October 2025
Geoff Huston

Some notes from RIPE 91

The 91st meeting of the RIPE community was held in Bucharest in October this year. It was a busy week and, as usual, there were presentations on a wide variety of topics, including routing, the DNS, network operations, security, measurement and address policies. The following are some notes on presentations that I found to be of interest to me.

Routing the Root

I was struck by this observation in one of the presentations at RIPE 91: "In BGP, the more you look, the more you find!" These days the routed Internet is quite large. There are more than a million distinct route objects in the IPv4 network, and the IPv6 network is rapidly growing, with just under a quarter of a million distinct route objects (237,000 in October 2025). In IPv4 there are 78,000 AS numbers and some 263,000 unique AS Paths from my BGP vantage point in AS 131072. In IPv6 there are 36,000 unique ASNs and 140,000 unique AS paths. So, it's probably unsurprising to note that if you look hard enough you can probably see every form of routing anomaly. The BGP routing protocol has been in use since the early 90's, and works on a simple form of rumour propagation, common to all forms of distance vector routing protocols. Such protocols are simple and scalable, but they assume significant levels of mutual trust. When a BGP speaker collects the best routes from each of its neighbours and forwards on its selection in turn, it's assumed that the router is not deliberating misrepresenting the routing information it has learned from its neighbours.

Lefteris Manassakis, from Thousand Eyes, looked at the recorded behaviour of BGP when it carries the routes that are used to reach the Root Servers of the DNS, and analysed this data to see if there are visible traces of efforts to tamper with the routing data associated with this root servers. These root servers are "interesting" in that they make extensive use of anycast to improve the scalability and performance of the DNS root service. However, this often makes detection of anomalous routing data challenging as the root server since each instance of the anycast service cloud originates its route is a unique location. Routes to these 13 servers are originated from some 1,998 locations, so it would be unsurprising to see local attempts to disrupt this service pass by largely unnoticed.

With 12 different root service operators there are multiple ways to set up these anycast networks. Most root servers use a single AS number and announce that prefix from a single origin AS, which is deployed in multiple across the network. Verisign, the operator of the A and J root server names, use a different approach, as they use a collection of ASes as the origin AS, and then announce all these instances via a common transit AS (AS7342) (this technique is described in RFC 6382) In other cases it's a simple case of a third party AS originating a route to a root server's IP address (Figure 1).

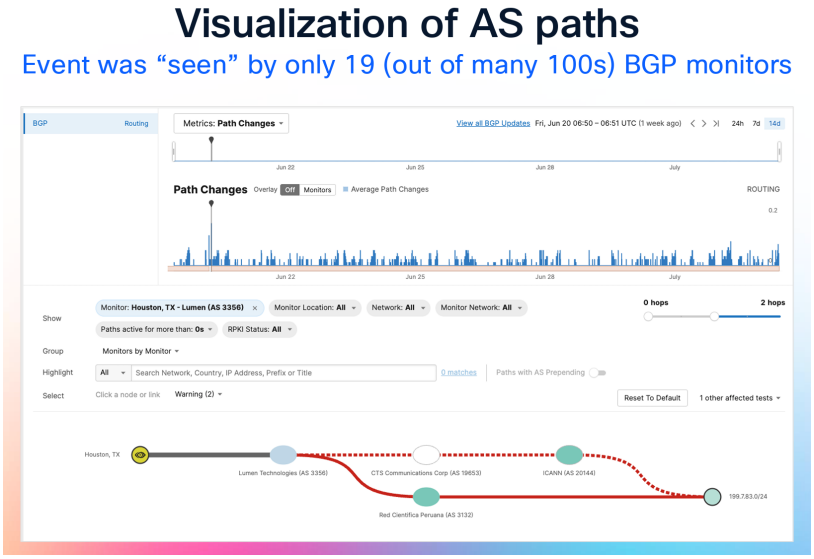

Figure 1 – Anomalous Origination L-Root Prefix (from Monitoring Root DNS Prefixes)

The presentation described some incidents where the IP addresses of root servers were observed with anomalous AS Path, suggesting that the routes were actively hijacked. It is not known what the consequences were. If the querying DNS resolver performs DNSSEC validation, then any attempt to insert bogus information into a root zone query response would be detected by the resolver. If this DNS resolver also performs query name minimisation and aggressive use of NSEC records, then even the extent of any unintended information leak due to re-routed query traffic is minimised.

