NAVAL
POSTGRADUATE
SCHOOL

Sundials in the Shade

An Internet-Wide Perspective on ICMP Timestamps

Erik Rye (CMAND)
Robert Beverly (NPS)

Passive and Active Measurement Conference
March 28, 2019

CMAND

Background

- ICMP nearly 40 years old! (RFC792)
- 27 ICMP types defined, 13 formally deprecated
- Communicate diagnostic and error information, e.g.:
  - Echo request/reply
  - Time exceeded
  - Destination/Port/Network Unreachable
- ***ICMP Timestamps***:
  - Type 13 (timestamp) and 14 (timestamp reply)

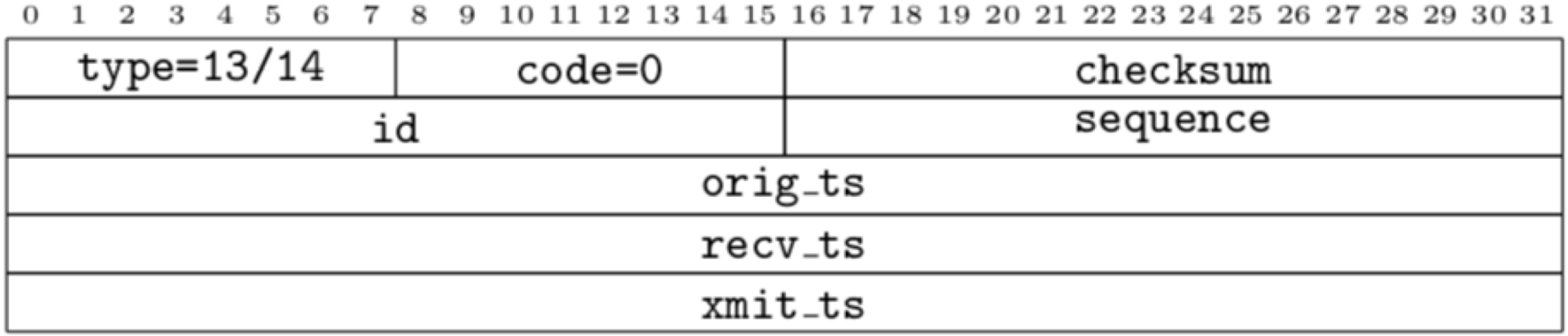
ICMP Timestamps

- Forgotten, but not gone (or deprecated)
- IPv4 only
- Intended uses:
  - Time synchronization (superseded by NTP)
  - One-way delay measurements
- No legitimate use in 2019 but, since not deprecated, devices must (try to) implement!

Related Work

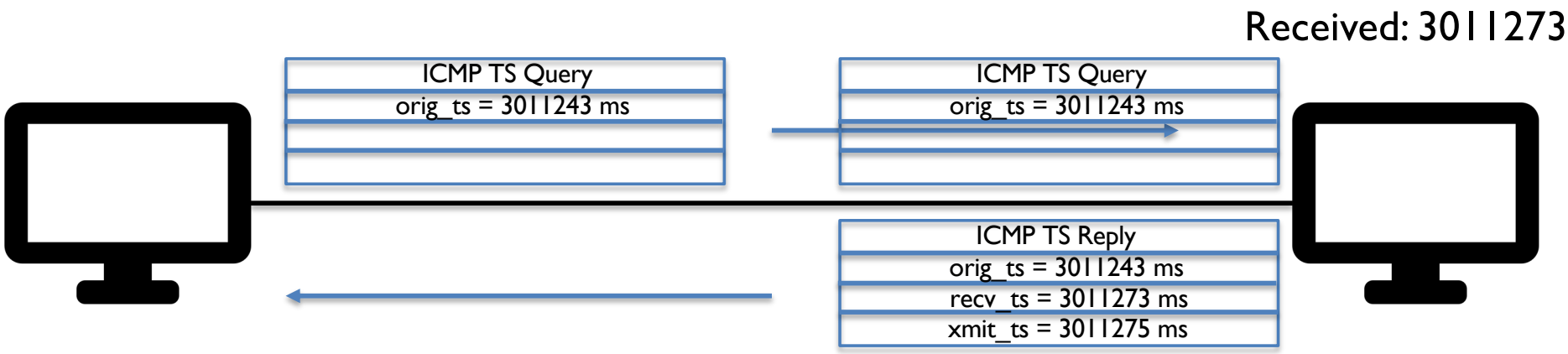
- Anagnostakis, *et. al* (2003), Mahajan, *et. al* (2003) – use ICMP timestamps to measure one-way delays and diagnose faults
- Kohno, *et. al* (2005) – use clock skew from TCP/ICMP timestamps to fingerprint devices
  - We focus on response behavior, rather than clock skew measurements
- Bucholtz, *et. al* (2007) – investigate clock behavior of 8,000 web servers
  - We probe 2,000x more hosts > 10 years later, not just web servers, and show novel fingerprinting/geo applications

