Update on Traffic Replay

Susan Hinrichs

shinrich@verizonmedia.com, shinrich@apache.org

Verizon Media/Yahoo

Committer/PMC, Apache Traffic Server

ATS Summit, October 2019

Goal of Talk

Present work performed with traffic replay and Apache Traffic Server

- Both historical efforts and more recent work this year

Work performed with my colleagues at Yahoo/Oath/Verizon Media and members of the Apache Traffic Server community

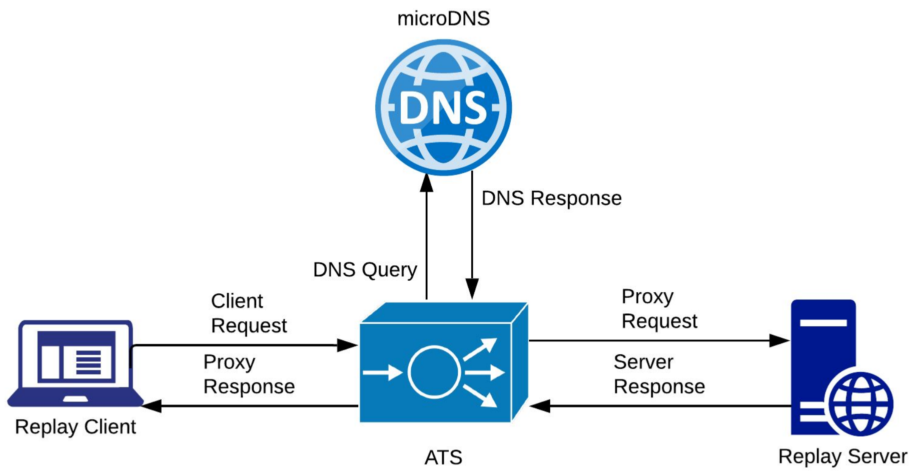
Goals of Traffic Replay

Simulate clients and server.

Replay traffic against an ATS process

Two major targets

- **Correctness testing**
- **Performance testing**



verizon✓ media

Differing requirements for replay performance and correctness testing

For correctness

- Speed is less of a concern
- Replay data is very carefully selected
- Deep header, timing, maybe content verification is important

For performance

- Speed is very important. Need production speeds for more
- Replay data can just be sampled data. Too much to hand curate
- Some verification is needed, but need to maintain speed

History of Traffic Replay Efforts at Yahoo/Verizon Media

Eric Schwartz added “wire trace” feature to dump TLS session details (2015)

Dan Xu (Dantern) wrote scripts to post process write trace data dumped in error.log to create replay files (2015-16)

He also wrote original versions of microserver (replay server), microdns, and a replay client in python

- Microserver has evolved to be the standard test server in autest
- Microdns is also used in some autest scenarios

More history

Zeyuan Yu created traffic dump as a means to statistically sample traffic patterns from production

- Captures only headers and content length. Not actual content.
- Alan Carroll worked with him to create a JSON dump file format
- Traffic dump is open source experimental plugin

Yet More History

Original python replay and microserver scripts very difficult to get working near production speeds

Alan Carroll wrote a C++ based http_replay project with a replay_client and replay_server

- Includes some basic header verification
- Cannot do the observer pattern in the C++ version
    - May not need observer if the replay_server can also verify headers

Replay File Format

JSON file that describes a list of timestamped sessions.

Each session contains a list of transactions. List of protocols associated with session: e.g. tls, http2

Each transaction has 4 header sections: client request, proxy request, server response, and proxy response.

Traffic Replay for Performance

Why make yet another traffic generator?

- **Standard traffic generators like wrk2 make very uniform traffic streams. Either all requests are over a small number of connections or each request is over a unique connection.**
- **There exist more sophisticated traffic generation tools like Tsung. Hard to get them to run at scale. Tedious to define traffic flows.**
- **Hardware generators like IXIA's ixload. Again tedious to create realistic traffic**

For all of these cases still need to stand up test origins to complete the Proxy test scenario.

