Measuring and Characterizing IPv6 Router Availability

Robert Beverly*, Matthew Luckie†, Lorenza Mosley*, kc claffy†

*Naval Postgraduate School
†UCSD/CAIDA

March 20, 2015

PAM 2015 - 16th Passive and Active Measurement Conference

Outline

Infrastructure "Uptime:"

- More formally: uninterrupted system availability
- Duration between device restarts
- Restarts due e.g. to planned device reboots, crashes, power failures

Our Work:

1. Development of an active network measurement technique to infer infrastructure uptime
2. Uptime measurement survey of $\sim 21,000$ IPv6 router interfaces over 5-month period
3. Validation of our uptime inferences by five autonomous systems

Infrastructure "Uptime:"

- More formally: uninterrupted system availability
- Duration between device restarts
- Restarts due e.g. to planned device reboots, crashes, power failures

Our Work:

1. Development of an active network measurement technique to infer infrastructure uptime
2. Uptime measurement survey of $\sim 21,000$ IPv6 router interfaces over 5-month period
3. Validation of our uptime inferences by five autonomous systems

Why

Who wants uptime data?

- Researchers
- Operators
- Policy makers
- Regulators:
  - For instance, FCC mandates reporting voice network outages (but not broadband network services)

- Despite importance of Internet as critical infrastructure, little quantitative data on Internet device availability exists!

Why

Who wants uptime data?

- Researchers
- Operators
- Policy makers
- Regulators:
  - For instance, FCC mandates reporting voice network outages (but not broadband network services)

- Despite importance of Internet as critical infrastructure, little quantitative data on Internet device availability exists!

Uptime and Security

Security Implications

- Understand whether a reboot-based security update/patch could possibly have been applied to a device (or whether device likely still vulnerable)
- Determine if an attack designed to reboot a device is successful
- Gain knowledge of a network's operational practices and maintenance windows

Obtaining Remote Uptime

How to remotely obtain uptime?

- Just login?
- Management protocols (e.g. SNMP)?
  - ...requires access privilege

Prior Network Availability Work:

- nmap, netcraft: use TCP timestamp rate to estimate uptime
  - ...only for old operating systems w/ low-frequency clocks
  - ...restricted to infrastructure w/ listening TCP
- Prevalence and persistence of BGP routes [P97, RWXZ02]
- Operational mailing lists [FB05]
  - ...indirect measures unreliable, miss events
- Edge probing [QHP13]
  - ...not infrastructure, not uptime

Obtaining Remote Uptime

How to remotely obtain uptime?

- Just login?
- Management protocols (e.g. SNMP)?
  - ...requires access privilege

Prior Network Availability Work:

- nmap, netcraft: use TCP timestamp rate to estimate uptime
  - ...only for old operating systems w/ low-frequency clocks
  - ...restricted to infrastructure w/ listening TCP
- Prevalence and persistence of BGP routes [P97, RWXZ02]
- Operational mailing lists [FB05]
  - ...indirect measures unreliable, miss events
- Edge probing [QHP13]
  - ...not infrastructure, not uptime

Objective

Instead, our objective:

- Find uptime of remote routers...
- which don't accept TCP connections from untrusted sources...
- without privileged access...
- using active measurement

Outline

Obtaining an Identifier

- Fundamentally, our work is active fingerprinting
- Uses an identifier from the router's IPv6 control plane stack

Obtaining an Identifier for IPv6 Routers

- We leverage our prior work on IPv6 alias resolution: **too-big-trick** (PAM 2013), **speedtrap** (IMC 2013)
- To remotely obtain an identifier without privileged access

Obtaining an Identifier

- Fundamentally, our work is active fingerprinting
- Uses an identifier from the router's IPv6 control plane stack

Obtaining an Identifier for IPv6 Routers

- We leverage our prior work on IPv6 alias resolution: **too-big-trick** (PAM 2013), **speedtrap** (IMC 2013)
- To remotely obtain an identifier without privileged access

Obtaining an Identifier

IPv6 Fragmentation Background

- No in-network fragmentation in IPv6
- If next hop interface MTU is smaller than packet, routers:
  - drop packet
  - send ICMP6 “packet too big” (PTB) to source
- IPv6 stack receiving PTB:
  - Caches per-destination maximum MTU
  - Sends packets with length > PMTU using IPv6 fragment header extension
- IPv6 fragment header contains ID

