2015

Measuring and Characterizing IPv6 Router Availability

Beverly, Robert

http://hdl.handle.net/10945/46378

Measuring and Characterizing IPv6 Router Availability

Robert Beverly[1], Matthew Luckie[2], Lorenza Mosley[1], and kc claffy[2]

[1] Naval Postgraduate School, Monterey, CA
[2] CAIDA, UC San Diego, CA
rbeverly@nps.edu, mjl@caida.org, ldmosley@cmand.org, kc@caida.org

Abstract. We consider the problem of inferring IPv6 router uninterrupted system availability, or *uptime*, from a remote vantage point without privileged access. Uptime inference is important to broader efforts to measure and characterize the availability of critical infrastructure, provides insight into network operations, and has subtle security implications. Our approach utilizes active probes to periodically elicit IPv6 fragment identifiers from IPv6 router interfaces, and analyzes the resulting identifier time series for reboots. We demonstrate the approach's potential by characterizing 21,539 distinct IPv6 router interfaces over a five-month period. We find evidence of clustered reboot events, popular maintenance windows, and correlation with globally visible control plane data. Our results, validated by five ASes, provide initial insight into the current state of IPv6 router availability.

1 Introduction

Significant recent work examines IPv6 adoption [6], usage [23], and performance [8]. Less well-studied is the *reliability* of IPv6 infrastructure. This paper focuses on measuring and characterizing the reliability of one of the most critical components of IPv6 infrastructure: IPv6 routers. Understanding IPv6 router reliability provides insights into not only the current maturity of production IPv6, but also operational properties of IPv6 networks, including vulnerability information.

We develop, analyze, and validate a technique to remotely estimate, without privileged access, the uninterrupted system availability, or *uptime*, of IPv6 devices. Our technique relies on inducing IPv6 fragments from remote endpoints and analyzing the sequence of IPv6 fragment identifiers returned as a result of periodic probing. Importantly, our active probing consists of ICMP6 echo requests and therefore is conducive to characterizing devices that do not permit TCP connections, e.g. routers. As a proof-of-concept, we apply our technique over a five-month period to a collection of 66,471 IPv6 router interfaces on the Internet; our technique made uptime inferences on 21,539 (32%) of these interfaces ($\simeq$47% of the interfaces were unresponsive, while 21% did not permit uptime inference). We validate our technique against five providers that positively confirm our reboot inferences. We find that while 68% of interfaces and 78% of core routers experience no restarts during our measurement period, a few devices experience many restarts. We further discover evidence of correlation between restarts and global BGP events observed in public route collectors. Finally, we show that router restarts occur most frequently on Tuesday and Wednesdays, and least often on weekends. Our contributions include:

1. An active probing method that permits remote IPv6 router uptime inference without privileged access.
2. Real-world deployment and validation of the technique.
3. Insights into how different devices send IPv6 fragments over time, including a previously unstudied cyclic behavior exhibited by Linux-based devices.
4. A five-month study of reboots among 21,539 IPv6 router interfaces where we find that core routers tend to have longer uptimes than border routers.

2 Technique and Data for Inferring IPv6 Router Uptime

We assume that a router's uptime can be estimated by inferring the last time the router rebooted, i.e. a *reboot event*. To infer a router reboot event, we rely on the fact that many routers maintain externally observable state that resets when rebooted. Specifically, the control plane IPv6 stack implementation on many routers maintains monotonically increasing IPv6 fragment identification (ID) counters that are initialized to zero or a random number [2]. By periodically probing routers to obtain and increment their ID field, and segmenting the time-series of IDs into monotonically increasing subsequences, we can infer that reboot events occur in the periods between subsequences.

In this section, we first explain the IPv6 ID field and our technique to obtain the ID time series. We then describe our experimental methodology and data (§2.2) and our algorithm (§2.3). We detail how we handle an important subset of interfaces (those that return cyclic ID sequences) in §2.4, and describe our procedure to identify routers and annotate them with their role and their timezone (§2.5). We discuss limitations of our technique in §2.6.