Traffic_dump

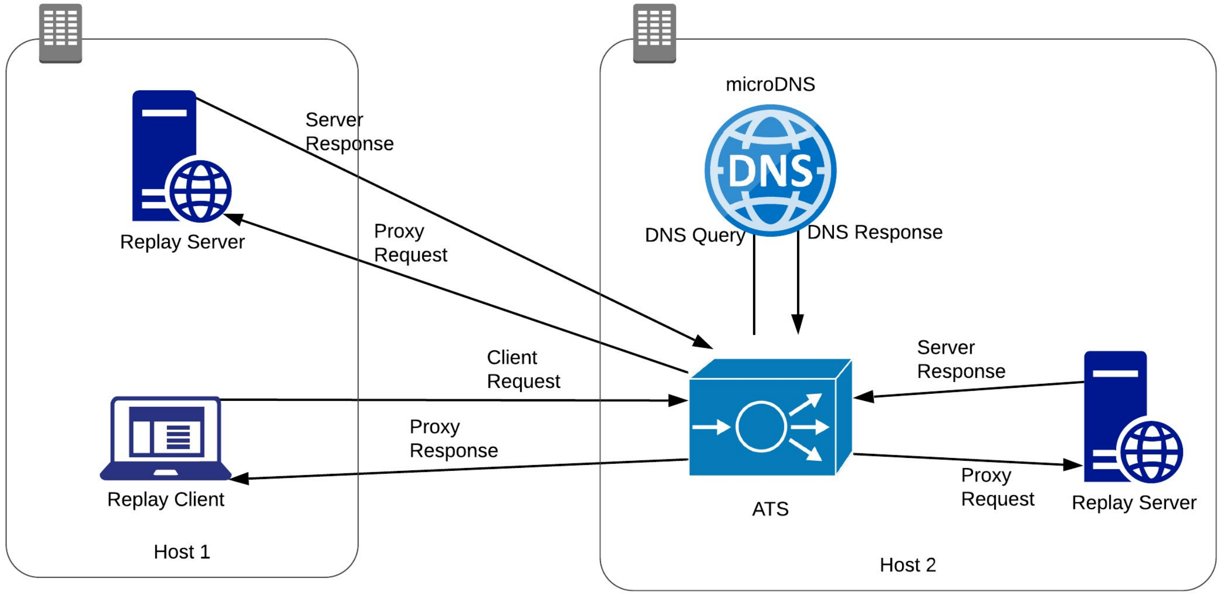
Traffic_dump lets us capture traffic patterns realistic for our environment.

- Plugin has controls to limit total disk used
- Can adjust the percentage of sessions captured
- Once a session is selected, all transactions on that session are captured

Post capture sanitizing

- Remove sensitive data
- Synthesize server responses for cache hits

Replay Performance Setup



verizon✓ media

Issues with Performance testing

Loading unique traffic descriptions takes a long time

- Even not capturing content, loading and parsing the traffic_dump files can take as long or longer than performing the replay
- Looking at a preprocessing phase to translate the data into a denser easier to parse format

Running out of ephemeral ports is a easy pitfall in a lab testing environment.

- Need to add more addresses to listen from and request from

Controlling the Replay

replay-client takes options to adjust the rate of replay

- Run as fast as possible
- Run at capture rate (not very interesting if you were sampling)
- Run at specified rate, e.g. 2000 rps or 25000 rps
    - Scales intervals based on the session timestamps to hit the target RPS
- Run through the data set multiple times

Preliminary performance tests

Set up in lab with two performance grade servers connected via 40Gbps network.

Working with 143K captured transactions

Production config but no plugins

TLS as indicated in captured data

No HTTP/2 support

Running internal 7.1.x ATS

RPS	%CPU ATS	1 iteration run time (s)
2000	7.5	71.8
6000	15	23.9
10000	25	14.4
16000	45.3	9.06
20000	61.8	7.3

Traffic Replay for Testing

The original microserver and microdns have been incorporated into our autests

The replay file is there but generally pretty minimal

```
server = Test.MakeOriginServer("server", ssl=True)
request_header = {"headers": "GET / HTTP/1.1\r\nHost: www.example.com\r\n\r\n", "timestamp":
"1469733493.993", "body": ""}
# desired response form the origin server
response_header = {"headers": "HTTP/1.1 200 OK\r\nConnection: close\r\n\r\n", "timestamp":
"1469733493.993", "body": ""}
server.addResponse("sessionlog.json", request_header, response_header)
```

The replay client got lost somewhere along the way. Most autests using curl instead

Traffic replay for creating regression tests

Alan has been pushing traffic replay for setting up regression test scenarios

File to exercise regex issue.

https://github.com/apache/trafficserver/blob/master/tests/gold_tests/pluginTest/regex_remap/replay/yts-2819.replay.json

Problem regex_remap Rule

```
# regex_remap configuration
^/alpha/bravo/[?]((?!action=(newsfeed|calendar|contacts|notepad)).)*$
http://example.one @status=301
```

verizon media

Augmenting autest to leverage traffic replay

Corwin Carroll has created autest extensions that given a replay file will launch replay_server, traffic_server and create a test run that will execute replay_client

```
ts,dns,replay_server, tr = Test.ReplaySetUp(data_dir)

# Commands to setup the configuration for traffic server

tr.run()
```

This allows for the creation of very minimal autest files. The primary portion of the test is in the replay data file.

https://github.com/corwin-carroll/trafficserver/commit/5b634ad3bb85e109c5e60e887353f66954fb7486

Replay header verification options

Will Wendorf (Will-tern?) added header verification operators this summer

Can add header constraints globally, per session, or per transaction

For each rule, specify header, action and value

- Action can be equality, presence, or absence
- Foo value must equal 10
- Foo must be present
- Foo must not appear

Next Steps

Adding HTTP/2 support to http_replay

Internally use this framework for validating 9.0.x releases

Open source http_replay

Contribute more replay based tests

- How best to provide replay binaries? bintray?

Q&A

shinrich@verizonmedia.com
shinrich@apache.org