Prior Insight:

Router’s control plane also implements PTB cache and sends fragments if necessary – providing an ID

Obtaining an Identifier

IPv6 Fragmentation Background

- No in-network fragmentation in IPv6
- If next hop interface MTU is smaller than packet, routers:
  - drop packet
  - send ICMP6 “packet too big” (PTB) to source
- IPv6 stack receiving PTB:
  - Caches per-destination maximum MTU
  - Sends packets with length > PMTU using IPv6 fragment header extension
- IPv6 fragment header contains ID

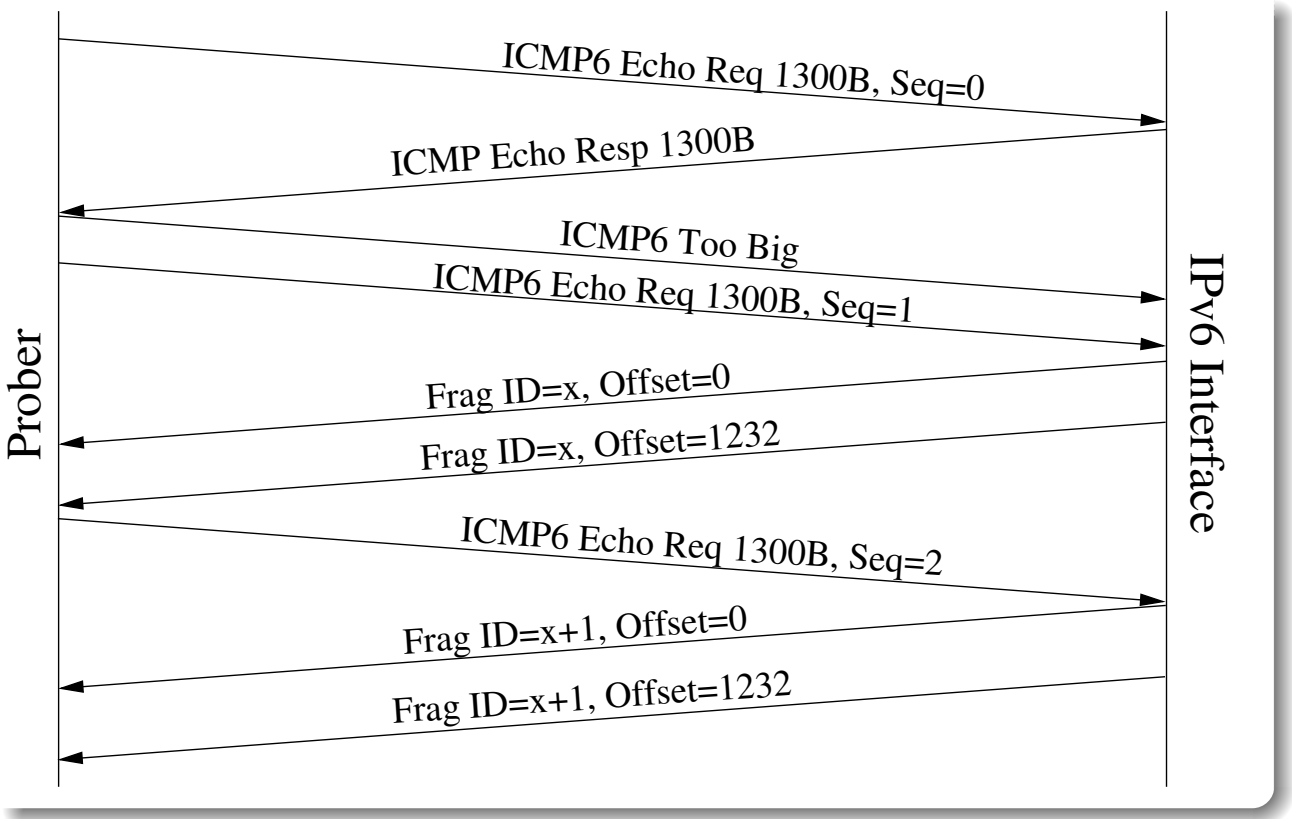
Prior Insight:

Router’s control plane also implements PTB cache and sends fragments if necessary – providing an ID

