A Platform for Unobtrusive Measurements on PlanetLab

Rob Sherwood    Neil Spring

University of Maryland
http://www.cs.umd.edu/projects/sidecar
Need for Measurements

Measurements benefit many applications:
- Performance optimization [OASIS]
- Overlay construction [i3]
- Network diagnosis [PlanetSeer],[CoMoN],[iPlane]

Grand Challenge:
- Collect a “Day in the Life”
  - CSTB *Looking Over the Fence* report
Why Not Just Measure Everything?

for src in PlanetLab do
    ssh to src
    for dst in All Addresses do
        Traceroute dst
        Ping dst
        Pathchar dst
        Other measurements ...
Why Not Just Measure Everything?

PL02

Send Probes
Why Not Just Measure Everything?

PL02

Limited
Bandwidth
Why Not Just Measure Everything?

PL02

Limited Bandwidth

Drop Probes

Firewall
Why Not Just Measure Everything?

PL02

Limited Bandwidth

Abuse Reports

Drop Probes

Firewall
Why Not Just Measure Everything?

PL02

Limited Bandwidth

Abuse Reports

Drop Probes

Firewall
Abuse Reports

- Measurement traffic exceptional
- Exceptional == suspicious
- Reports of network abuse are handled with care
  - Thank you Mark Huang and PL Staff!
- Curtails experiment scope
Measurement Platform: Sidecar

Inject probes into normal traffic:
- Probes are retransmissions
- Avoids abuse report
- Allows firewall/NAT traversal

General measurement platform:
- Latency — Sideping
- Bottleneck location — Artrat
- Topology — Passenger [IMC06]
Talk Overview

- How Sidecar works
- Learning from Sidecar
- What Sidecar can do
Talk Overview

- How Sidecar works
- Learning from Sidecar
- What Sidecar can do
Sidecar Probes

Sender

Sidecar

Receiver

Data

ACK
Sidecar Probes

Sender  Sidecar  Receiver

Data

ACK  Dup Data
Sidecar Probes

Sender

Data

ACK

DUP ACK

Sidecar

Dup Data

Receiver
Sidecar Probes: Sender’s View

Sender

Data

ACK

DUP ACK

Receiver

Delayed or Duplicate Data
Sidecar Probes: Receiver’s View

Sender

Data

ReXmit Data

ACK

× Dropped

ACK

Receiver
Firewall and Nat Traversal

Sender  Sidecar  NAT  Receiver

Data

ACK

DUP ACK

Dup Data
Sidecar Probes

- Probes are retransmissions
  - Requires no end-point support
  - Send probes when connection is idle

- Modify probes for specific measurement
  - Reduce TTL
  - Send probes in train
  - Add IP options

- Can send probes after connection closes
  - Receiver in TIME_WAIT state
libpcap filter: “tcp port 80”
Sidecar API

- Tracks connection state and data

<table>
<thead>
<tr>
<th>Conn. 1</th>
<th>Conn. 2</th>
<th>Conn. N</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ESTAB</strong></td>
<td><strong>TIMEWAIT</strong></td>
<td><strong>SYNSENT</strong></td>
</tr>
<tr>
<td>547 bytes recorded</td>
<td>3K bytes recorded</td>
<td>0 bytes recorded</td>
</tr>
</tbody>
</table>

libpcap → Sidecar → raw socket → Network → Tool

New connection
Packet send()
Applications register callbacks

Events: new connection, probe returned, idle, close, timeout

Connect 1
ESTAB
547 bytes recorded

Connect 2
TIMEWAIT
3K bytes recorded

Connect N
SYNSENT
0 bytes recorded

new connection

Packet send()
Example Code for Sidecar Tool

1. `sc_register_connection(connectCB);`
2. `sc_init(“tcp port 80”);`
3. `connectCB(conn *c){
   sc_register_idle(c,idleCB);
   sc_register_in(c,inCB);} `
4. `idleCB(conn *c){
   send_probe(c);
   sc_register_timeout(c,timeoutCB) }`
5. `inCB(conn *c, packet *p){
   print “Response@” + calcRTT(p)}`
Talk Overview

- How Sidecar works
- Learning from Sidecar
- What Sidecar can do
What We Learned: Overview

