Yarrp'ing the Internet: Randomized High-Speed Active Topology Discovery

Robert Beverly

Naval Postgraduate School

November 16, 2016

ACM Internet Measurement Conference 2016

Outline

1. Background
2. Methodology
3. Results
4. Future

Active Topology Probing

- **Years** (and years) of prior work on Internet-scale topology probing
- Current production systems take **days** from 100's of vantage points to gather even coarse-grained network map
- Topology "snapshots" are a misnomer! – network can **change** during probing

It's 2016:

- Why can't we traceroute to *every* IPv4 destination *quickly*?
- i.e., $O(minutes)$?
- (The ZMap[a] and Masscan[b] folks can do it – why can't we?)

[a]Durumeric et al., 2013
[b]Graham, 2013

Active Topology Probing

- **Years** (and years) of prior work on Internet-scale topology probing
- Current production systems take **days** from 100's of vantage points to gather even coarse-grained network map
- Topology "snapshots" are a misnomer! – network can **change** during probing

It's 2016:

- Why can't we traceroute to *every* IPv4 destination *quickly*?
- i.e., $O(minutes)$?
- (The ZMap[a] and Masscan[b] folks can do it – why can't we?)

[a] Durumeric et al., 2013
[b] Graham, 2013

Existing traceroute-style approaches:

- Maintain **state** over outstanding probes (identifier, origination time)
- Are path-**sequential**, probing all hops along the path.

Implications:

- **Concentrates load:** along paths, links, routers (potentially triggering rate-limiting or IDS alarms)
- Production systems probe **slowly**

Existing traceroute-style approaches:

- Maintain **state** over outstanding probes (identifier, origination time)
- Are path-**sequential**, probing all hops along the path.

Implications:

- **Concentrates load:** along paths, links, routers (potentially triggering rate-limiting or IDS alarms)
- Production systems probe **slowly**

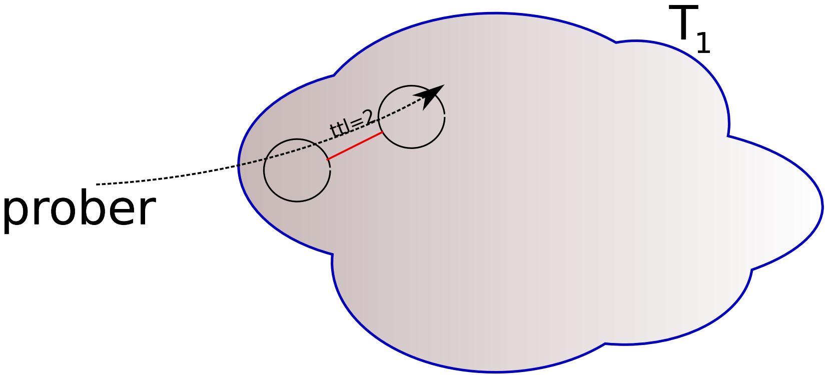
Outline

Yarrp

Yarrp: "Yelling at Random Routers Progressively"

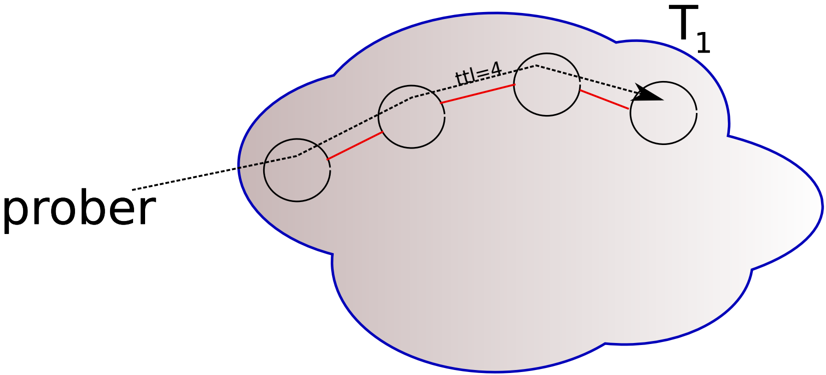
- Uses a block cipher to **randomly permute** the $\langle IP, TTL \rangle$ domain
- Is **stateless**, recovering necessary information from replies
- Permits **fast** Internet-scale active topology probing (even from a single vantage point)