2.1 Obtaining and using the IPv6 ID field

Unlike the IPv4 header, the IPv6 header lacks an IP ID field used for fragmentation and reassembly. IPv6 routers perform no in-network packet fragmentation; the IPv6 protocol shifts this burden of fragmentation to the sender. If a sender must fragment a packet, it adds an IPv6 extension header on each packet fragment that includes a 32-bit ID field to facilitate reassembly. While routers primarily perform data-plane forwarding, they also run an IPv6 stack as part of their control plane. Building on our technique in [2], now implemented in speedtrap [16] and integrated into the scamper packet prober [15], we elicit IPv6 IDs by sending ICMP6 packets to a router's control plane via one or more of its interfaces. Specifically, we induce a router's IPv6 stack to originate IPv6 fragment IDs by sending an ICMP packet too big message (PTB) with an MTU value smaller than the size of the packets subsequently solicited from it. We use scamper to first send 1300-byte ICMP echo request packets; when we receive

1300-byte echo replies, we send the router a PTB message with an MTU of 1280 bytes. If the router follows the IPv6 protocol [7], it will subsequently send fragmented echo replies containing fragment IDs in response to our probes.

Note that except for responses to our probes, a router does not typically send fragmented IPv6 traffic, and hence the IPv6 counter has no natural background rate of change (velocity). In contrast, control plane IPv4 packets sent by a router increment the fragment ID counter because every IPv4 packet contains a fragment identification field. In addition, the IPv4 fragment identifier field is 16 bits, but 32 bits in IPv6. Our prior work used IPID-based inferences to infer IPv6 router aliases [16]; in this work we extend this method to enable new inferences, leveraging the monotonicity of IPv6 IDs to infer router uptime.

2.2 Obtaining IPv6 router interface addresses

We assemble a set of candidate IPv6 router interfaces from traceroutes conducted in January and February 2014 by CAIDA's macroscopic IPv6 topology discovery infrastructure [12]. The union of all interfaces discovered in this period across 32 geographically distributed vantage points (VPs) includes 66,471 unique IPv6 router interfaces. Although these interfaces are a subset of the complete IPv6 Internet, the set is sufficiently large and diverse to demonstrate our uptime inference technique, and reveal preliminary insights into IPv6 router availability.

We probed these interfaces every 6 hours between March 5th and July 31st 2014 from a single host on the Virginia Tech campus (an educational network on the U.S. east coast with native IPv6). The set of interfaces were randomly permuted before each probing round. For each interface, we sent four ICMP6 echo requests (per §2.1); probing this set of addresses at 20 packets per second required approximately 2.5 hours per run. Our probing host had two multi-day outages: March 18-25 and July 2-9, 2014.

2.3 Uptime algorithm

For each IPv6 interface k, our periodic probing produces a time series of n IPv6 fragment ID (f_i) and timestamp (t_i) pairs: $F_k = (f_1, t_1), (f_2, t_2), \ldots, (f_n, t_n)$ where $t_i < t_{i+1}$. Some interfaces were unreachable while ICMP6 blocking prevents the PTB from reaching others; in such cases F is the empty set[1]. 31,170 interfaces (46.9%) either were unresponsive (e.g. due to ICMP6 filtering, address changes, network changes) or did not return fragment IDs (e.g. due to PTB or fragment filtering).

For the remaining 35,301 interfaces that returned IDs, we observed a variety of router implementation-specific behavior. In total, 20.1% of the interfaces returned random IDs; our prior work [16] found that random IDs were attributable to BSD-based devices, including Juniper routers. Because random IDs are by definition non-monotonically increasing, we cannot form uptime inferences over this

[1] A small fraction of the interfaces return only a small number of IDs over the experiment duration; we exclude those where $n < 20$

Classification	Interfaces	
Monotonic	20,429	30.7%
Cyclic	1,110	1.7%
Odd	432	0.6%
Random	13,330	20.1%
Unresponsive	31,170	46.9%
Total	66,471	100.0%

Table 1. Classification of IP-ID behavior. We can infer reboot events for interfaces we classify as monotonic or cyclic, and some events for interfaces we classify as odd.

set. Thus, before performing uptime inference, we segment the interfaces in our dataset into classes as summarized in Table 1.