Too-Big Trick

Too-Big Trick

- Our prober sends ICMP6 echos and fake PTBs
- Inducing remote IPv6 router to *originate* fragmented packets

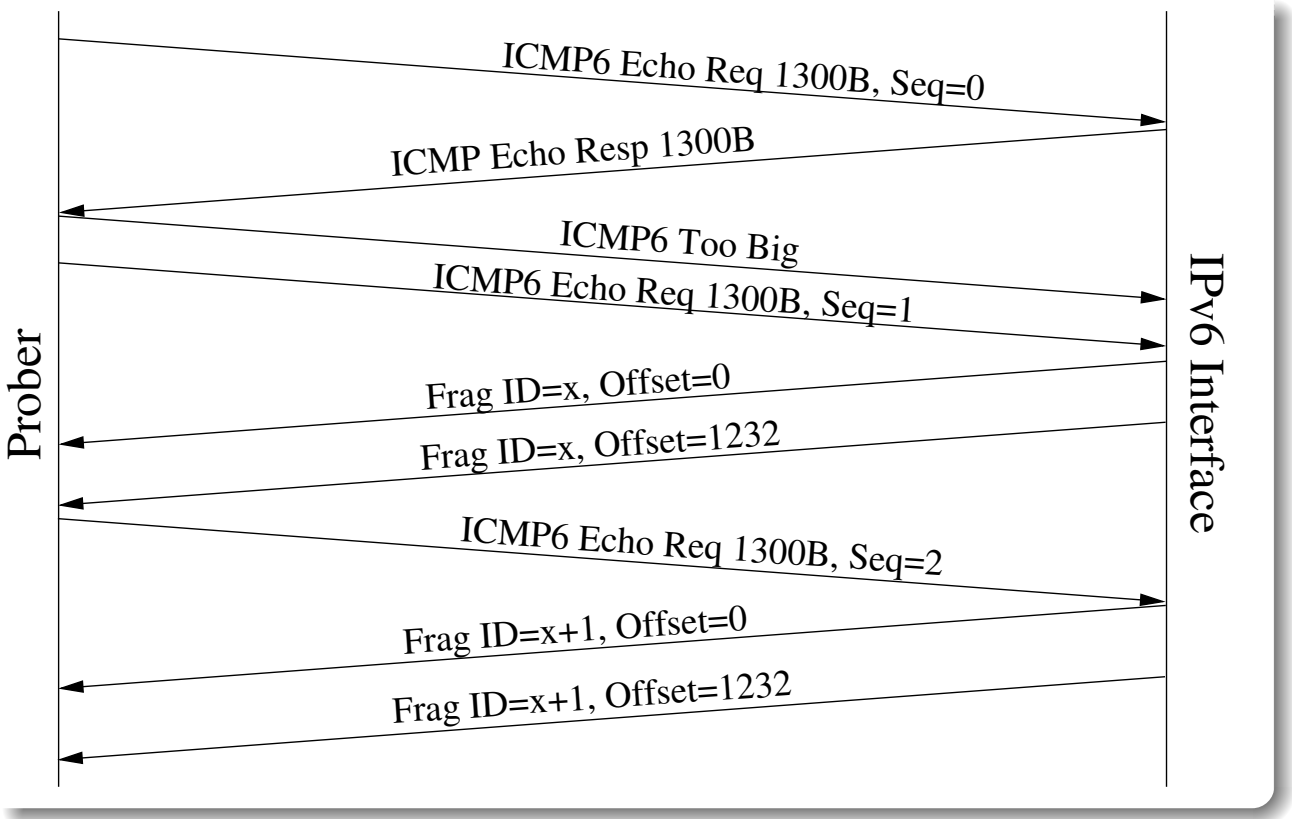


Fragment identifier is (frequently) monotonically increasing and resets to 0 on (most) IPv6 stacks, including routers

Too-Big Trick

Too-Big Trick

- Our prober sends ICMP6 echos and fake PTBs
- Inducing remote IPv6 router to *originate* fragmented packets



Fragment identifier is (frequently) monotonically increasing and resets to 0 on (most) IPv6 stacks, including routers

Methodology

High-Level:

- Periodically probe IPv6 routers with PTB and ICMP6 echo request (using scamper packet prober)
- For interface k, obtain a time series of fragment IDs and timestamps: $F_k = (f_1, t_1), (f_2, t_2), \ldots, (f_n, t_n)$ where $t_i < t_{i+1}$
- If $f_{i+1} < f_i$, then k rebooted between t_{i+1} and t_i

Real example, 3 probes per cycle:

Mar 4 21:30:01: `0x00000001, 0x00000002, 0x00000003`
Mar 5 04:25:05: `0x00000004, 0x00000005, 0x00000006`
...
Apr 21 09:39:12: `0x000001b0, 0x000001b1, 0x000001b2`
Apr 21 16:42:54: **`0x00000001`**`, 0x00000002, 0x00000003`

Real-world heterogeneity

Not as easy in practice:

- Odd behaviors, corner cases require de-noising, e.g.,:
  - `.., 405, 406, 407, 850815256, 408, 409, ...`
- Different router vendors == Different IPv6 stacks
- BSD-based devices (notably Juniper) return random fragment IDs
- Linux-based devices return cyclic fragment IDs

Cyclic Fragment IDs

Linux Kernel 3.1-3.9:

- Sets the fragment counter per-*inet peer* using keyed hash of destination IP
- The per-*inet peer* data structure times out or is garbage collected
- Hence, we get the same repeating sequence every probe cycle
- Can still detect reboots, because the random secret for the hash is recomputed at system start!

Real example, 3 probes per cycle:

Mar 27 16:42:31: `0x7943f889, 0x7943f890, 0x7943f891`
Mar 27 22:01:41: `0x7943f889, 0x7943f890, 0x7943f891`
...
Apr 26 17:45:02: `0x7943f889, 0x7943f890, 0x7943f891`
Apr 26 22:52:12: `0xc2f9dcd7, 0xc2f9dcd8, 0xc2f9dcd9`

Cyclic Fragment IDs

Linux Kernel 3.1-3.9:

- Sets the fragment counter per-*inet peer* using keyed hash of destination IP
- The per-*inet peer* data structure times out or is garbage collected
- Hence, we get the same repeating sequence every probe cycle
- Can still detect reboots, because the random secret for the hash is recomputed at system start!

Real example, 3 probes per cycle:

Mar 27 16:42:31: `0x7943f889, 0x7943f890, 0x7943f891`
Mar 27 22:01:41: `0x7943f889, 0x7943f890, 0x7943f891`
...
Apr 26 17:45:02: `0x7943f889, 0x7943f890, 0x7943f891`
Apr 26 22:52:12: **`0xc2f9dcd7`**`, 0xc2f9dcd8, 0xc2f9dcd9`