Traditional Traceroute



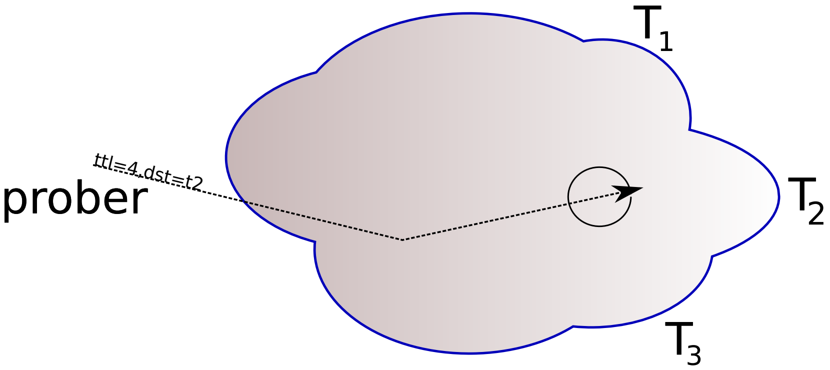
Traditional traceroute sends probes with incrementing TTL toward destination T_1

Traditional Traceroute



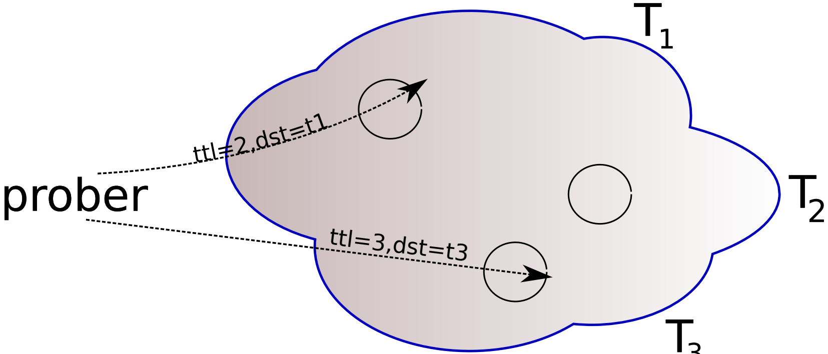
... continuing until finished with T_1 (reach destination or gap limit). Prober must maintain state, while traffic is concentrated on *prober* $\rightsquigarrow$ T_1 path

Yarrp



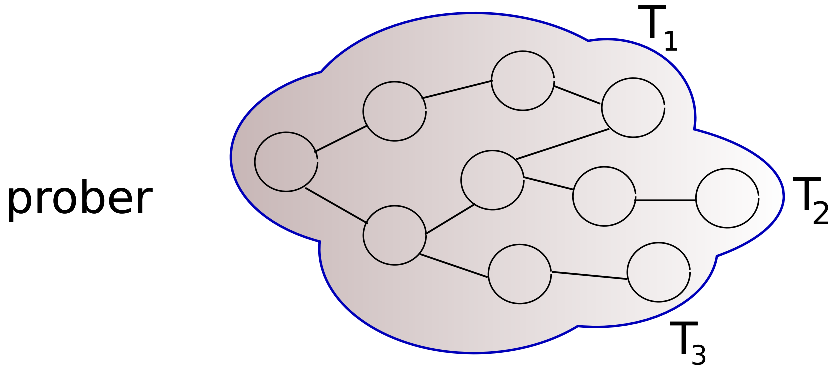
In contrast, Yarrp iterates through randomly permuted $< Target, TTL >$ pairs

Yarrp



In contrast, Yarrp iterates through randomly permuted $< Target, TTL >$ pairs

Yarrp

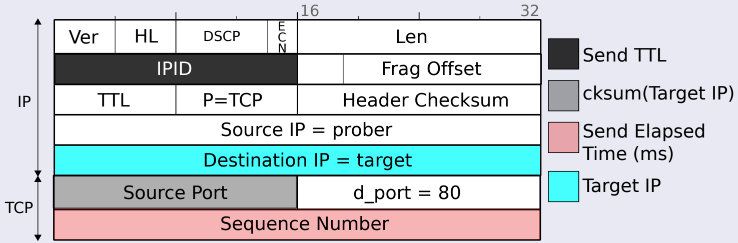


Finally, stitch together topology. Requires state and computation, but decoupled (off-line after probing completes).