The observation in this presentation is that the extent of propagation of these "false" routes would've been curtailed if the root server prefixes were covered by ROA objects in the BGP routing security framework. Surprisingly, there are still four Root servers where their routes are not described in a ROA, namely E-root (AS 21556), G-root (AS 5927), H-root (AS 1508), ands L-root (AS 20144), which leaves these services exposed to various forms of potential AS origin route hijack. Of course, it must be noted that plugging this gap with a ROA is not necessarily a complete solution. There is still the potential of AS Path manipulation, which ROAs do not protect.

It also must be remembered that the overall objective is for recursive resolvers to use queries to access the contents of the root zone with some assurance of the authenticity of the response and the privacy of the transaction. It seems to me that a far better way to achieve this objective is to bypass this entire issue of potential abuse and information exposure within the network and work with a local copy of the entire root zone! RFC 8806 describes how to do this.

Monitoring Root DNS Prefixes, Lefteris Manassakis, Cisco

Post Quantum Cryptography for the RPKI

So far, we have been working in a "scalar" computing environment. What this means is that algorithms that are "linear", such as finding the prime number factors of a composite number, require twice the compute power, or twice the time when the difficulty of the problem doubles. In recent times there has been considerable interest in the development of so-called *quantum computers*. These systems exploit quantum mechanical phenomena where a unit of information is not a classical bit with a value of 1 or 0, but a *qbit* that is the superposition of its two basic states simultaneously. There are many sources of descriptions of quantum computers, so I'll not go into

any further detail here, but while there is much optimism that quantum computers will be refined in the coming years to the point where they would be able to solve significant computational challenges, the current state of quantum computers is in its very early stages. They represent the equivalent of a massively parallel processing, sometimes described as parallel processing at exponential scale. However, the engineering challenges with quantum computers are significant, and progress in engineering a quantum computer has so far been slow and extremely expensive. What's kept many projects going is the prospect that a significantly capable quantum computer could solve a range of computational challenges that are simply beyond the practical reach of scalar binary computers.

So, at some indeterminate time in the future, assuming a process of continual improvement in the capability of quantum computers, a quantum computer could out-perform a conventional computer of the day and "solve" the underlying cryptographic problem in real time. An often-cited parameter to frame this consideration is 20 years in the future, and assuming the continued functioning of Moore's law where computing capability doubles approximately every 2 years, then in 20 from now computing systems that are 1,000 times more capable than today's systems will be available. (Of course, this assumption about the continued applicability of Moore's Law over the next 20 years is a massive one, but that's not a topic for here!) Now, if you have a digital secret that you want to keep as a secret for a minimum of 20 years you can't just use an encryption algorithm and key that can withstand the assault of today's computers, but you need to think about selecting an algorithm and key size that is 1,000 times "harder" than what you may need today.

