inbound packets to inject
  - outbound packets to expect
  - TCP, UDP, ICMP

- system calls: strace-like syntax
  - system calls to invoke
  - output to expect
  - blocking or non-blocking

- shell commands
  - to configure or inspect the machine under test
- Python scripts
  - assertions about internal socket state: TCP_INFO

Example: TCP Fast Retransmit


```
0    socket(..., SOCK_STREAM, IPPROTO_TCP) = 3
+0   bind(3, ..., ...) = 0
+0   listen(3, 1) = 0

+0   < S 0:0(0) win 32792 <mss 1000,nop,nop,sackOK,nop,wscale 6>
+0   > S. 0:0(0) ack 1 <mss 1460,nop,nop,sackOK,nop,wscale 6>
+.1  < . 1:1(0) ack 1 win 257
+0   accept(3, ..., ...) = 4

+0   write(4, ..., 4000) = 4000
+0   > .   1:1001(1000) ack 1
+0   > .   1001:2001(1000) ack 1
+0   > .   2001:3001(1000) ack 1
+0   > P.  3001:4001(1000) ack 1

+.1  < . 1:1(0) ack 1 win 257 <sack 1001:2001,nop,nop>
+0   < . 1:1(0) ack 1 win 257 <sack 1001:3001,nop,nop>
+0   < . 1:1(0) ack 1 win 257 <sack 1001:4001,nop,nop>
+0   > . 1:1001(1000) ack 1  // expect fast retransmit!
```

Example: test new API sendmsg(FASTOPEN)


```
//
// Test: poll() does not return until handshake completes in Fast Open
//

0.200 socket(..., SOCK_STREAM, IPPROTO_TCP) = 4
0.200 fcntl(4, F_SETFL, O_NONBLOCK) = 0

0.200 sendto(4, ..., 2000, MSG_FASTOPEN, ..., ...) = 1000
0.200 > S 0:1000(1000) <mss 1460,nop,nop,sackOK,nop,wscale 6,
                        FO abcd1234>

0.210...0.220 poll([{fd=4,
                     events=POLLIN|POLLOUT|POLLERR,
                     revents=POLLOUT}], 1, 100) = 1

0.220 < S. 1111:1111(0) ack 1001 win 60000 <mss 1040,nop,nop,sackOK>
```

Experiences with packetdrill

- Test Google production Linux kernel for 1.5 yrs
  - ~1000 test cases run in 30 min
  - Covers congestion control, loss recovery, buffer management, IPv4/v6 etc
  - Found (and fixed) dozens bugs in Linux TCP

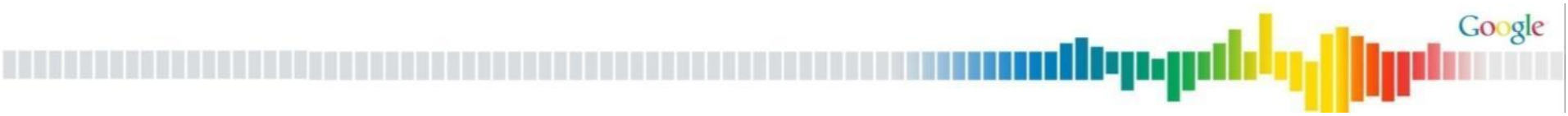
- Developed major TCP features
  - Fast Open, Tail Loss Probe, FEC, F-RTO re-impl, Proportional Rate Reduction

Conclusion

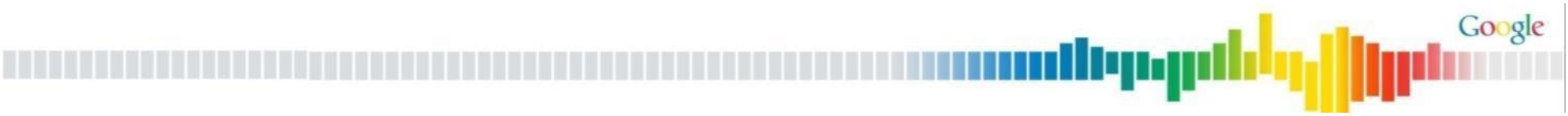
- Test *real* kernels on *real* hardware in *real* time
  - Socket APIs, TCP, UDP, offload mechanisms, IPv4 and IPv6
  - Linux, FreeBSD, OpenBSD, NetBSD
    - Portable to any POSIX
  - GPLv2

- Take packetdrill for a spin
  https://code.google.com/p/packetdrill/

Backup slides...



Our Test Suite for Google's Linux TCP



- large and growing
  - 347 scripts x {IPv4, IPv4-mapped-IPv6, IPv6}
    - 880 test cases total
- reproducible
  - rate of spurious failures (due to jitter): 1 in 2500
    - at current default timing tolerance of 4ms
- quick
  - 657 test cases in 26 minutes (2.4 sec per test case)

packetdrill Timing Models

- packetdrill allows flexibility in timing assertions
- supported timing models:
  - absolute: 0.750
  - relative: +0.100
  - range: 0.500~0.600
  - relative range: +0.100~+0.200
  - wildcard: *
- command line option for default tolerance for all events:
  - --tolerance_usecs=800
- blocking system calls: 0.100...0.200