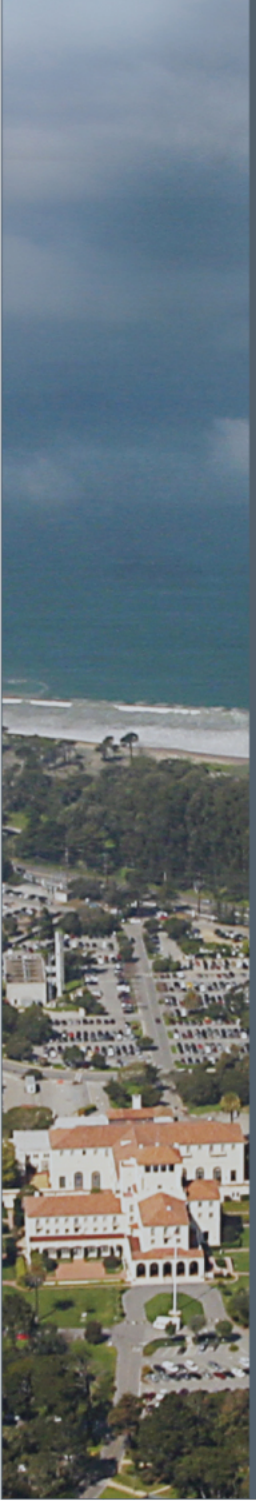

Large Scale Measurement Machinery

ArkQueue and Scamper Tools

Justin P. Rohrer Robert Beverly

Center for Measurement and Analysis of Network Data
@NPS
{jprohrer,rbeverly}@nps.edu

March 31, 2015

Outline

Outline

- Ark Background
  - what is Ark?
  - user's perspective
  - how do we tell it what to do?
  - how do we get useful data?
- ArkQueue
  - python module for interfacing with Ark
  - (i.e. what commonly goes wrong, and how we fixed it)
- Scamper Tools
  - probing with Scamper
  - Scamper data collection
- Ark wishlist

Ark Background

Archipelago (Ark) is CAIDA's next-generation active measurement infrastructure and represents an evolution of the skitter infrastructure

- Practically speaking?
  - many geographically distributed vantage points, contributed or hosted by volunteers
  - supporting infrastructure
- Usage:
  - CAIDA uses ark to systematically collect topology data (e.g. ITDK)
  - CAIDA grants access to other researchers to conduct measurements via Topology on Demand (ToD)
  - performance is non-deterministic due to shared resources and distributed host institutions

106 (and growing) Ark Monitor Locations

Figure: `http://www.caida.org/data/monitors/monitor-map-ark.xml`

CAIDA's Ark:

- We have all (at least) heard of CAIDA's Archipelago...
- And maybe even used it?

A User's Perspective

We've been using Ark for several years, e.g.

- ***Spoofer project:*** receive/coordinate spoofed probes from clients [IMC09]
- ***Net mapping:*** efficient primitives [IMC10], ingress point spreading [PAM14]
- ***TCP HICCUPS:*** ascertain path-mangling within TCP to cooperate with middleboxes [SIGCOMM14]
- ***Router geolocation:*** via landmark active probing
- ***IPv6 mapping:*** exhaustive probing of all /48's in all /32's to understand IPv6 subnetting, IPv6 probing heuristics

A User's Perspective

We've been using Ark for several years, e.g.

- ***Spoofer project:*** receive/coordinate spoofed probes from clients [IMC09]
- ***Net mapping:*** efficient primitives [IMC10], ingress point spreading [PAM14]
- ***TCP HICCUPS:*** ascertain path-mangling within TCP to cooperate with middleboxes [SIGCOMM14]
- ***Router geolocation:*** via landmark active probing
- ***IPv6 mapping:*** exhaustive probing of all /48's in all /32's to understand IPv6 subnetting, IPv6 probing heuristics

I can talk about these more later...

A User's Perspective

We've been using Ark for several years...

- It's powerful and useful for measurement research
- Have run into most issues others are likely to encounter
- (Issues either of the system, or the user's perception of how it's supposed to work)

A User's Perspective

We've been using Ark for several years...