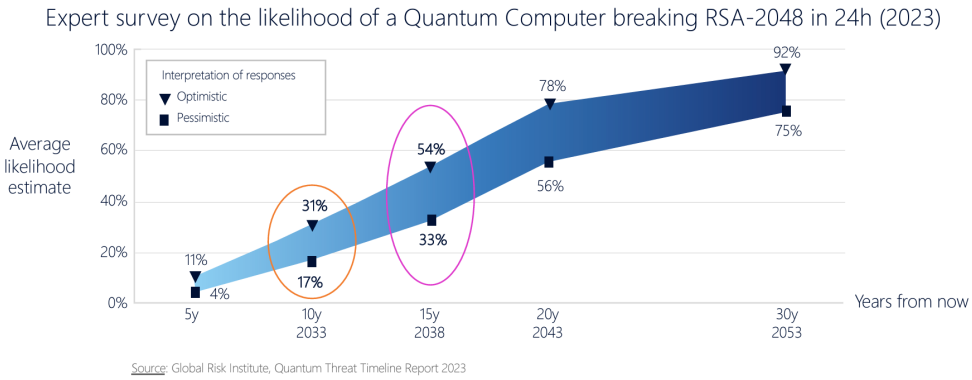

Figure 2 – Expectations of a CRQC timeline ("In-Flight Data Protection in the Quantum Age", Chris Jansen - Nokia, Presentation to NANOG 92)

Now 20 years is "interesting" in every sense in the development of quantum computing. Obviously, we can't know for sure, but there is a common expectation out there that in 20 years from now quantum computing would be able to break a RSA-2048 encrypted cyphertext. It you want to ensure that what you are encrypting today remains a secret for the next twenty years, then it may be prudent to assume that quantum computers will be used to try and break this secret sometime within that twenty-year period. So, even though realistically capable quantum computers are yet to be built, we need consider the use of quantum-resistant cryptographic algorithms today in certain areas where long-held integrity of the encryption process is important. The present danger lies in an attacker performing data capture now, in anticipation of being able to post-process it at a later date with a CRQC. There is even an acronym for this, Harvest Now, Decrypt Later (HNDL). This consideration obviously applies to the area of channel encryption, used in tools such as TLS.

But what about the DNS and DNSSEC? Is it necessarily to consider the integration of Post-Quantum Cryptography in DNSSEC today? Not really. It would be totally foolhardy to use

DNSSEC credentials with a 20-year lifetime, so we can assume that no digital signatures and the associated key values will still be in use in twenty years from now. The issue to consider is that DNSSEC is used for *authenticity*, not *encryption*. In that light, what is the rationale to justify the deployment of PQC in DNSSEC today? What is the problem where authentication of a remote party, or the authenticity of a DNS transaction that occurred today can be subverted twenty years after the original transaction that used this assertion of identity?

So far, I've not heard of an answer to that question. The only rationale I've heard is that we need to work on it now in order to "be prepared" when it will be needed, as if the advent of PQC will be a sudden and unheralded event. This seems like a weak justification to me.

Crypto credentials in PQC are larger than we've been using so far. The large increase in the size of digital signatures imply that DNSSEC using quantum-resistant algorithms over a UDP transport is not really an operationally robust choice. This would imply that DNSSEC transactions should really be supported using TCP. Using the "fallback" mechanism by firstly using a query over UDP and receiving a truncated response adds one round-trip delay to each DNSSEC-signed DNS transaction, and a further delay to establish the TCP session. Perhaps it would make more sense to combine the use of the DNSSEC-OK flag in queries to the initial use of TCP in queries, bypassing UDP altogether for such queries, but there are more considerations here. In the case of the stub-to-recursive resolver the use of a long-lived TCP session allows the application to amortise the cost of the initial TCP handshake (and more if using DOH, DOQ, or DOT) across multiple subsequent queries. In the case of recursive-to-authoritative queries the prospect of session re-use is not so great, so the overhead of session start is greater. Will it take us a further twenty years to perform this change to the DNS? I doubt it. The slow uptake of DCNSSEC is due to a poor business case, where the additional effort to sign and validate DNS material is not adequately offset by the perceived benefit. Adding to the effort side of that balance with the use of PQC only further erodes the case for the use of DNSSEC.