Outline

Data Collection

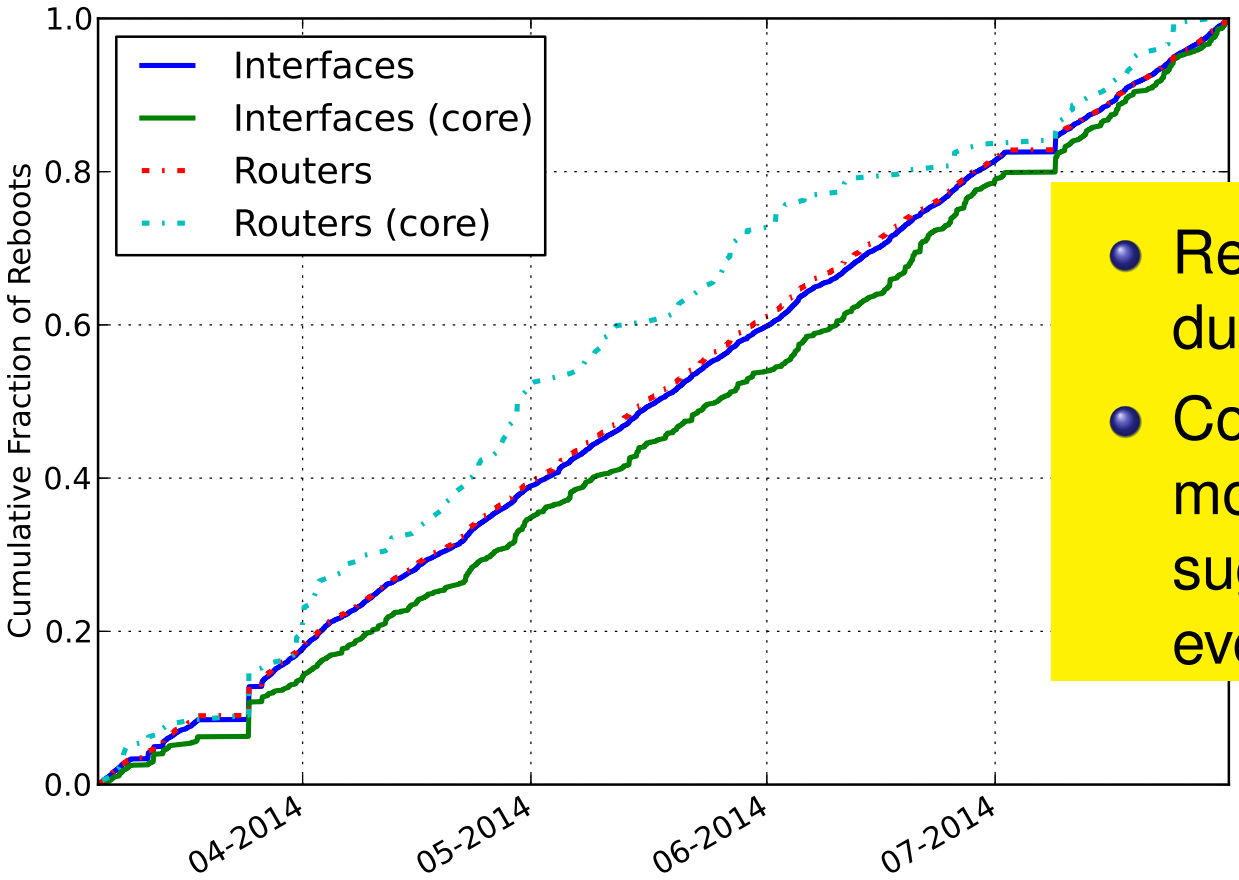
Data

- Gathered 66,471 IPv6 interfaces from CAIDA's Ark traceroutes (31,170 unresponsive, 13,330 random)
- We probed 21,539 distinct IPv6 router interfaces that return monotonic or cyclic fragment IDs
- Probed each on average every 6 hours from March 5 - July 31, 2014 from single native IPv6 vantage point

Interface Reboots $\rightarrow$ Router Reboots (see paper for details)

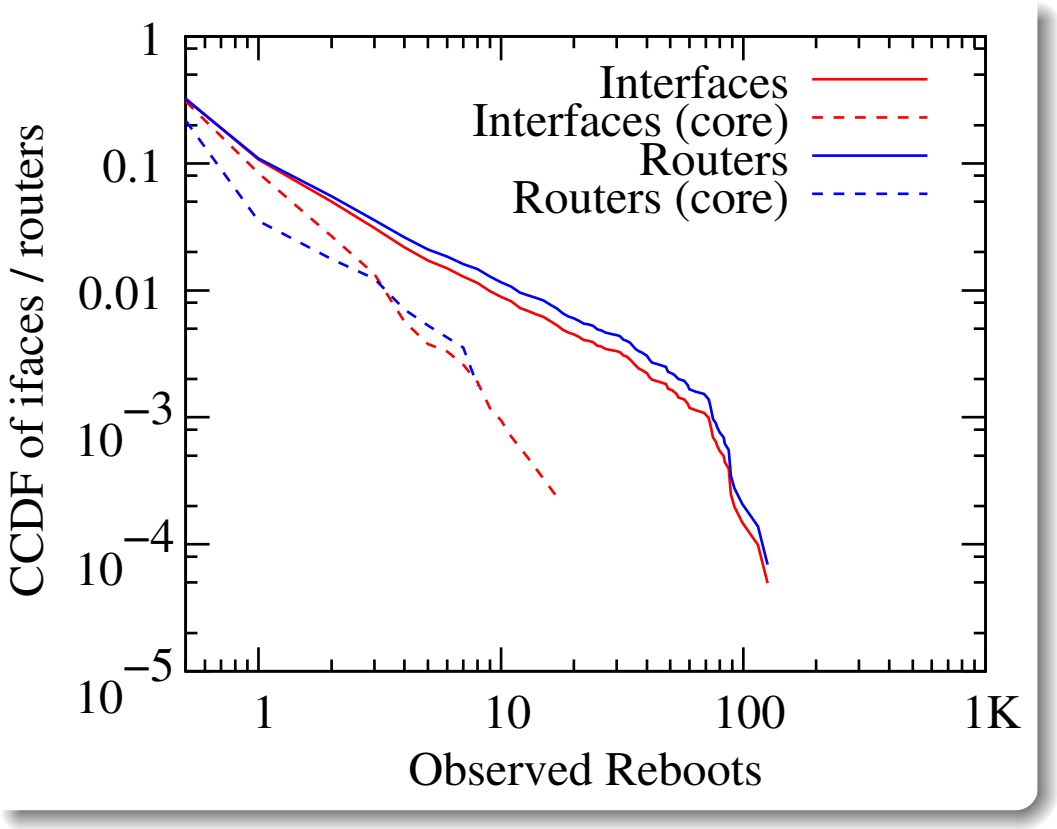
- Use Speedtrap to resolve aliases
- Separate into "core" routers (intra-AS) versus border routers (inter-AS)