- It's powerful and useful for measurement research
- Have run into most issues others are likely to encounter
- (Issues either of the system, or the user's perception of how it's supposed to work)

To ease our own pain, we created some tools that may be of use to the community

1. Provide host where probes will be initiated and get it authenticated by CAIDA

1. Provide host where probes will be initiated and get it authenticated by CAIDA
2. Use `tod-client` command-line interactive application on host to submit probe requests and get results

   Probe request "`<probeID> <monitor> <cmnd> <target>`"

1. Provide host where probes will be initiated and get it authenticated by CAIDA
2. Use `tod-client` command-line interactive application on host to submit probe requests and get results

   Probe request "`<probeID> <monitor> <cmnd> <target>`"
   Probe result e.g.

```
<probeID> data <target> 2001:470:1f06:ee1::2 0 1 1424475896 R 78.375 7
2001:610:1:80bb:192:87:102:97,0.432,1 2001:610:f01:9168::169,21.344,1
2001:7f8:1::a500:6939:1,12.839,1 2001:470:0:2d0::1,6.412,1
2001:470:0:2cf::2,72.077,1 2001:470:0:5d::2,75.762,1
```

3. Parse results

Simple, right?

ToD/`tod-client` usage issues

- results arrive asynchronously after seconds, minutes, or never...
- typical (small) experiment requires >100,000 probes
- and submitting *too many* requests at once may break Ark
- and may require requests to be a function or prior results

During any given experiment some subset of Ark monitors down or too slow

- subset changes over time
- Ark/ToD does not expose monitor status
- causes head-of-line blocking

- Scenarios we kept experiencing
- Students, ourselves, and general ToD-newbies

- Scenarios we kept experiencing
- Students, ourselves, and general ToD-newbies
- Student: "**Which vantage points should I use...**"
  - How to select randomly? How to select up/responsive monitors? How to know what are the monitors in the first place?

- Scenarios we kept experiencing
- Students, ourselves, and general ToD-newbies
- Student: "**Which vantage points should I use...**"
  - How to select randomly? How to select up/responsive monitors? How to know what are the monitors in the first place?
- Student: "**Which VP did this result come from...**"
  - ToD responses only include (potentially private IP) of source

- Student: "**My experiment stopped running...**"
  - typically caused by waiting for results from downed monitor
  - limited number of probes can be "in-flight" ($\simeq 100$)
  - even when choosing monitors at random, eventually all in-flight probes waiting on downed monitor

- Student: "**My experiment stopped running...**"
  - typically caused by waiting for results from downed monitor
  - limited number of probes can be "in-flight" ($\simeq 100$)
  - even when choosing monitors at random, eventually all in-flight probes waiting on downed monitor
- Student: "**Ark is b0rken...**"
  - by multiple tod-client processes using same session ID
  - ...or a downed monitor
  - ...or a non-existent monitor
  - ...or a monitor busy doing other things
  - ...or a monitor that doesn't support the command (e.g. IPv6)

- Student: "**Ark is b0rken (again)...**"
  - submitting millions of requests to monitor X
  - program wrapping tod-client died/crashed
  - requests at monitor X still pending
  - new requests seemingly unresponsive (monitor still busy with old/stale requests that will never be fetched)

- Student: "**Ark is b0rken (again)...**"
  - submitting millions of requests to monitor X
  - program wrapping tod-client died/crashed
  - requests at monitor X still pending
  - new requests seemingly unresponsive (monitor still busy with old/stale requests that will never be fetched)
- Student: "**I got more results than probes submitted...**"
  - old/stale results arriving from an earlier experiment
  - when a monitor "wakes up" all the previously queued probe requests are executed

- Student: "**Ark is b0rken (again)...**"
  - submitting millions of requests to monitor X
  - program wrapping tod-client died/crashed
  - requests at monitor X still pending
  - new requests seemingly unresponsive (monitor still busy with old/stale requests that will never be fetched)
- Student: "**I got more results than probes submitted...**"
  - old/stale results arriving from an earlier experiment
  - when a monitor "wakes up" all the previously queued probe requests are executed
- Student: "**How do I make `tod-client` do X...**"
  - `tod-client` written in Ruby
  - limits accessibility for those willing to customize/tweak/deploy