The classification logic divides a time series F_k into sequentially increasing subsequences such that $f_i + 1 = f_{i+1}$. This step breaks random ID series into singleton subsequences, while preserving groups of monotonic runs. We infer interfaces with all singleton sequences to be random, and classify each non-random subsequence as monotonic or cyclic. If all labels agree, we classify the interface with that label.

Several factors complicated this classification. When the labels for subsequences did not all agree, we classified the interface as *odd*. For instance, some interfaces changed behavior during the course of our experiment, suggesting a hardware or software change. Other interfaces returned deterministic IDs, e.g. always zero, or returned multiple replies for each probe. In total, 0.6% of the interfaces exhibited odd or inconsistent behavior, and we excluded them from our analysis. Another complicating factor is interfaces that return cyclic IDs with large offsets. We identify cyclic interfaces as those with IDs greater than 10,000 that appear in multiple subsequences. In §2.4 we discuss root causes of cyclic IDs and how we accommodate them.

Finally, we infer uptimes for the set of interfaces we classify as monotonic. First, we filter noise in the time series. For example, we obtained an f sequence: `..., 405, 406, 407, 850815256, 408, 409, ...` . At present, we cannot positively identify the cause of these infrequent, but clearly erroneous IDs. We therefore remove element i if and only if $f_{i-1} + 1 = f_{i+1}$, i.e. we remove an outlier in the midst of an otherwise exact sequence. We then form monotonically increasing subsequences such that if $f_{i+1} < f_i$, we know that a reboot event occurred between t_i and t_{i+1}. Because $t_{i+1} - t_i$ is bounded by the frequency at which we probe, as much as six hours in our experiment, there is inherent error. The difference between the last and first sample in a subsequence therefore provides a conservative uptime estimate.

2.4 Cyclic interfaces

We classified 1.7% of the interfaces in our set as cyclic because they exhibited an incrementing but cyclic pattern of IP-ID values, e.g. (N, N+1, N+2, N, N+1, N+2). This interface IP-ID behavior is consistent with some versions of the Linux kernel. The last version of the Linux kernel that used a single central counter was 3.0, released Jul 21, 2011. For Linux kernel versions 3.1 (released Oct 24, 2011) through 3.9 (released Jun 30, 2013) the kernel uses a counter per destination IP address, with the initial value computed as a function of (1) the destination IP address and (2) a randomly generated secret obtained when the system booted. The kernel creates an *inet peer* structure per IP address, where it stores information including the next IP-ID value to use when sending a packet. These *inet peer* structures are discarded when the route times out or is garbage collected (the kernel is limited to 65,664 inet peer structures). When the structure is later recreated for the same destination IP address (for instance during our next probing round) it will use the same initial IP-ID value if neither the secret value (initialized on system boot) nor the destination address value (which we control) changed. As a real world example, a particular interface in our dataset returned the same sequence f = `0x28c2c283, 0x28c2c284, 0x28c2c285` on each probing cycle until it rebooted and then returned a different sequence with the same period: f = `0x415bd0cc, 0x415bd0cd, 0x415bd0ce`.

For these cyclic interfaces, we detect a reboot event as an abrupt change in IP-ID value, which indicates the secret has changed. We empirically define an abrupt change as either an IP-ID value that is lower than the range of previous values or at least 2000 higher.