RFC792



- id/sequence used to correlate replies with requests
- Three 32-bit timestamps encode the number of ms since midnight UTC
  - Originate – time the sender *sends* request
  - Receive – time the receiver *receives* request
  - Transmit – time the receiver *transmits* reply
- If UTC-reference unavailable, hosts can set MSB and use any reference they wish

ICMP Timestamps 101 – RFC 792



*Assuming both machines have correct clock and are RFC792 compliant

Research Questions

1. **How many** devices on the Internet respond to ICMP timestamp requests?
2. How do devices **implement** ICMP timestamps?
3. How can we leverage ICMP timestamps for novel / **new measurement insights**?

Sundial Methodology

- Probe ISI hitlist – 1 IPv4 address per routable /24 (~16 million addresses)
- Vantage points in Boston and San Diego on university networks
- Vantage points are NTP synchronized (stratum 2 or better)
- To each IP, send four different probes to *elicit variety of response behaviors*
  - *Standard* – set origin timestamp correctly, 0 recv/xmit
  - *Bad clock* – intentionally set origin timestamp incorrectly
  - *Bad checksum* – intentionally set chksum incorrectly
  - *Duplicate timestamp* – set origin timestamp correctly, copy value into recv/xmit fields in request
- Id/seq = hash of dest IP + timestamp to detect middlebox tampering

Response Taxonomy

		Request	Response		
Num	Class	cksum	orig_ts	recv_ts	xmit_ts
1	Normal	valid	-	$\neq$ xmit_ts, $\neq 0$	$\neq 0$
2	Lazy	valid	-	= xmit_ts	$\neq 0$
3	Checksum-Lazy	bad	-	-	-
4	Stuck	valid	-	const	const
5	Constant 0	valid	-	0	0
6	Constant 1	valid	-	1	1
7	Constant LE 1	valid	-	$htonl(1)$	$htonl(1)$
8	Reflection	valid	-	$\texttt{request}_{\texttt{recv\_ts}}$	$\texttt{request}_{\texttt{xmit\_ts}}$
9	Non-UTC	valid	-	$> 2^{31} - 1$	$> 2^{31} - 1$
10	Timezone	valid	-	$\lvert\texttt{recv\_ts} - \texttt{orig\_ts}\rvert \% \left(3.6 \times 10^6\right) < 200\ ms$	-
11	Little Endian	valid	-	$\lvert htonl(\texttt{recv\_ts}) - \texttt{orig\_ts}\rvert < 200\ ms$	-
12	Linux *htons*() Bug	valid	-	$\%2^{16} = 0$	$\%2^{16} = 0$
13	Unknown	valid	-	-	-

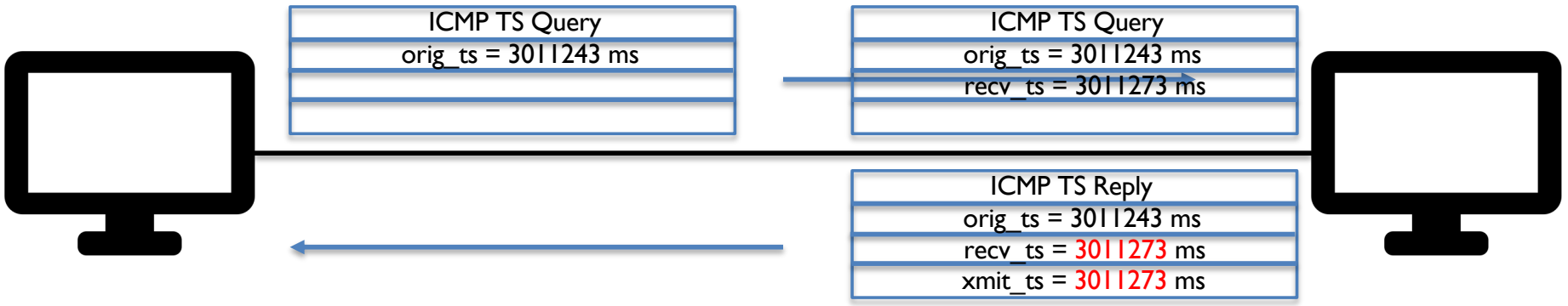
Reflectors

Stuck, constant values!