Challenges:

1. Randomize probing order
2. Map responses to probe's destination, TTL, and xmit time
3. Accommodate load-balancing
4. Avoid over-probing a path (when to stop)
5. Reconstructing topology from un-ordered responses

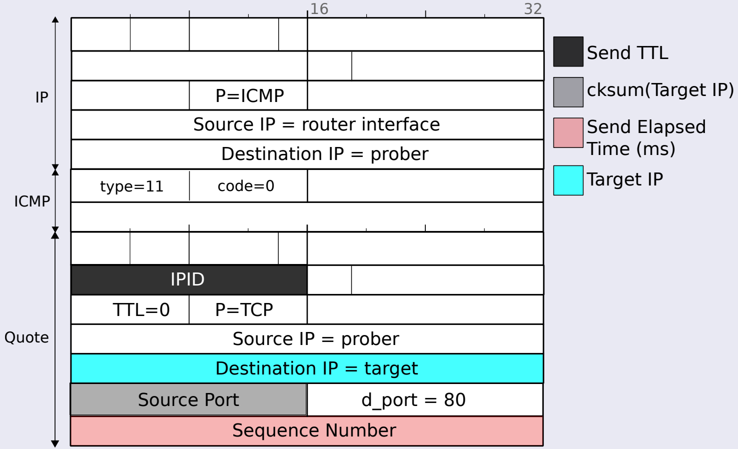
Yarrp Probes – Encoding State



- IPID = Probe's TTL
- TCP Seq No = Probe send time (elapsed ms)
- TCP Source Port = cksum(Target IP destination)[a]
- Per-flow load balancing fields remain constant (ala Paris)
- (Assume routers quote only 28B of expired packet)

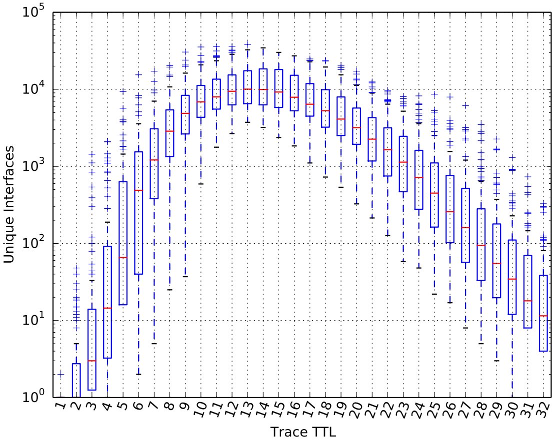
[a] Malone PAM 2007: ≈2% of quotations contained modified destination IP

Recovering State



ICMP TTL exceeded replies permit recovery of: target probed, originating TTL (hop), and responding router interface at that hop.

Distribution of unique interfaces discovered vs. TTL for all Ark monitors, one CAIDA routed /24 probing cycle



- Problem: when to stop?
- Little discoverable topology past TTL=32
- $\Rightarrow$ limit $< IP, TTL >$ search space to $TTL \leq 32$

Decoupling Probing from Reconstruction

Receive thread runs independently

- Recovers state and writes responses
- Because probing is randomized, replies are un-ordered:

```
# yarrp $Id: yarrp.cpp 40 2016-01-02 18:54:39Z rbeverly $
# Started: Tue May 10 12:52:41 2016
# Source: 18.26.2.84, Count: 0 Rate: 4000
# Rand: 1 Nbrh: 0 Entire: 0 BGP: bgptable.20160510.txt.gz TraceType: 3
# Input IPlist: /home/rbeverly/c004710.san-us.targets MaxTTL: 16
# target, sec, usec, type, code, ttl, hop, rtt, ipid, psize, rsize, rttl, rtos
109.112.178.108, 1462899605, 97182, 11, 0, 8, 198.71.47.61, 22, 0, 40, 56, 248, 0
75.227.91.50, 1462899605, 97299, 11, 0, 9, 4.68.110.82, 5, 0, 40, 56, 246, 0
150.243.54.100, 1462899605, 97418, 11, 0, 6, 18.192.7.2, 1, 2310, 40, 96, 250, 0
179.130.181.73, 1462899605, 98230, 11, 0, 14, 200.220.224.253, 206, 10160, 40, 56, 235, 72
42.97.123.149, 1462899605, 99366, 11, 0, 11, 64.57.20.146, 54, 0, 40, 56, 245, 0
198.48.67.42, 1462899605, 100550, 11, 0, 1, 18.26.0.2, 10, 55674, 40, 56, 255, 0
104.3.115.120, 1462899605, 100666, 11, 0, 10, 12.122.130.170, 50, 25157, 40, 168, 240, 0
84.106.41.175, 1462899605, 100953, 11, 0, 13, 84.116.195.246, 133, 48736, 40, 56, 241, 0
76.216.172.133, 1462899605, 101268, 11, 0, 15, 12.122.30.30, 83, 23223, 40, 172, 239, 0
74.150.100.227, 1462899605, 102383, 11, 0, 10, 68.85.184.198, 8, 10, 40, 56, 246, 192
108.76.185.84, 1462899605, 102395, 11, 0, 14, 12.122.30.25, 78, 28971, 40, 172, 242, 0
155.198.102.65, 1462899605, 103470, 11, 0, 11, 62.40.98.76, 83, 0, 40, 56, 245, 0
```

