On Inferring TCP Behavior

Jitendra Padhye
Sally Floyd

AT&T Center for Internet Research at ICSI
(ACIRI)

http://www.aciri.org/tbit/
TBIT: TCP Behavior Inference Tool

An *active* tool to infer TCP behavior of Internet hosts.

In this talk:

- Motivation
- How it works and what makes it different
- Selected results from a survey of TCP behavior of web servers
Motivation

- TCP handles a majority of today’s Internet traffic
- Understanding TCP behavior is important:
  - OS vendors and customers: better/correct implementations
  - Networking research: measurement, modeling
  - Standards organizations
- TCP behaviors of web servers is of special interest
Understanding TCP behavior

- TCP is a complex protocol. Many variants.
- Standards document specify many options.
- Need to understand TCP behavior on two fronts:
  - Mathematical modeling
  - Understanding real implementations
Example

- Initial window used by TCP: amount of data sent out in a “burst” before any ACKs are received.
- RFC 2414: min (4*MSS, max (2*MSS, 4380 bytes))
- We have found TCPs that send 8000+ bytes with MSS of 512!
- Large bursts of packets ⇒ buffering problems, loss, delays.
How to test implementations?

- Passive monitoring [Paxson 97]
  - Right conditions must occur during test period

- Controlled laboratory tests [Gao and Madhavi 2000]
  - Can not uncover information about popular configurations etc.

- Active testing [TBIT]
Salient features of TBIT

- Ability to test any web server at will
  - No special privileges needed on servers
  - Robust to prevailing networking conditions
  - Traffic generated should not appear hostile

- Modular, extensible architecture
How it works: The basic idea

- Send “fabricated” TCP packets over raw IP sockets.
- Host firewall prevents kernel from seeing response packets.
- BPF delivers blocked packets to user process.
- Net effect: a user-level, user-controllable TCP, without kernel changes.

Based on “Sting” project at Univ. of Washington by Stefan Savage
How it works: An example

Determine TCP initial window used by a web server.

- Send SYN. Wait to receive SYN-ACK.
- Send HTTP 1.0 GET request for “/”
- Do not ACK any incoming packets.
- Wait until first retransmission.
- Initial window $= \text{Max. sequence number received.}$
How it works: Difficulties

- Too few packets: set smaller MSS?
- Lost packets: repeat test multiple times.
- Multiple hosts answering same IP address: non-repeatable results?
- No easy way to test without a web server.
Tests implemented so far

- **Handshake tests**: Timestamp used? SACK-capable?

- **Congestion response**: Reduce congestion window? NewReno/Reno/Tahoe?

- **SACK**: Construct SACKs correctly? Respond to SACKs correctly?

- **Other**: Initial window? ECN-capable?
Experimental setup: 1

- Several lists of web sites:
  - 100hot.com
  - ISP proxy trace (Dax Kelson)
  - List from [Arlitt and Krishnamurthy 2001]
- Total 4550 unique IP addresses
- Each host returns at least 3000 bytes when base page is requested.
Experimental setup: 2

- NMAP ran against each host to provide OS guess
- Each tests repeated at least five times.
- Results reported only if at least three tests complete, and all completed tests return consistent results.
TCP flavor

- Based on tests in [FF96]
- Results based on 3728 hosts. (out of 4550).
- NewReno most popular
- Surprise: 1010 show no fast retransmit: timeout for any packet loss in a window
- Windows bug (for small transfers?). Microsoft acknowledges, but not fixed.
NewReno

www.june.com 195.81.253.100

Sequence Number vs Time graph showing packet reception, acknowledgment, and drops.
No Fast Retransmit

www.attach.net 209.150.120.5
SACK usage

- 1759 of 4550 hosts negotiate SACK during SYN exchange. But do they use it correctly?

- Drop three packets from a large window

- Correct SACK usage: Packets retransmitted in a single RTT

- Results based on 1309 hosts

- 759 of these do not appear to use SACK information.
Correct SACK usage

www.earthlink.net 207.217.114.200

Sequence Number vs. Time

- Rcvd
- Ack
- Drop
SACK info not used

www.coda.cs.cmu.edu 128.2.194.223

Sequence Number

0 0.8 0.85 0.9 0.95 1 1.05 1.1 1.15 1.2

Time

Rcvd  Ack  Drop

0 500 1000 1500 2000 2500 3000 3500

0 0.8 0.9 0.95 1 1.05 1.1 1.15 1.2
Result highlights: 1

Quantitative data regarding:

- Deployment of TCP variants: Tahoe, Reno, NewReno etc.
- SACK deployment in servers, SACK correctness
- Initial Window sizes
- MSL durations
- Delayed Acknowledgment
Result highlights: 2

Bug detection and fixes:

- IBM: Timestamp option processing.
- Sun: Response to single packet drop in Solaris 2.5
- Cisco: ECN option processing. (joint with Dax Kelson)
Future work

- Make tests more robust.
- Additional tests: Slow start, RTO . . . .
- A ”server” version of TBIT
- Automatic generation of NS models.
- Extend this approach to investigate other behaviors of the Internet infrastructure
Finally ....

- Source code and detailed results: 
  http://www.aciri.org/tbit/

- We encourage people to use the software and add their own tests.