On-Demand IPv4 and IPv6 Topology Measurements

Young Hyun
CAIDA

ISMA 2012 AIMS-4 Workshop
Feb 9, 2012
Introduction

* common need: **measure Internet topology from multiple vantage points**
  * RTT and reachability from pings
  * IP paths from traceroute
Introduction

* common need: measure Internet topology from multiple vantage points
  * RTT and reachability from pings
  * IP paths from traceroute

* useful for studying ...
  * network performance, outages, and censorship
  * routing stability, optimality, and resiliency
  * address space usage: routed vs. occupied
  * AS relationships and global Internet structure/evolution
  * router- and PoP-level maps
  * geolocation
Introduction

* desiderata for measurement facility
  * do not require accounts on individual monitors
    • secure, single point of access to many vantage points
  * support bulk measurements (hundreds of 1000’s)
  * support varying levels of complexity
    • simple to learn and use for simpler tasks; slightly harder for harder tasks; makes complex tasks possible
  * support adaptive measurements
    • dynamic and feedback-driven
  * be scriptable: schedule probes and select vantage points and targets under program control
    • hard to design a non-scriptable scheduling system (e.g., a job submission GUI) that is flexible enough to handle complex non-uniform schedules
Introduction

* topo-on-demand (tod) service on Ark
  * scriptable interface for performing IPv4 and IPv6 traceroutes and pings
  * measurements from 57 Ark monitors (28 with IPv6)
    * globally distributed in both commercial and R&E networks
  * supports varying levels of user sophistication and needs
a client accesses the topo-on-demand service through an Ark server

* client remotely connects to a single access point
* client requests measurement from an Ark monitor
  * can issue multiple concurrent requests
* client receives results asynchronously

request: ping 192.172.226.123 from ams-nl

result: RTT 163.355ms
benefits of single-point of access

- clients do not need login accounts on Ark monitors
- clients do not need to implement software to manage a distributed system
  - issuing requests to and collecting results from remote monitors
topo-on-demand service is decentralized

Ark server only provides communication access

client measurement scripts communicate with the topo-on-demand server on each monitor with Marinda

Marinda is Ark software that provides a high-level communication abstraction, a *tuple space*
Marinda

* **tuple space**: a distributed shared memory + operations
  * clients store and retrieve tuples
    * retrieval by pattern matching
  * **tuple**: an array of values
    * strings, numbers, true/false, wildcard, nested arrays

* Marinda is used for decentralized communication and coordination
  * simplifies network programming in a distributed system
  * provides, for example,
    * message-oriented synchronous and asynchronous group communication
    * a persistent connection (reconnects transparently after loss)
    * automatic marshaling of structured data (tuples)
Client Access

* varying levels of user sophistication and needs
  * case 1: simply want to probe a set of targets in a file and save results to a file
    - for example: ping all targets from a single monitor
  * case 2: want greater control of measurements; possibly adaptive (dynamic, feedback-driven)
    - case 2a: want to use any choice of implementation language
    - case 2b: willing to use Ruby and Marinda directly
Client Access

- tod-client
- tod-feed-targets
- simple address list
- Marinda
  - client measurement scripts (Ruby + Marinda)
  - client measurement scripts (any language)
- client computer
case 1: simply want to probe a set of targets in a file and save results to a file

- targets file with one address per line
- use provided client access tools:
  
  ```
  $ cat targets | tod-feed-targets --source=san-us --ping --options=attempts=1 | tod-client > results
  ```
case 2a: want greater control of measurements; possibly adaptive; *want to use any language*

- write a measurement script that issues commands to `tod-client`
  
  ```bash
  $ doit1 | tod-client >results
  ``

- for feedback-driven measurements, write a script that `popens` `tod-client`
  
  - that is, run `tod-client` as a subprocess that can be written to and read from
  
  ```bash
  $ doit2 >results
  ```
**tod-client**

- tod-client provides a text-based gateway to the topo-on-demand service
  - familiar Unix shell paradigm: programs communicating through pipes
  - tod-client accepts text commands on stdin and writes measurement results to stdout
  - designed to be run as a subprocess and to provide high throughput
    - accepts any number of commands without blocking client
    - executes measurements asynchronously and in parallel
example of use:

- `-h` ⇒ human readable output

$ tod-client -h
1 san-us ping 192.172.226.123

ping from 192.172.226.5 to 192.172.226.123
  1:  192.172.226.123    0.092 ms  64 TTL
  2:  192.172.226.123    0.112 ms  64 TTL
  3:  192.172.226.123    0.166 ms  64 TTL
  4:  192.172.226.123    0.079 ms  64 TTL
tod-client

Example of use:

$ tod-client -h

2 lax-us trace 192.172.226.123

traceroute from 137.164.30.25 to 192.172.226.123
  1.1:  137.164.30.1    0.183 ms
  2.1:  137.164.46.105    0.787 ms
  3.1:  137.164.46.54    2.623 ms
  4.1:  137.164.47.15    9.649 ms
  5.1:  137.164.23.130    9.681 ms
  6.1:  132.249.31.6    9.903 ms
example of use:

$ tod-client

1 san-us ping 2001:48d0:101:501::132 attempts=1

2001:48d0:101:501::132 0 1 1328149101 R
0.353 1 64 S 0
2001:48d0:101:501::132,0.353,64

2 lax-us trace www.caida.org attempts=1,method=icmp-paris

2 data www.caida.org T 137.164.30.25 192.172.126.123 0
1 1328145600 R 9.766 7 58 S
0 C 137.164.30.1,0.147,1
137.164.46.105,1.045,1 137.164.46.54,2.559,1
137.164.47.15,9.750,1 137.164.23.130,17.992,1
132.249.31.6,9.886,1
a command is a single line of structured text:

```
<request_id> <source> <command> <target> <options>
```

1 san-us ping www.caida.org attempts=1
2 lax-us trace www.caida.org attempts=1,method=icmp-paris

* <request_id>: arbitrary numeric value provided by client
  * used by client for probe-response matching
* <source>: Ark monitor
* <target>: IPv4/IPv6 address or hostname
  * hostname resolved on monitor; useful for probing anycast targets
* <options>: scamper ping/traceroute options
  * src/dest port, initial/max TTL, probing method (TCP, ICMP, UDP, paris versions), attempts, wait time between attempts, probe size, TOS, payload bytes, etc.
**measurement result: single line of tab-delimited fields**

\[
\text{<request_id> <type> <target> <data_1> ... <data_n>}
\]

<table>
<thead>
<tr>
<th></th>
<th>data <a href="http://www.caida.org">www.caida.org</a></th>
<th>P</th>
<th>192.172.226.5</th>
<th>192.172.226.123</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1328145562</td>
<td>R</td>
<td>0.297</td>
<td>1</td>
<td>64</td>
</tr>
<tr>
<td>0</td>
<td>192.172.226.123,0.297,64</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>data <a href="http://www.caida.org">www.caida.org</a></th>
<th>T</th>
<th>137.164.30.25</th>
<th>192.172.226.123</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1328145600</td>
<td>R</td>
<td>9.766</td>
<td>7</td>
<td>58</td>
</tr>
<tr>
<td>0</td>
<td>C</td>
<td>137.164.30.1,0.147,1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>137.164.46.105,1.045,1</td>
<td>137.164.46.54,2.559,1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>137.164.47.15,9.750,1</td>
<td>137.164.23.130,17.992,1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>132.249.31.6,9.886,1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3 error "1234.1234.1234" "malformed target or couldn't resolve hostname to IP address"
special command syntax:

```
<request_id> <source> <command> <target> <options>
```

1. `@any ping @prefix=192.172.226.0/24`
2. `@any ping @ark=san-us`
3. `@any:ipv6 ping @ark=san-us:ipv6`
4. `@any:ipv6 ping @ark=any:ipv6`

- `@any` ⇒ pick any Ark monitor (`@any:ipv6` ⇒ that has IPv6); pick a different monitor on each use, cycling through monitors in random order
- `@prefix=<IPv4/IPv6 prefix>` ⇒ pick a random destination in prefix
- `@ark=<monitor> / @ark=any` ⇒ use an Ark monitor as the destination (`:ipv6` ⇒ probe IPv6 address)
Client Access

* case 2b: want greater control of measurements; possibly adaptive; *willing to use Ruby and Marinda*
  
  * write a measurement script that interacts directly with the topo-on-demand servers via Marinda
    - allows for maximum flexibility and control
  
  * actually fairly easy to do ... sample code in next slide
#!/usr/bin/env ruby

require 'rubygems'
require 'marinda'

$c = Marinda::Client.new(UNIXSocket.open("/tmp/localts.sock"))
$c.hello
$tod = $c.open_port 2000, true

# 2 lax-us trace www.caida.org attempts=1,method=icmp-paris
$tod.write ['"TRACEROUTE", "ark", 2, "lax-us", "www.caida.org",
             ["attempts", 1], ["method", "icmp-paris"]]
result = $tod.take ['"RESULT", "ark", nil, nil, nil, nil, nil, nil]
p result

$ ./tod-example
["RESULT", "ark", 2, "lax-us", "www.caida.org", "data", "T
 \t137.164.30.25\t192.172.226.123\t0\t1\t1328226507\tR\t9.838\t7
 \t58\tS\t0\tC\t137.164.30.1,0.176,1\t137.164.46.105,1.110,1
 \t137.164.46.54,3.015,1\t137.164.47.15,9.681,1
 \t137.164.23.130,10.178,1\t132.249.31.6,9.860,1"]
Future Work

* web interface to topo-on-demand service
* possibly give accounts on a CAIDA box to conduct topo-on-demand measurements
  * remove need to install software by users
* possible support services
  * BGP queries (e.g., current route to a given destination)
  * pick random destination in an AS/country/organization
  * IP to AS/prefix mapping
  * IP to router mapping (via ITDK)
  * geolocation lookups (via MaxMind’s free database)
Thanks!

For more information or to request data:

www.caida.org/projects/ark

For questions, or to offer hosting: ark-info@caida.org