Decoupling Probing from Reconstruction

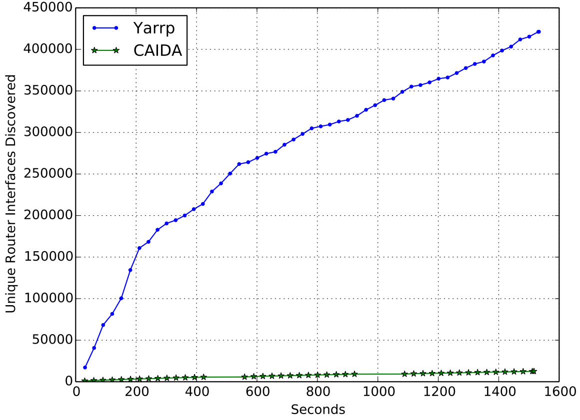
Topology Reconstruction

- `yrp2warts.py`: assembles un-ordered Yarrp responses into series of binary warts-formatted traces

```
traceroute from 18.26.2.84 to 190.144.172.20
 1  18.26.0.2  1.000 ms
 2  128.30.0.245  1.000 ms
 3  128.30.13.5  1.000 ms
 4  18.4.7.1  4.000 ms
 5  18.192.2.1  1.000 ms
 6  18.192.7.2  1.000 ms
 7  207.210.143.109  1.000 ms
 8  192.5.89.21  1.000 ms
 9  192.5.89.222  6.000 ms
10  198.71.46.174  24.000 ms
11  200.0.207.9  36.000 ms
12  200.0.204.6  84.000 ms
13  200.0.204.182  147.000 ms
```

Outline

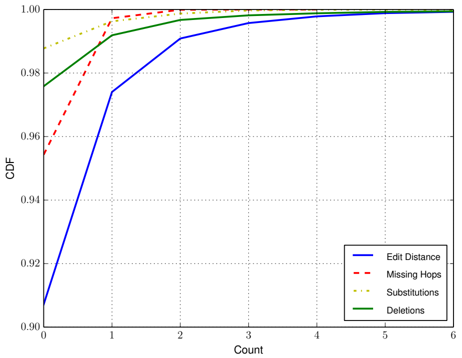
Yarrp vs. CAIDA



- Well-provisioned university vantage
- Yarrp on KVM (1 core @ 2.27GHz) at 100*kpps*, 52% CPU
- Yarrp: ≈ 280 unique router interfaces / sec
- Ark: ≈ 8 unique router interfaces / sec

Short-Lived Dynamics

Application: Rapid Snapshots



- 67k targets, three Yarrp snapshots in succession
- Examine edit distance between S_1 and S_2
- 91% of paths identical, 6% have single hop difference
- (4% are 1 hop diffs due to missing response, 1% substitutions)

Short-Lived Dynamics

Example, probe toward ASN 262316

```
... 18.192.9.2 4.53.48.97 4.69.144.80 4.69.144.80 4.26.0.166 201.48.50.161 201.48.50.154 201.48.44
... 18.192.9.2 207.210.142.229 198.71.47.57 * 67.16.148.6 201.48.50.161 187.115.214.189 187.115.21
... 18.192.9.2 38.104.186.185 154.54.30.41 154.54.47.30 154.54.11.110 64.210.21.110 213.155.131.23
```

Inferred AS_PATH

```
S1: 3 3356 16735 28303
S2: 3 10578 11164 3549 16735 18881 4.172
S3: 3 174 3549 1299 25933 16735
```