The presentation by Dirk Doesburg at RIPE 91 was on the topic of PQC for the RPKI. If the private keys used in the RPKI were readily discoverable using quantum computing, then it could allow the synthesis of ROAs, and other RPKI objects, that could validate BGP routes that would otherwise be provably invalid, or the synthesis of Certificate Revocation Lists that attempt to force an otherwise valid attestation to be considered invalid. But being able at some point in the future to be able to compromise the cryptographic material in use today is hardly a major issue. As in DNSSEC, it would be a foolhardy operator who used crypto material today with extended validity lifetimes of decades!

There is a visible level of academic research activity there days in quantum computing and the application of post-quantum cryptography, but I suspect that the primary motivation for this level of research activity is that the successful path through various research funding bodies these days is to make liberal use of the vocabulary of quantum computing and digital security in the research funding application! It is far less obvious to me that in most cases, and here I explicitly include RPKI and DNSSEC in this, that there is a reasonable justification for such research at present that is made on a more clinical evaluation of future needs. There is some justification in the area of digital encryption for the use of post-quantum crypto due to the expectation that the data being encrypted has ongoing value as a secret for the next couple of decades or more. However, the crypto objective in DNSSEC and RPKI is different, as they relate to the timely authentication of a remote party, or opportunistic encryption of a time-bounded transaction, so it is far more challenging to understand the rationale for heading into this aspect of post-quantum cryptography right now. And even if we get to build a realistically capable quantum computer in the future, if it's cost to use it is eye-wateringly large, then it's use will only be justifiable in high-end esoteric

areas and our future selves may well regard today's enthusiasm for shifting our attention to the post-quantum risk of real-time generation of synthetic DNS answers or false RPKI credentials as simply jumping too early!

PQC for the RPKI Dirk Doesburg, Radboud University

IPv6 Scanning

I've recently set up a DNS authoritative server on a public IP address. It serves a private DNS domain and is not referenced as a name server in any part of the public DNS framework. Yet within a minute or so I'm receiving DNS queries on UDP port 53 for all kinds of domain names. It's all unsolicited noise of course, and irrespective of whether the motives here are to discover and measure services that network owners and operators have publicly exposed as part of some data gathering exercise, or if their motives are to locate exploitable servers, its still unsolicted noise. And there is a lot of it, and it's getting worse (https://www.potaroo.net/dark/bps.pdf). There's enough of this toxic unsolicited scanning that if this was background radiation it would be vivid throbbing light in the dark! But it's pretty much all IPv4 traffic. Why?

It's a size thing. IPv4 has a total of some 4 billion end point addresses. If you could scan some 50,000 IP addresses second, then it would take around one day to scan the entire IPv4 address space. But if you used the same scanning engine and applied it to the IPv6 network, then a full scan at this rate would take far longer than the age of the universe!

The way we defend against scanning attacks in IPv4 is with stateful NATs. Incoming packets are only admitted if they use the same IP address, the same port numbers and the same transport protocol as a recent matching outgoing packet. Casual scanning of the IPv4 space only discovers services that are not located behind stateful NATs. But stateful NATs have a poor reputation among IPv6 proponents. A large party of the motivation of the IPv6 effort was to regain a coherent end-to-end address space in the IP architecture. The deployment of IPv6 was intended to preclude the inclusion of IPv6 NATs. IPv6 hosts were to use public IPv6 addresses on the public IPv6 Internet. What "protected" these hosts from casual discovery? The very large size of the IPv6 address space was intended to prevent scanners from operating in IPv6. That and stateful firewalls!

However, it has left an open question on the table that as proved to be simply too attractive for many hackers, irrespective of their good or bad motivations. Rather than blindly stepping through the IPv6 address space, is it possible to use some guidance to narrow the search space.

In the original IPv6 address structure model (RFC2464) the IPv6 address is space is divided in half, and the least significant 64 bits (the interface identifier) is build using the 48-bit interface MAC address, expanded by a constant 16-bit fill pattern.