2.5 Inferring routers, their roles, and their location

We used speedtrap [16] to resolve aliases (i.e. map multiple IP addresses to the same physical router) for the set of 66,471 monitored interfaces. The speedtrap resolution was performed between Sept 19th 16:00 UTC and Sept 20th 07:00. Speedtrap also exploits IPv6 fragmentation identifiers: two interfaces are aliases for the same router if they produce a sequence of non-overlapping IPID samples whose IPID values strictly increase, suggesting the IPID samples are derived from the same counter. Because we ran speedtrap from a different vantage point than from where the uptime IPID samples were collected, and after our data collection completed, our inferred aliases do not perfectly overlap with the interfaces probed. Our speedtrap alias resolution run observed 19,103 interfaces that assign ID values from a monotonically increasing counter. Using speedtrap to find aliases, these interfaces correspond to 12,866 routers. For 9,035 interfaces (70.2%), speedtrap inferred no aliases, and we treat these as routers with a single interface. 20.1% of the remaining routers had two interfaces, leaving approximately 10% of the routers with three or more aliases. For the remaining monotonic and cyclic interfaces that were unresponsive to speedtrap probes, we did not infer aliases and treat each as a router with a single interface.

A well-known limitation of mapping IP addresses to ASes is that an interface may be mapped to a different AS than the AS that owns and operates the router. For example, network service providers frequently allocate one of their own IP addresses to an interface on a customer router. This ambiguity affects our uptime analysis because the reboot of an interface with an address of provider A may actually be a reboot event within customer B's network.

We therefore classify some routers as being *core AS routers* if we believe they represent a router within an AS (intra-AS) as opposed to a border router connecting to other ASes (inter-AS). We examined the AS origin of each IPv6 hop in the corresponding Ark traceroute data for Jan and Feb 2014. We classified a router with interface B_2 as a core router for AS B if we observed a traceroute IPv6 address sequence $A_1B_1B_2B_3C_1$ where AS B originates BGP prefixes for B_1, B_2, and B_3. In contrast, neither B_1 nor B_3 would be classified as a core AS router with this path because they are preceded and succeeded by interface hops belonging to different ASes. This conservative definition of a core AS router allows us to better characterize the origin of reboot events for some large networks with many customers. We inferred 20,093 interfaces as belonging to core routers in their respective ASes.

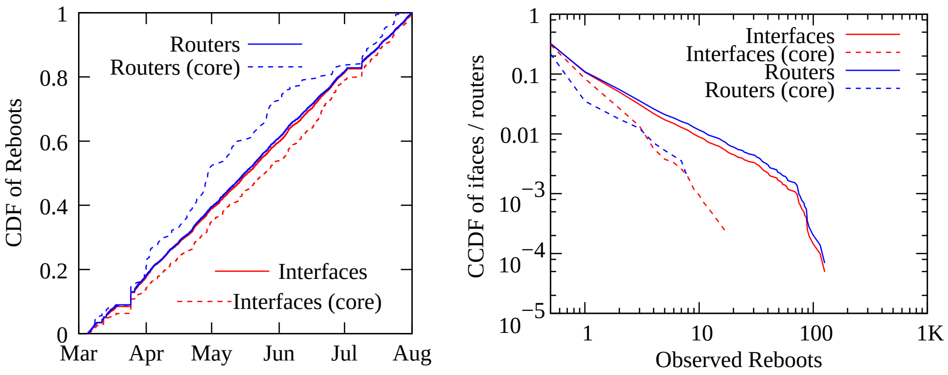
Finally, we annotated each router with an inferred local timezone, based on its offset from GMT reported by Digital Envoy's NetAcuity commercial geolocation database [9]. This database reported GMT offsets for 65,451 of 66,471 (98.5%) of the interfaces probed. For routers with more than one observed address, we mapped all interfaces to the same timezone, i.e. we discovered no disagreement among timezones of the interfaces belonging to routers. The quality of IPv6 geolocation databases for individual router interfaces is unknown, and we only probed interfaces every six hours, so we were wary of inferring fine-grained temporal patterns of rebooting. However, we use timezones to estimate the aggregate distribution of reboots across days of the week (§3.4).

2.6 Limitations

Our uptime technique has several limitations. First, inferences are only possible for those interfaces that return IPv6 fragments, and only for the subset of those with non-random IDs. In addition, because of potential security vulnerabilities introduced by fragmentation, future IETF guidance may deprecate IPv6 fragmentation [4], thereby invalidating our technique. However, in practice we are able to elicit fragments from a large fraction of production routers and do not expect this ability to change in the near-term.