Endianness

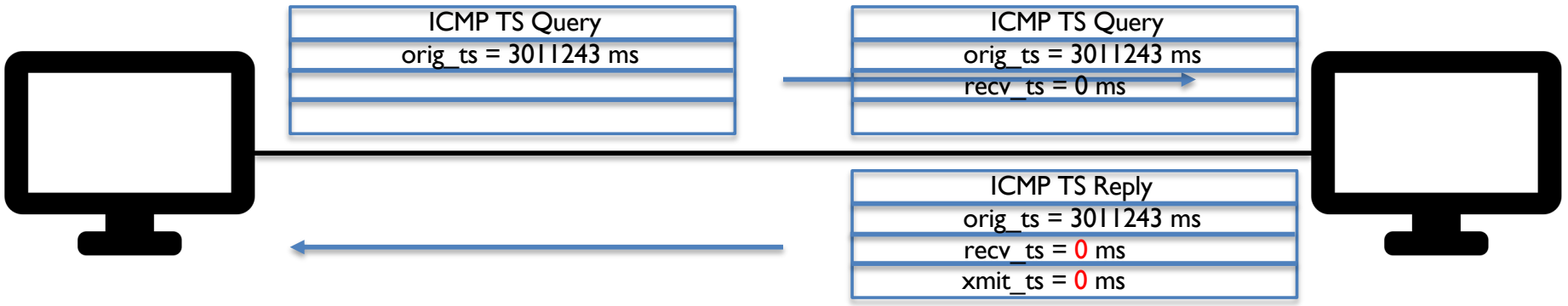
Wide variety of behaviors!

Example – Lazy Replies



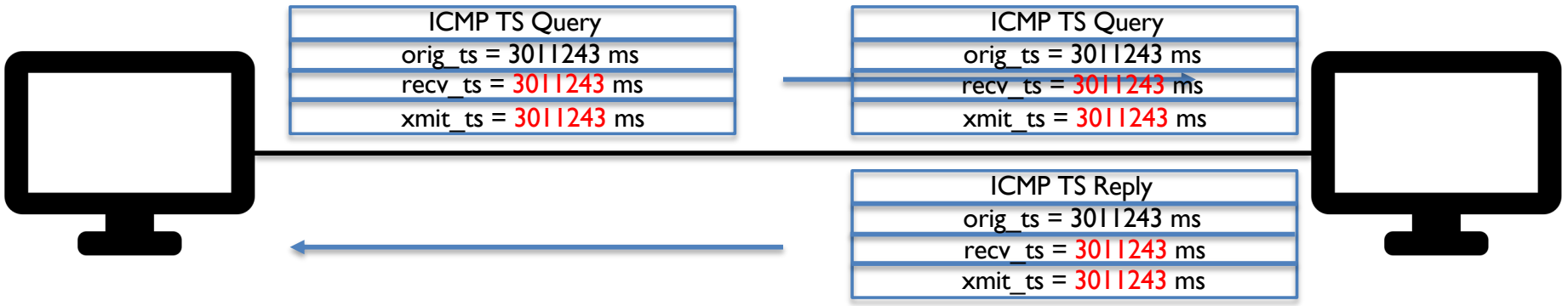
- Results from receive timestamp being copied into transmit timestamp (only one time-getting function call) **OR**
- Receive and transmit timestamps taken within same ms (two time-getting function calls)