The dramatically reduces the size of the search space. The weaknesses of this approach were quickly apparent, and the IETF pushed out a number of refinements in succeeding years, including RFC 494, using a random value, RFC3972, using a crypto algorithm, RFC7271, also using a random generator and RFC8981, using a temporary random value.

There are a number of helpful sources to further narrow the search space. TLS certificates are helpful, and the certificate transparency logs are a rich source of domain names, and the DNS can be used to map these names to IP addresses. There have been related studies using similar forms of indirect hints. But why go to all this trouble when someone else has already gone before and

loaded it into github? The resource https://ipv6hitlist.github.io maintains a list of some 3.6B reachable IPv6 host addresses. It's hard for me to see this as anything other than a hostile resource!

In looking at the profile of unsolicited IPv6 traffic, Matsuzaki Yoshinobu observed a preponderance of HTTP-related TCP probes using a EUI-64 interface ID pattern, with addresses that have been entered in the ipv6hitlist. What can prevent IPv6 addresses being added to this hitlist are to avoid responding to unsolicited traffic (which is challenging if the host is an IPv6 server), to use IPv6 privacy addresses, or to use a stateful firewall to block all unsolicited incoming traffic that does not match the host's service profile if it is a server, or just use an IPv6 NAT as a front end!

A Day in the Life of IPv6 Scanning Yoshinobu Matsuzaki, Internet Initiative Japan Inc. (IIJ)

A BCP on Hyperlocal Default

I guess all groups develop their own terminology over time as an expression of tribalism. For those who are not part of the DNS nerd tribe, this heading translates to a proposal for DNS recursive resolvers to pre-fetch the entire root zone as a default practice. As Jim Reid noted, the reasons to do so are certainly compelling, in that it improves the performance of the DNS, it improves the privacy profile of the DNS, reduces the DOS attack surface and defuses some of the more troublesome political governance and control questions about the operation of the DNS (as anyone who want to integrate the root zone into their DNS resolver instance can do so). It also shifts a component of the support burden of the provision of the root zone service from a dozen entities who provide this service for free back to the millions of recursive resolvers who can substitute a huge volume of external root queries with a single zone transfer.

What surprises me is that this concept has been around for more than a decade in one form or another without any shift in the convential modes of DNS resolver behaviour. For a group working supposedly at the high frontier of technology we can behave in surprisingly conservative ways at times!

A BCP on Hyperlocal Root Jim Reid

TTLs in the DNS

As a massively distributed database the DNS would've collapsed under an overwhelming query load decades ago were it not for the use of local caching. Caching allows the local resolver to maintain a copy of a response in its local cache and reuse it for subsequent queries for the same query name and record type. Caching represents a compromise between efficiency and timeliness of the information.

This becomes an issue when a DNS operator wishes to alter the contents of a DNS record. The change will only take effect across the entire DNS once the old information has been flushed from every resolver's cache. (This can be an issue, such as the Slack outage a couple of years ago, when resolvers cached an assertion that the slack.com DNS name had no address records, and the outage extended for 24 hours because of cache retention.)

The DNS uses the concept of a "time to live" (TTL) to allow the zone administrator to specify how long a record from the zone should be retained in a resolver's cache. The TTL value is a count of seconds, using an unsigned 31-bit field. Values can be 0 seconds through to 68 years! But TTL values are not a firm rule. They are more of a guideline. Recursive resolvers may not cache at all, or they can impose their own local retention lifetime upon a cached resource record. More commonly, resolvers impose their own maximum TTL on a DNS record, as very long cache

lifetimes have memory overheads and can force the resolver to hold stale information. This applies to minimum cache lifetimes well, in order to mitigate potential denial of service attacks through *query thrashing* when a 0-second TTL is provided that in effect is a "do not cache" directive.