Second, our technique depends on periodic probing. The granularity of our uptime inferences is governed by the rate of probing the remote interface; obtaining high-fidelity uptime inferences may induce unwanted traffic load, especially as the IPv6 Internet grows. Further, we cannot detect multiple reboots of an interface that occur between probing samples. Third, we cannot discern the root cause of a reboot, e.g. a power failure, natural disaster, human error, software fault, or intentional maintenance upgrade.

We currently focus our effort on IPv6 routers. While similar IP-ID behavior is found in IPv4 routers, there are three important differences. First, the IPID counter behavior in IPv4 is much more erratic because routers increment the ID counter every time they create a packet, causing the counter's velocity to be large for routers with chatty routing protocols or SNMP reporting [1, 13]. Second, the counter itself only has a range of 65536 values, requiring more frequent probing than in IPv6 to prevent a counter wrap from being interpreted as a reboot.



(a) Reboot events over time

(b) Reboot events per device

Fig. 1. Distribution of reboots in time and frequency for interfaces and routers measured over five-months. Core interfaces and routers refer to intra-AS devices.

Finally, there are many more router interfaces in the IPv4 topology than in the IPv6 topology; even conducting Internet-scale IPv4 alias resolution on >2M IP addresses is challenging due to the volume of probes required. Sampling router interfaces to infer reboot events would require even more frequent probing than Internet-scale alias resolution. We leave IPv4 uptime inference to future work.

3 Results

We used our uptime inference algorithm to characterize the availability of IPv6 interfaces and routers. Figure 1(a) shows the cumulative distribution of reboot events for the duration of our experiment. The overall rate of interface reboots is relatively uniform – indicating a constant background rate of IPv6 interface reboots without the presence of individual events affecting many interfaces. In contrast, the set of core routers exhibits more variation, suggesting correlated reboot events among routers within a provider or organization.

Figure 1(b) depicts the complementary cumulative distribution of interface and router reboots. The distribution is heavy-tailed: most routers and interfaces experienced no reboots while a few experienced many reboots. Overall, 68% of the interfaces we monitored experienced no reboots over the measurement period, while ∼22% of interfaces had a single reboot. 99% of interfaces had 10 or fewer reboots, but three interfaces reboot more than 100 times. Core interfaces and routers were more stable than the broader set. 78% of core routers did not reboot during our experiment, but 98% rebooted two or fewer times.

Figure 2 shows the distribution of uptimes, inclusive of only those devices that experienced a reboot. Only 15% of observed uptimes were less than a day; the median interface uptime was approximately 23 days. Reboots among core interfaces and routers are again relatively more stable, with a median uptime of approximately 50 days, while 10% had an uptime of 125 days or more. The

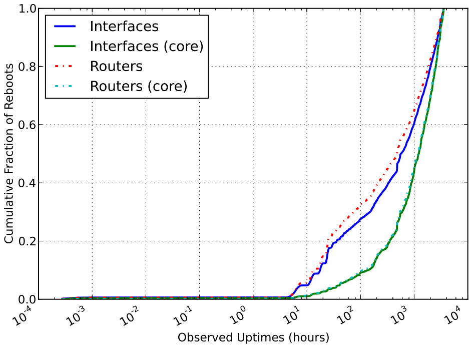


Fig. 2. Distribution of observed uptimes across all observed reboot events (excluding routers and interfaces with no restarts).

largest uptime, approximately 150 days, corresponds to our full measurement period and represents an interface that rebooted at the beginning of our probing.

3.1 Linux router behavior

As described in §2.4, Linux kernels between 3.1 and 3.9 use a separate counter per source IP address, and the counter appears to wrap to the same initial ID value when the state associated with the source IP address is removed. We detect reboots when we observe an abrupt change in IP-ID value that implies the randomly generated secret used to set the initial ID value has changed. In total, we detected 2,312 reboot events involving the 1,110 cyclic-ID interfaces. The events were evenly distributed throughout the five months of our probing.