Results



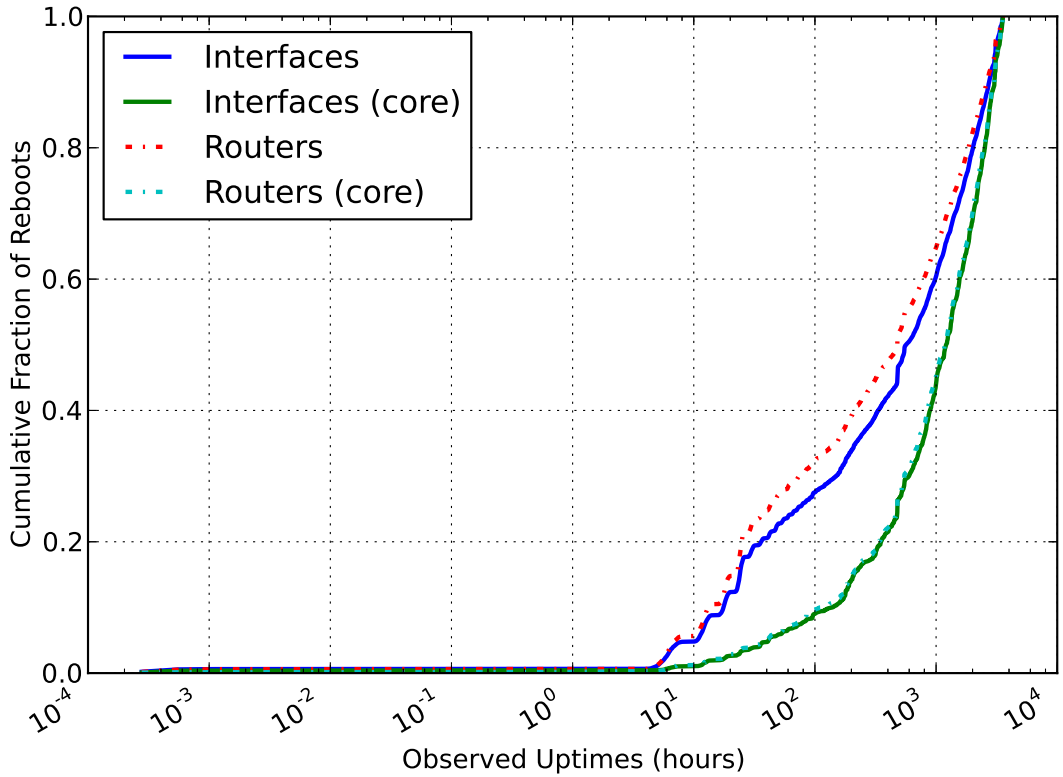
- Reboots throughout duration of experiment
- Core routers exhibit more variation, suggesting correlated events

Results



- Overall, 68% of interfaces had no reboots, while 22% had one
- Core routers and interfaces relatively more stable
- 78% of core routers had no reboots, 98% rebooted $\leq$ 2 times

Results



- Experiment duration: about 150 days
- 15% of uptimes were less than 1 day
- Median uptime of 23 days
- 10% had uptime $\geq$ 125 days

Validation

Solicited Validation from Operators of 12 ASes:

- 5 operators confirmed our inferences
- Total of 15 router restarts validated
- No false positives
- Reboots on May 18 and June 1, 2014:
  - Operators confirmed; due to TCAM exhaustion
  - Predates 512K FIB bug discussion in August, 2014!