Shane Kerr's presentation used an unsigned wildcard record with a maximum TTL and then used a set of RIPE Atlas to lookup a unique name twice, with the first lookup intended to load the record into the local resolver's cache and the second intended to replay the cached value and expose the cache TTL used by the resolver back to the Atlas probe. His results show that DNS resolvers generally impose a maximum cache time on cached records, with the most common cache timers being 24 hours, 6 hours, 12 hours and 7 days. A small set of resolvers were observed to impose a time of 10 seconds or less, and a small set appeared to honour the 68 year TTL value! On the popular open DNS resolvers, Cloudflare's 1.1.1.1 uses the provided TTL, Google's 8.8.8.8 service imposes a 6 hour maximum cache lifetime, PCH's 9.9.9.9 uses 12 hours and Cisco's OpenDNS uses 7 days.

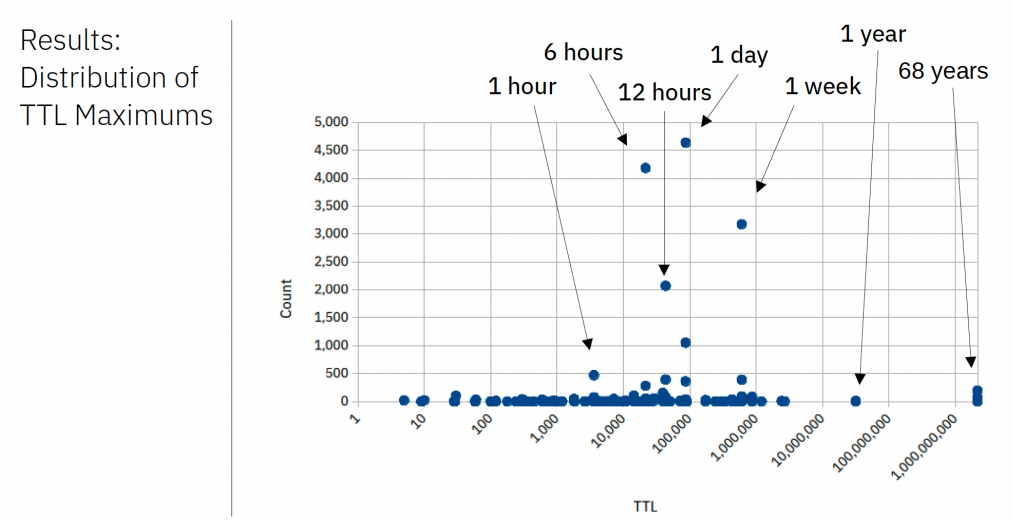

Figure 3 – Anomalous Origination L-Root Prefix (from DNS TTL Upper Limits in Practice)

TTLs in the DNS are more like suggestions than rules, and the longer the longer the proposed TTL value the greater the likelihood that the resolver will impose its own shorter value.

There is an earlier (2011) study of TTL treatment for short TTLs in the DNS by Frank Denis can be found at https://00f.net/2011/11/17/how-long-does-a-dns-ttl-last/ which exposes a comparable variation in cache retention behaviours.

DNS TTL Upper Limits in Practice Shane Kerr, IBM NS1 Connect

BGP Max Prefix Limit

> "On 8 November 2023, the the Australian provider Optus network experienced a nation-wide outage affecting their fixed phone, internet and mobile services (the outage). The outage also impacted CSP resellers of Optus' network. The outage commenced around 4am AEDT and continued for approximately 12 hours, with services being restored by 4pm AEDT the same day." Australian ACMA Report, 2023.

It appears that the base problem was a route leak in BGP that overwhelmed some routers in this provider's network and the default response to the situation of learning more routes than it was

configured to learn was to shutdown the routing session. This is a common behaviour, and it is as much to prevent the propagation of route leaks as it can be a self-protection mechanism to prevent local router's memory structures from overflowing and crashing the router.

In the case of this Optus outage, the BGP links in question were apparently the only links between the only onshore domestic platform and the offshore technical support, so when the sessions were automatically shut down there was evidently no way that the network's support engineers could access the routers and repair the situation.