3.2 Validation

We solicited validation data from operators of 12 ASes who had previously provided feedback on our AS relationship inferences [17]. Five replied with evidence that supported our inferences of reboot events for 15 routers, either direct validation from system logs, or implicit validation by correlating the event with a BGP session closing. Operators could not confirm all reboots we asked them about, since some routers were using the operator's address space but belonged to customers, so the operator could not verify uptime. Through operator feedback we also learned that the two reboot events we detected for one router on May 18 and June 1 2014 were because the router ran out of TCAM to store the routing table; these reboots occurred before the publicity in August 2014 where individual provider tables reached 512K [5].

3.3 Reboot event correlations with BGP

We manually searched public BGP data for BGP prefix withdrawal events that correlated with our inferences of a reboot event. To rule out confounding factors

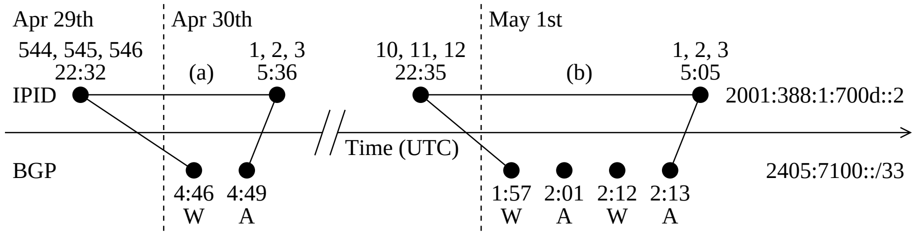


Fig. 3. Timeline of IPID events for `2001:388:1:700d::2` (assigned to a router owned by AS36474) and BGP events for the `2405:7100::/33` prefix announced by AS38474. The time of each event is represented by a dot with inferred uptime events above observed BGP events. The reboot events labeled (a) and (b) correlate in time with the withdrawal (W) and announcement (A) of the prefix as observed at routeviews. No other BGP events involving this AS occurred during this time.

involving events from upstream networks, we searched BGP data provided by the AS where we observed an event involving a customer router, as labeled in DNS. Figure 3 illustrates an example involving two detected reboot events in two days between April 29 and May 1 UTC. The interface `2001:388:1:700d::2` has a DNS PTR record of `gw1.er1.aad.cpe.aarnet.net.au`, i.e. a customer premises equipment (cpe) router at the Australian Antarctic Division (aad) which is a customer of AARNet, AS7575. AAD is AS38474 in BGP, and announced two IPv6 prefixes: `2405:7100::/33` and `2405:7100:8000::/33`. We downloaded all update messages archived from AS7575 by Routeviews' Sydney collector between April 29 and May 1 UTC, and then searched for BGP events involving these two prefixes. The reboot events labeled (a) and (b) in Figure 3 that occurred between our probing correlate with prefix withdrawal and announcement events, and no other BGP events for AS38474 occurred during this time window. We saw the same behavior involving other customers of other networks in BGP. An open question is the degree to which reboot events in core AS routers result in a BGP event; we hypothesize that neighbors of a network where a core AS router reboots are much less likely to propagate a BGP event than the case in Figure 3 where a provider propagates a BGP event caused by a customer-edge router reboot. Previous work on pinpointing the cause of a routing change [25] suggested a coordinated approach, where ASes maintain a view of routing changes within their own network, which can be queried when an event occurs. Our results demonstrate the potential to correlate customer edge router reboot events with BGP routing events.

3.4 When do routers reboot?

Table 2 reports the day of the week, in the router's local time zone, when we detect a reboot event. In our data, router reboots were more than twice as prevalent on Tuesday and Wednesday than on Saturday, Sunday, and Monday, regardless of the router classification we made. We found the reduction in reboots over the weekend relative to the rest of the week surprising. We hypothesized

	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	

Table 2. Router reboots by day-of-week and router type (§2.5). Router reboot events were twice as common on Tuesday and Wednesday as on Saturday, Sunday, or Monday.

that reboots due to maintenance would occur on the weekend when the network demand and thus potential impact of any disruption is lower. Instead, our data suggests that maintenance occurs during the middle of the week, perhaps due to the difficulty and expense of having a network team available during the weekend. We restrict ourselves to day-of-week granularity because of the relatively coarse (6 hourly) probing we used to collect the data. In the future, we would like to optimize our probing algorithm to obtain a finer granularity that would allow us to pinpoint reboot events to within one hour. For example, maintenance events might occur in early morning during weekdays, to minimize disruptive impact.