Pathologies – Constant 0 Replies



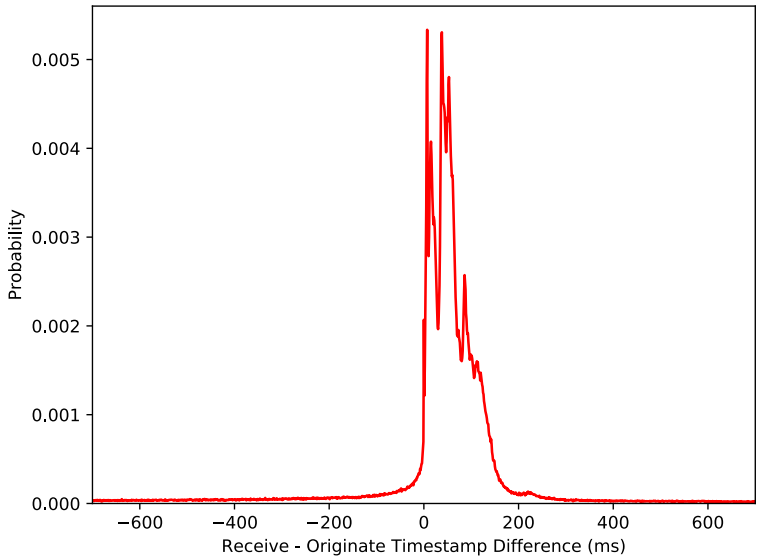
- About half of hosts exhibiting the constant 0 behavior showed signs of middlebox tampering using *tracebox*, suggesting this is possibly an OS **and** middlebox behavior

Pathologies – Reflection Replies



- Send both the **standard** and **duplicate timestamp** requests
- Receive/transmit timestamps of reply match receive/transmit timestamps of request
- Must send both requests to distinguish from constant 0

Correctness



- Empirically derive correctness bound from timestamp replies
- Consider receive timestamps within +/- 200ms of originate timestamp as “correct”

Results

Lazy behavior dominates

Category	Boston	Both	San Diego	Category	Boston	Both	San Diego
Normal	40,491	19,819	40,363	Stuck	855	849	873
Lazy	2,111,344	1,899,297	2,112,386	Constant 0	547	546	555
Checksum-Lazy	28,074	23,365	28,805	Constant 1	200	199	207
Non-UTC	249,454	211,755	249,932	Constant LE 1	22	19	23
Reflection	2,325	2,304	2,364	`htons()` Bug	1,499	665	1,536
Correct	850,787	803,314	850,133	Timezone	33,317	23,464	33,762
Correct LE	11,127	5,214	11,290	Unknown	38,495	11,865	32,956
Correct - MSB	1,048	586	973				
Total					2,194,180	1,934,172	2,189,524

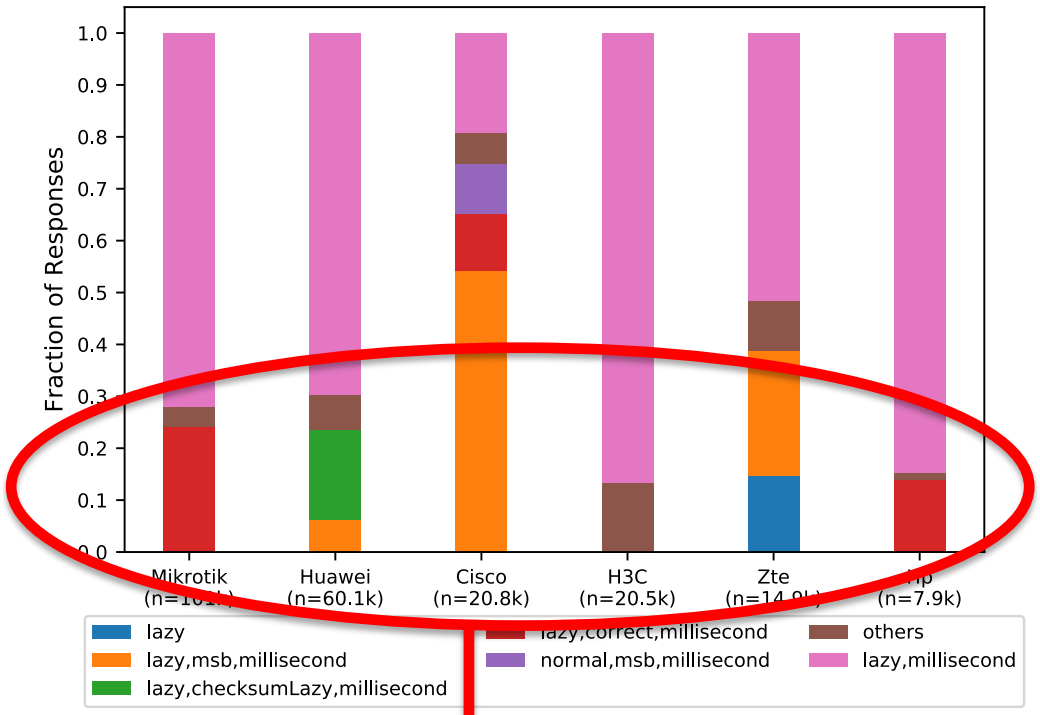
Majority of timestamps outside 200ms of correct!