The question asked in this presentation was how common is this situation?

The approached described was to use the maximum prefix field in the PeeringDB entry and compare this to the visible peers and the count of prefix that they advertise, as seen by the RIPE RIS route collectors (https://ris.ripe.net/docs/route-collectors/#peer-meta-data). Of the 88,000 ASes seen by the route collectors, one third of them have PeeringDB entries, and half of these have updated those entries in the past two years. Their investigation found that only a relatively small number of ASes exceeded their stated maximum prefix value, and of those far fewer are observed to drop their BGP sessions. A likely reason is that the actual maximum prefix count on the routers may differ from the value recorded in the PeeringDB system.

It has been a protracted discussion as to whether it's "better" to drop a session when the total prefix count exceeds some configured limit or just to freeze the session and accept no further prefix announcements until the prefix count drops below some recovery threshold value. Freezing a session can cause traffic loops which becomes a much larger problem, whereas dropping a session has its own repercussions, as Optus (and many others) have found out.

In a service-critical environment, which is what the Internet now is, dropping a routing session is no longer a casual matter, and should be treated as a last resort response. But long-held sessions have their own problems, not the least of which is stuck routes. Many implementations of BGP support Graceful Restart (RFC4724), allowing the BGP state to be refreshed without pulling down the previous packet forwarding state. But if the problem is a route flood from a BGP peer, graceful restart can't fix that. So, should we dispense completely with this maximum prefix limit? No. Because is the choice is between bringing down just one BGP session due to a route leak-induced session overload or having the entire router crash due to a memory exhaustion problem, then losing the session but keeping the router up is the far better outcome. "Hope for the best, but prepare for the worst." is relevant advice in using maximum prefix limits in BGP routers.

What happens when you overshare? A Look into the BGP Maximum-Prefix Feature, Orlando Eduardo Martínez-Durive, NetAI & IMDEA Networks, Antonios Chariton, Cisco

Stuck Routes in BGP

What happens when the BGP engine is not perfect in its operation? What happens if a BGP neighbour sends a withdrawal about a prefix and the local BGP router manages to drop this information rather than process it? The basic answer is nothing of much consequence happens. At its worse it might result in a routing loop for this prefix, and at best a deferred packet drop. It was long suspected that BGP route entries might get stuck in routers in this manner, but it appears that this topic was not widely noted until Gert Doring started doing his regular report on the state of the IPv6 routing table at RIPE meetings in Europe in the early 2000's. For example, in his report to RIPE 42 in April 2002, Gert reported on an IPv6 "zombie" route he had found in the IPv6 routing table. At the time there were just some 300 IPv6 routes in total, so it was feasible to examine the entire route set manually and identify these stuck routes with some certainty! It was

unclear at the time if this was an isolated situation, or one that was related to the relatively little-used code paths of IPv6 support in routers at that time.

Researchers have returned to this topic at regular intervals, setting up short term route advertisements and looking for prefixes that are still visible in some of the routing tables after their scheduled withdrawal. It appears that this concept of "sticky" routes is a relatively constant attribute of the BGP routing system, visible in both the IPv4 and IPv6 routing tables. One form of research has been to try and trigger the sticky behaviour, but the problem is how can you tell if a route is stuck or not? An inventive solution to generating potential stuck routes was found in IPv6, where the time a route was advertised was embedded in the IPv6 route prefix, and the route was advertised only for 15 minutes, to be replaced by the next prefix.

In this presentation Thousand Eyes presented the BGP Stuck Route Observatory, based on a larger IPv6 prefix pool that allows for a longer quiescent period before each prefix is readvertised. It's still early days in this work and there is much more in the way of analysis to understand how to isolate anomalous individual behaviours in the collected data, but the works looks like it has promise!