4 Applications and Implications

Despite the Internet's critical importance, relatively little quantitative data exists on its service availability or reliability. A precise definition of Internet infrastructure reliability has yet to solidify [14], although the U.S. FCC has supported efforts to measure reliability from a consumer's perspective [3, 24]. At the provider-level, the FCC mandates reporting of significant outages for voice networks, including VoIP networks [10], but there are no outage reporting requirements for broadband network services.

While anyone with management access to a router (e.g. SNMP, ssh) can determine its uptime, our technique uses ICMP6 and requires no privileged access to infer the uptime of a remote router, enabling Internet-wide study of IPv6 router availability and reliability. Prior work has sought to infer reliability indirectly. For instance, Paxson introduced metrics of routing reliability such as route prevalence and persistence [20, 22], while Feamster *et al.* analyzed operational mailing lists to characterize the frequency of faults [11]. More recently, Quan *et al.* used active probes to infer edge network availability [21]. In contrast, we restrict our attention to the availability of IPv6 routers, but obtain uptime data directly from the routers via active measurement.

Closely related to our technique are uptime inferences using TCP timestamps. For example, nmap [18] gathers remote TCP timestamps to determine the rate at which timestamps increase, and extrapolates to estimate uptime assuming the timestamp resets to zero upon boot. Netcraft uses this technique to infer the uptime of Internet web servers, but notes that uptimes cannot be determined for hosts running modern operating systems due to their use of high-frequency clocks [19]. More importantly, routers rarely listen on any TCP port, rendering active-open TCP-based uptime methods infeasible.

Inferring device reboots also has important security implications. An attacker able to observe when a remote device last rebooted can infer whether that device has installed certain security patches; devices that have not rebooted since a vulnerability announcement are more likely vulnerable. Attackers can also gain knowledge of the likely maintenance windows for different networks, as well as when an attack designed to crash a router is successful.

5 Conclusions

To our knowledge, we have developed, validated, and demonstrated the first remote uptime inference technique applicable to routers. While our method is currently limited to routers supporting IPv6, and only works for 61% of the responsive routers in our study, it is a first step toward broader insights into Internet infrastructure reliability and operational practices.

While we have demonstrated evidence of correlation between our inferred reboots and BGP events visible in the global routing table, in future work we hope to systematically investigate the relationship between reboots and both IPv4 and IPv6 routing system events. We observed instances of correlated reboots among IPv6 interfaces that are not aliases, implying that multiple routers rebooted within the same time window. Careful analysis of such correlations can reveal hidden relationships among not only routers, but also providers, and potentially reveal hidden correlations such as co-located routers rebooting due to a common power failure.

We focused on IPv6 routers, but our technique applies to any IPv6 device that responds with monotonic fragment IDs, including Linux and Windows machines serving as infrastructure, e.g. web servers, DNS resolvers, etc. A more ambitious longitudinal study, using a higher probing rate, would enable unprecedented macroscopic characterization of the availability of critical IPv6 infrastructure.

Acknowledgments

We thank Stefan Savage for the uptime idea, and to the networks that validated our inferences. This work supported by NSF grants CNS-1111445 and CNS-1111449 and Department of Homeland Security (DHS) contracts N66001-2250-58231 and N66001-12-C-0130. Views and conclusions are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. government.