Outline

- Designed to finesse the submission of probe-requests to ToD
- Removes the need for every user to handle common scenarios in their own code
- Changing subset of down or slow Ark monitors
  - ArkQueue sorts and queues user probe requests by VP
  - runs separate instance of `tod-client` for each VP
  - tracks VP response time and stops submitting if unresponsive
  - reports unresponsive VPs to user for future reference
- Submitting "too many" requests may break Ark
  - ArkQueue maintains "just enough" requests outstanding

- Results arrive asynchronously
  - ArkQueue assigns unique session- and probe-IDs
  - maps responses to requests and returns both
  - uses callback hook to process replies on arrival
- Results stay in system even after `tod-client` terminates
  - ArkQueue cleans up outstanding requests using `tod-debug`
- ArkQueue facilitate intelligent probing patterns, where future probes depend of feedback from earlier probes
- Along with ArkQueue, python module is provided for parsing `tod-client` output

ArkQueue

```
from arkqueue import ArkQueue

def submit(x):
    sys.stdout.write('+')

def finish(out, request):
    sys.stdout.write('^')

ark = ArkQueue(monitorfile="monitors.yaml",
        sessionid="ArkQueueSample", yaml=True, verbose=False,
        submit_hook=submit, finish_hook=finish, idle_hook=None,
        concurrency=10, timeout=60, monitor_blacklist=list())
vps = ark.getMonitors()
targets = ['128.61.2.1', '130.207.244.244', '2607:f8b0:4005:802:
ark.start()
for i in range(0, 100):
        vp = vps[i % len(vps)]
        target = targets[i % len(targets)]
        ark.addProbe(targets=[vp + ' ' + target], priority=3)
```

ArkQueue

```
[rbeverly@bob ~/research/direct/ark/arkqueue]$ ./arkqueue_sample.py
[ arkqueue_sample.py ]Interacting with Ark ToD, will maintain 1 traces
in flight per vantage point.
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
++++++++++++++++++^++++^+++++^^+++++^++++^+++^^^^^^^^^^^^^^^^^^^^^^^^^
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
^^^^^^^
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!.!..!..!!!.!!!..!.!.!!!!!!!!!.!!!!!.!!!!!!!!..!!.!!.!!!!!!!!!.!!!!
Exiting Ark
2015-03-31 04:57:15,614 - [ArkQueue]:INFO: Thread asked to exit.
2015-03-31 04:57:16,288 - [ArkQueue]:INFO: Waiting for VP threads to ex
2015-03-31 04:57:49,941 - [ArkQueue]:INFO: Waiting for callbacks to fin
------------
ArkQueue Probing Summary:
Total number of probes submitted: 100
Number of probes completed: 86
Average probe completion time: 7.28698908488 s
Number of probes not completed: 14
Number of Ark vantage points used: 72
```

Outline

Scamper is CAIDA's software daemon that runs on each Ark monitor to create probes and collect data

- Why do we care?
  - Scamper can do much more than the probe type allowed by `tod-client`
  - may have vantage points outside of Ark on which to run Scamper
- How do we use it?
  - run Scamper interactively, or as a daemon
  - CAIDA-provided `sc_attach` utility submits commands to a Scamper daemon, and writes the results to a warts file
  - other stand-alone utilities and a c-library are available from CAIDA to parse the warts files into human-readable formats

Most of our (and our students) probing and analysis is performed using Python programs

`sc_attach.py`

Native Python module for issuing commands to a Scamper instance. Receives results and writes to a binary warts file.

`sc_wartsdump.py / sc_analysis_dump.py`

Native Python module for parsing a binary warts file to text for further analysis

- `sc_wartsdump` also used for parsing warts data collected by CAIDA
- ArkQueue and Scamper Tools publicly available to facilitate wider adoption and use of Ark/ToD

Outline

Ark Wishlist (not in order):

- Expose list of available monitors (and tell user of new monitors put into production)
- Warts output (ToD produces tab delimited partial output)
- Ability to clear large numbers of tuples in a timely manner (takes very long time)
- Full control over scamper options/flags (can only use what Ark exposes)
- Fix marinda memory leak
- Visibility into outstanding request tuples
- Visibility into individual monitor queue/status

Thanks!

- http://www.cmand.org/direct
- https://github.com/cmand/arkqueue
- https://github.com/cmand/scamper