The BGP Clock and the BGP Observatory: Hunting Stuck Routes Antonios Chariton, Cisco

Columnar Databases for BGP Data

I've grown very used to row-oriented databases, perhaps because of their natural match to structured data used in programming languages. Row-oriented databases support random access models where one or more rows are selected based on a match of a field against a query value, and all the row's fields are accessible. The typical use of a column-oriented database is to compute aggregate values on a limited number of columns, as opposed to try and retrieve all/most columns for a given entity.

RIPE NCC'S Ties de Kock presented on work to load BGP RIB snapshot and BGP update in column-orientation, using Amazon Parquet. Parquet only loads the selected data in a query, and can perform an analysis in a shorter time interval. It's not clear if this is a useful data format for BGP analysis, so the RIPE NCC will publish BGP data in this column-oriented format for the next 18 months, and then gather feedback as to continue with this publication format going forward.

Querying the DFZ, Ties de Kock, RIPE NCC

Is that the Right Time?

The Network Time Protocol (NTP) has been around for many years. The original specification for this protocol, RFC985 was published in September 1985. The most recent specification of NTP, version 4, is RFC5905, published in 2010. Authentication and channel security was added to NTP in 2020 (SNTP) with RFC8915, allowing a client to receive time from the server in an authenticated and secure manner.

A study by students at TU Delft looked at a number of popular NTP client implementations and their behaviour. There is a wide variation in behaviours, with one client, NTPD-RS performing some 250 exchanges in an hour, while the SMTP clients operate with a far lower transaction rate. When configured with multiple time servers, NTP clients normally query all servers to establish a reference point, avoiding the potential of being misled by a single rogue NTP server, with the exception of the TimesyncD clients, which latches onto a single server. If an attacker can seize control of an NTP server and provide a shifted time it is possible to mislead clients. All STNP

clients were shown to be vulnerable to such time shifting attacks, while none of the NTP clients were vulnerable. Interestingly the Default NTP clients for MAC OS, Ubuntu, Debian and Windows were shown to be vulnerable to tome shifting attacks, some to the level of a shift by some years.

It would be reassuring to learn that this client-side vulnerability has been recognized, and patches have been devised and deployed. As is common in a diverse environment the picture is not so clear, with only some implementations reacting to this vulnerability with timely fixes.

Are NTP Clients Always Right? Shreyas Konjerla, TU Delft

Apple Wireless Woes

There is a hidden cost to "seamless" wireless connectivity in the user environment, as explained by Cristoff Visser. The observed problem was regular stuttering in a streaming application using WiFi.

The issue is that in order to perform seamless connectivity within the Apple ecosystem, they use the Apple Wireless Direct Link (AWDL) protocol which uses WiFi Channels 6, 44, and 149 to periodically advertise its availability and negotiate a WiFi channel for data. The implication is that if your streaming WiFi service is using a channel other than these three channels, then the conventional connection service will be interrupted to allow the WiFi to channel switch to one of these AWDL channels and perform an AWDL advertisement and then switch back. This causes streaming stuttering.

The fix is surprisingly simple. Just configure your WiFi to use one of these channels by default, and then there is no need to interrupt the data session to channel hop to perform the AWDL functions. The change can be quite dramatic, and Cristoff reported a stable speed increase from 50Mbps to 105Mbps when using one of these channels!

Apple Wireless Direct Link: Apple's Network Magic or Misery Christoff Visser, IIJ - Research Lab

RIPE 91

This is a small sample of the material presented at RIPE 91. It is heartening to see a strong interest from researchers in looking behind the covers to expose the behaviour of the myriad of component systems that support the Internet's service environment, looking at both current issues and potential opportunities to improve.

The presentations and recordings are all liked from the conference's web page: https://ripe91.ripe.net/programme/meeting-plan/sessions/

Disclaimer

The above views do not necessarily represent the views or positions of the Asia Pacific Network Information Centre.

Author

Geoff Huston AM, M.Sc., is the Chief Scientist at APNIC, the Regional Internet Registry serving the Asia Pacific region.

www.potaroo.net