- Sidecar generates no abuse reports
- Generate traffic carefully
- Clocks are not accurate
- Causally related packets are reordered
- Firewalls unset DF bit
- IO systems calls lag
Probed all traffic to CoDeeN clients
Experiment ran for 1 week
Sidecar traceroute to each client
13.4M hosts probed
No abuse reports generated
Instrument custom web crawler with Sidecar probes
- 168K web servers × PL Nodes
- Caused ten abuse reports
- ...but from web crawler, not Sidecar

Correct User-Agent, Virtual hosts

Crawlers synchronized → traffic spikes

- N1: AAABBCD
- N2: AAABBCD
- N3: AAABBCD
- N4: AAABBCD
Instrument custom web crawler with Sidecar probes
- 168K web servers × PL Nodes
- Caused ten abuse reports
- ... but from web crawler, not Sidecar

Correct User-Agent, Virtual hosts

Crawlers synchronized → traffic spikes

N1: ABCD
N2: ABCD
N3: ABCD
N4: ABCD
Instrument custom web crawler with Sidecar probes
- 168K web servers × PL Nodes
- Caused ten abuse reports
- ...but from web crawler, not Sidecar

Correct User-Agent, Virtual hosts

Crawlers synchronized → traffic spikes

<table>
<thead>
<tr>
<th>N1</th>
<th>N2</th>
<th>N3</th>
<th>N4</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABCD</td>
<td>BACD</td>
<td>ADBC</td>
<td>DCAB</td>
</tr>
</tbody>
</table>
Instrument custom web crawler with Sidecar probes
- 168K web servers × PL Nodes
- Caused ten abuse reports
- ...but from web crawler, not Sidecar

Correct User-Agent, Virtual hosts

Crawlers synchronized → traffic spikes

Application logs
Clocks would change rate, jump backwards
  - Similar to [Myths05]

PlanetLab nodes have diverse hardware

Future work: add RDTSC sanity check
  - Signal tool that clock jumped
VNET + libpcap interaction caused causally related packets to arrive out-of-order
- e.g., ACK before SYN|ACK
- Replicated by other researchers

Time stamps correct

Made state transition logic more robust

Future: patch VNET code
Causal Packets Reordered

- VNET + libpcap interaction caused causally related packets to arrive out-of-order
  - e.g., ACK before SYN|ACK
  - Replicated by other researchers
- Time stamps correct
- Made state transition logic more robust
- Future: patch VNET code
Recursive packet train measurements overflowed libpcap [Sigcomm04]

Tried to use DF bit to ignore payload
- Some firewalls unset DF on incoming packets

Implications for MTU discovery measurements
Many concurrent writes

*strace* -T showed that *open()* and *write()* calls took 1-3 seconds to return

Intermittent; could not diagnose
Talk Overview

- How Sidecar works
- Learning from Sidecar
- What Sidecar can do
**Sideping**

- 24/482 PL nodes drop ICMP Echo
  - All nodes allow Sidecar probes
- Sideping traverses firewalls and NATs
  - Exposes higher latency extra hop

![CDF of RTT (ms)](image)

- CDF
- RTT (ms)

**Plot Details:**
- X-axis: RTT (ms)
- Y-axis: CDF
- Two curves:
  - ICMP Echo
  - Sideping
Artrat: Active Receiver-side TCP Rate Analysis Tool

- Locate bandwidth bottleneck from receiver
- Sanity check PlanetLab experiment conditions
Artrat: Technique

- Use IP timestamp option with ICMP echo to measure queuing delay
- Router with highest correlated delay is bottleneck
Conclusion

- Sidecar is a technique and API/package for unobtrusive probing
- Probes caused no* abuse reports
- Symbiotic relationship between service and measurement projects
  - Measurements ⇔ application traffic
- Download API and tools from
  http://www.cs.umd.edu/projects/sidecar
2. [i3] http://i3.cs.berkeley.edu/
3. [PlanetSeer]
1. [IMC06] “Touring the Internet in a TCP Sidecar”
Rob Sherwood, Neil Spring.


3. [Sigcomm04] “Locating Internet Bottlenecks: Algorithms, Measurements and Implications”.
Ningning Hu, Li Erran Li, Zhuoqing Morley Mao, Peter Steenkiste, Jia Wang