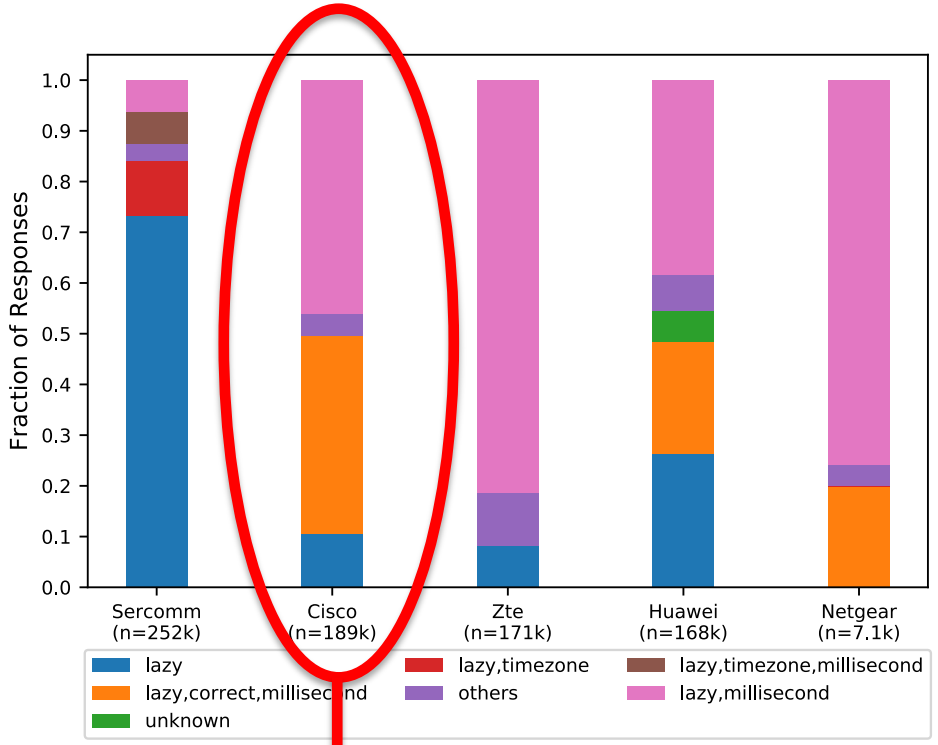
~2.2M responses!

Telnet/CWMP Ground Truth

- Characterize behavior of well-known OS (see paper)
  - But, lots of behavior of unknown origin remained
- Many application-layer protocols leak device manufacturer:
  - Telnet banners
  - CPE WAN Management Protocol (CWMP) GET responses
- Leverage IPv4-wide Telnet and CWMP scans from `scans.io`
- Parse manufacturer strings, IP addresses from data
- Use *sundial* to classify behavior by manufacturer