References

1. Bender, A., Sherwood, R., Spring, N.: Fixing Ally's growing pains with velocity modeling. In: ACM SIGCOMM IMC. pp. 337–342 (Oct 2008)

2. Beverly, R., Brinkmeyer, W., Luckie, M., Rohrer, J.P.: IPv6 alias resolution via induced fragmentation. In: Passive and Active Measurement. pp. 155–165 (2013)
3. Bischof, Z.S., Bustamante, F.E.: A time for reliability: The growing importance of being always on. In: Proceedings of ACM SIGCOMM. pp. 131–132 (2014)
4. Bonica, R., Kumari, W., Bush, R., Pfeifer, H.: IPv6 Fragment Header Deprecated. Internet Draft (Jul 2013)
5. Cowie, J.: Internet touches half million routes: Outages possible next week (Aug 2014), `http://research.dyn.com/2014/08/internet-512k-global-routes/`
6. Czyz, J., Allman, M., Zhang, J., Iekel-Johnson, S., Osterweil, E., Bailey, M.: Measuring IPv6 Adoption. In: Proceedings of ACM SIGCOMM. pp. 87–98 (2014)
7. Deering, S., Hinden, R.: Internet Protocol, Version 6 (IPv6) Specification. RFC 2460 (Dec 1998)
8. Dhamdhere, A., Luckie, M., Huffaker, B., k claffy, Elmokashfi, A., Aben, E.: Measuring the deployment of IPv6: Topology, routing and performance. In: ACM SIGCOMM IMC. pp. 537–559 (Nov 2012)
9. Digital Element: NetAcuity Edge, `http://www.digitalelement.com/solutions/`
10. FCC: Outage reporting to interconnected voice over internet protocol service providers (2012), `https://apps.fcc.gov/edocs_public/attachmatch/FCC-12-22A1.pdf`
11. Feamster, N., Balakrishnan, H.: Detecting BGP configuration faults with static analysis. In: Proceedings of NSDI. pp. 43–56 (2005)
12. Hyun, Y., k. claffy: Archipelago measurement infrastructure (2014), `http://www.caida.org/projects/ark/`
13. Keys, K., Hyun, Y., Luckie, M., k claffy: Internet-scale IPv4 alias resolution with MIDAR. IEEE/ACM Transactions on Networking pp. 383–399 (Apr 2013)
14. Lehr, W., Bauer, S., Heikkinen, M., Clark, D.: Assessing broadband reliability: Measurement and policy challenges. In: Research Conference on Communications, Information and Internet Policy (2011)
15. Luckie, M.: Scamper: A scalable and extensible packet prober for active measurement of the internet. In: ACM SIGCOMM IMC. pp. 239–245 (2010)
16. Luckie, M., Beverly, R., Brinkmeyer, W., k claffy: Speedtrap: Internet-scale IPv6 alias resolution. In: ACM SIGCOMM IMC. pp. 119–126 (Oct 2013)
17. Luckie, M., Huffaker, B., Dhamdhere, A., Giotsas, V., k claffy: AS relationships, customer cones, and validation. In: ACM SIGCOMM IMC. pp. 243–256 (Oct 2013)
18. Lyon, G.F.: Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning. Insecure (2009)
19. Netcraft: Which operating systems provide uptime information? (June 2014), `http://uptime.netcraft.com/accuracy.html#uptime`
20. Paxson, V.: End-to-end routing behavior in the internet. Networking, IEEE/ACM Transactions on 5(5), 601–615 (1997)
21. Quan, L., Heidemann, J., Pradkin, Y.: Trinocular: Understanding internet reliability through adaptive probing. In: Proceedings of ACM SIGCOMM (2013)
22. Rexford, J., Wang, J., Xiao, Z., Zhang, Y.: Bgp routing stability of popular destinations. In: Proceedings of the 2nd ACM SIGCOMM IMW. pp. 197–202 (2002)
23. Sarrar, N., Maier, G., Ager, B., Sommer, R., Uhlig, S.: Investigating IPv6 Traffic. In: Passive and Active Measurement. pp. 11–20 (2012)
24. Sundaresan, S., De Donato, W., Feamster, N., Teixeira, R., Crawford, S., Pescapè, A.: Broadband internet performance: a view from the gateway. In: ACM SIGCOMM computer communication review. vol. 41, pp. 134–145 (2011)
25. Teixeira, R., Rexford, J.: A measurement framework for pin-pointing routing changes. In: ACM SIGCOMM NetTs Workshop (Aug 2004)