- Confirmed BGP churn visible at routeviews
- Vantage point AS using different egresses due to churn
- These dynamics would be *invisible* to existing active topology probing systems

Outline

Yarrp Enhancements

- **UDP Probing:**
  - TCP probing is blocked more often and triggers more alerts
  - Encode timestamp into the length and checksum; create a payload to make checksum correct
- **ICMP Probing:**
  - Encode timestamp into identifier and sequence number; create payload s.t. each packet has same checksum
- **IPv6 Probing:**
  - Different IPv6 headers imply different encoding
  - But, full quotes in ICMP6 enable more flexibility

Distributed Probing

- Use crypto permutation to divide probing among many monitors
- Minimal communication overhead, distribute *key*, size of domain $|D|$, number of monitors n, and monitor id v. Then:

for $i \in |D|$ **do**
  $(ip, ttl) = E_{key}(i)$
  if $ip\%(n-1) == v$ **then**
    probe(ip, ttl)

- Speed scales linearly with n
- Given 100kpps and $n = 128$, traceroute to every routed IPv4 address in < 1 hour

Yarrp'ing the Internet

- New technique for rapid active topology discovery
- Redefine notion of a topology "snapshot"
- Demonstrate ability to detect short-lived dynamics
- Publicly available implementation

Thanks! – Questions?

https://www.cmand.org/yarrp

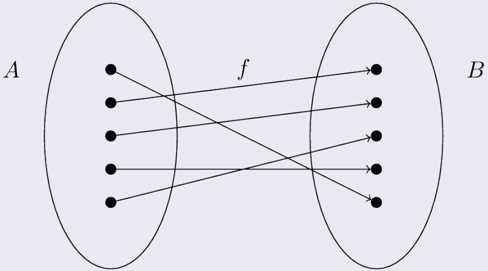
Implementation

Yarrp Implementation

- C++ ~2,500 SLOC
- Independent send and receive threads
  - Send thread uses raw sockets
  - Receive thread uses libpcap
- Portable to variety of UNIX-like platforms
- Publicly available: `https://www.cmand.org/yarrp`

Pseudo-random Probing Order

- We use RC5 block cipher with 32-bit block size
- Encrypt $i = 0, \ldots, 2^{32} - 1$ with key k to obtain /24's and TTLs:
  - $C_i = RC5_k(i)$
  - /24 = $C_i[0:23]$
  - TTL = $C_i[24:31]$
  - Least-significant octet: $f(C_i[0:23])$



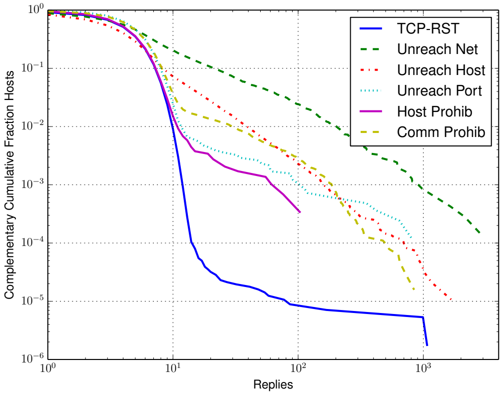
Optimizations

- Base Yarrp requires no state
- (Must reconstruct traces, but that's an offline local process)
- If we're willing to maintain some space, we can optimize: Time Memory Trade Off
  1. Probe only routed destinations (radix trie BGP RIB)
  2. Avoiding repeated re-discovery of prober's local neighborhood (state over small number of interfaces near prober)
- (See paper for full details)

Ethical Concerns

- High-speed probing increases chance traffic perceived as abusive
- Yarrp sends TCP ACK probes (less abusive than ZMap's SYNs)
- Random probing order avoids overloading networks
- Stateless nature implies multiple probes with different TTLs may reach a single destination
- We follow good "Internet citizenship" guidelines:
    - Coordinated with local network admins
    - Informative web page at address of prober
    - DNS PTR record indicates research nature
    - Provide links to opt-out
- In our $\leq$ 60 min Yarrp runs, we received no abuse reports or opt-outs

Non-TTL Exceeded Replies



- Yarrp's TCP probing elicits a variety of responses
- ∼95K ICMP Host Unreach, ∼63K ICMP Communication Prohib
- Received ∼1.2M TCP RST packets
- But, 99.1% of hosts sending a RST sent $\leq$ 10
- (3 IPs in Wanadoo send majority)