Telnet/CWMP Ground Truth





And within manufacturers

Wide variety of behaviors between manufacturers

Application 1: Operating System Fingerprinting

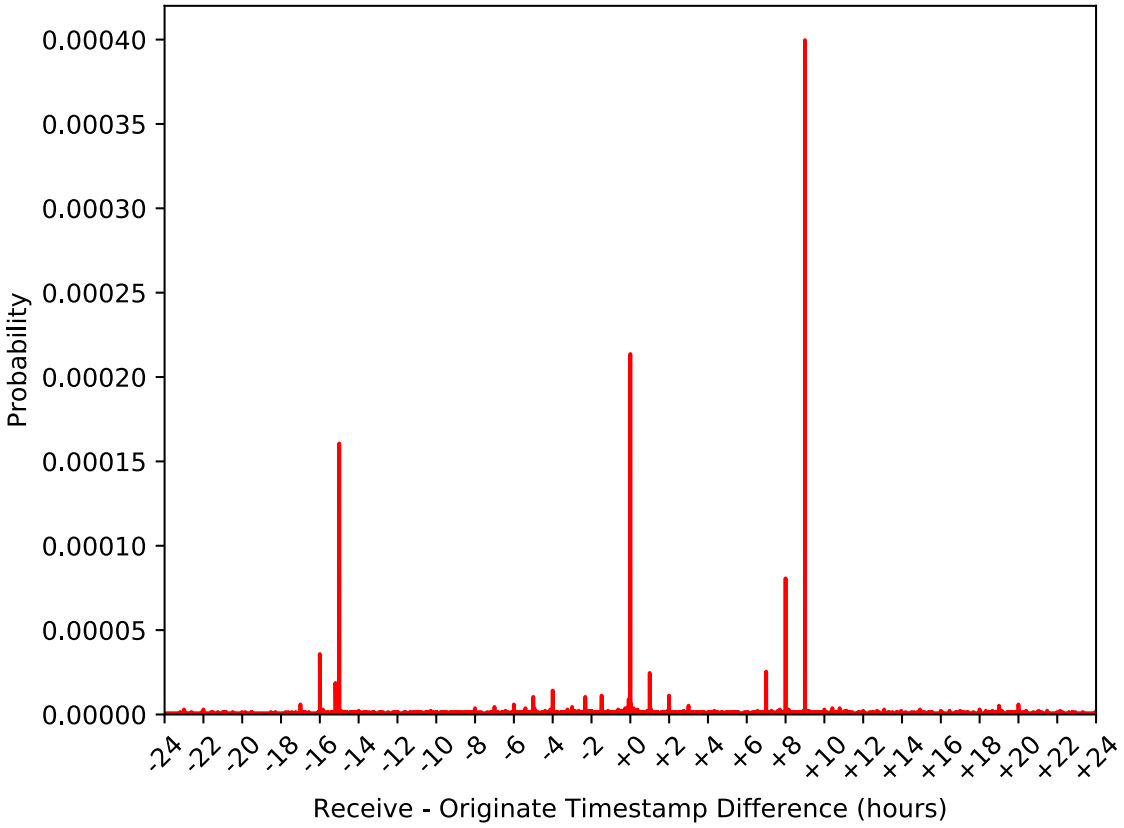
```
icp->icmp_type = ICMP_TSTAMPREPLY;
icp->icmp_rtime = iptime();
icp->icmp_ttime = icp->icmp_rtime;      /* bogus, do later! */
goto reflect;
```

- Linux/*BSD & derivatives
  - Copy recv timestamp into xmit – always lazy
- Windows
  - Off when Windows Firewall enabled, lazy little-endian when disabled
- Cisco IOS
  - MSB set (non-UTC) when NTP disabled, lazy/correct when configured

Application 2: Linux htons() Bug

- Linux kernel v3.18 updated with Y2038-compliant timestamps…
- …but kernel function **inet_current_timestamp()** erroneously returns htons() rather than htonl()
- So, 3.18 Linux returns timestamps with lower two bytes zeroed in network order
- Error made its way into Android kernel 3.18 as well
- Fixed in subsequent releases
- Allows for fine-grained OS fingerprinting!

Application 3: Coarse-Grained Geo



- Spikes at hour-interval differences from sender's correct UTC-reference
- >90% agreement of timestamp-inferred timezone w/MaxMind GeoLite-2
- Large number of devices in +9 timezone (Japan, South Korea), unclear why

Conclusions

- Despite no modern use, ICMP timestamps widely supported
- Because no applications depend on them, implementation behavior varies wildly
  - Allows for OS fingerprinting, coarse geolocation
- We developed a tool, *sundial*, and behavioral taxonomy to classify responders
- Sundial Zmap module and all data from our study available: https://www.cmand.org/sundial/

Thanks!

Backup

Future Work

- Probe all IPv4 addresses; done, >220M responders!
- Fine-grained OS fingerprinting, e.g., intra-manufacturer behavior
- Uptime: continual measurement of hosts that keep uptime rather than reset at midnight UTC.

#dosattack

From: **Lopez, Raymond** <Raymond_Lopez3@comcast.com>
Date: Wed, Mar 20, 2019 at 2:58 PM
Subject: dosattacks
To: bauer@mit.edu <bauer@mit.edu>
Cc: Wilkerson, Joseph <Joseph_Wilkerson@comcast.com>, Halcomb, Joshua <Joshua_Halcomb@comcast.com>

Hello, my name is Ray and im a communicatons technician at comcast. Just curious why our ip is in your research survey. Ive seen it a few times in our modems so you probably have a big block of IPs we use .

Please remove comcast IPS from your list