When do Routers Reboot

- Geolocate routers to infer timezone using NetAcuity
- Weekend reboots much less likely (maintenance windows during week)

Reboots by day-of-week

	Core		All	
Monday	110	9.7%	925	11.2%
Tuesday	226	20.0%	1684	20.4%
Wednesday	227	20.0%	1553	18.8%
Thursday	197	17.4%	1313	15.9%
Friday	157	13.9%	1120	13.5%
Saturday	115	10.2%	864	10.4%
Sunday	101	8.9%	813	9.8%
	1133		8272	

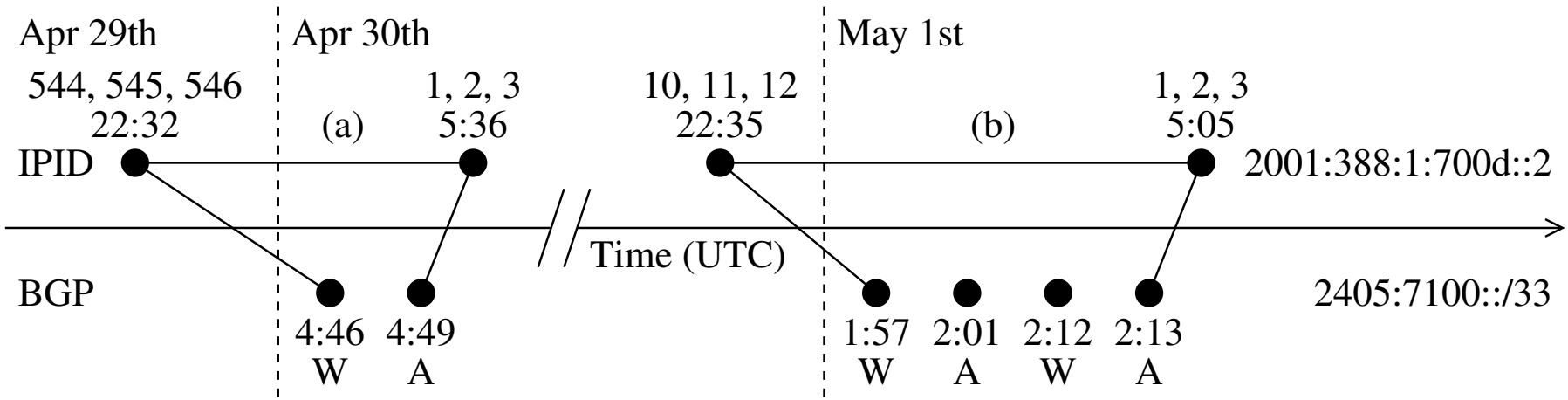
Control Plane Correlation

Correlation

- Finally, we sought to determine if the reboot events we infer are also observed in the control plane
- Manually searched routeviews BGP data for a prefix withdrawal corresponding to a reboot
- Focused on customer routers single-homed to provider (where a globally visible withdrawal is likely)

Example Reboot Correlation w/ BGP

- CPE router at AAD, customer of AARNet
- Upper dots represent our inferred reboot events for router with interface `2001:388:1:700d::2`
- Lower dots represent global BGP events for the prefix (`2405:7100::/33`) announced by the router



Outline

Summary

- Developed technique to infer the uptime of remote IPv6 devices without privileged access
- First quantitative wide-scale study of IPv6 router availability and reboot behavior

Thanks!

Questions?

`http://www.cmand.org/ipv6/`

Backup Slides

Limitations

Limitations of methodology:

- Only applicable to IPv6; IPv4 is subject of current research
- Does not work for random fragment IDs (Juniper)
- Inferred reboot granularity limited to polling rate
- Can't detect multiple reboots that occur between polls
- Can't attribute reboot to root cause (power failure, software fault, upgrade)

Future Directions

Future Directions

- Probe and characterize other IPv6 critical infrastructure, e.g. web and DNS servers
- Smarter/faster probing techniques to increase granularity of reboot time inferences
- Broader correlation with IPv4 and IPv6 BGP events
- Develop uptime inferences for IPv4