The Internet Protocol Journal

March 2010 Volume 13, Number 1

A Quarterly Technical Publication for Internet and Intranet Professionals

In This Issue

From The Editor

Previous articles in IPJ have described *Domain Name System Security Extensions* (DNSSEC), the security system for the *Domain Name System* (DNS). DNSSEC introduces security into the DNS through the use of cryptographic keys and digital signatures. Interest in DNSSEC has grown in recent months, as the *Internet Corporation for Assigned Names and Numbers* (ICANN) and VeriSign have undertaken a phased program to deploy DNSSEC across the root server system in the first half of 2010. In an article by four DNS practitioners, we will explore some side effects of DNSSEC, and examine what happens in two widely used DNS resolver implementations when DNS clients lag behind in synchronizing their local copy of trust keys with the master keys used by the zone administrators to sign their DNS data.

Several articles in IPJ have dealt with various concerns related to scaling of the Internet. In this issue, Paul Francis and Xiaohu Xu describe *Virtual Aggregation*, a new routing technology being developed by the GROW working group of the IETF to reduce the size of the *Forwarding Information Base* (FIB) held in memory by routers.

The *Request For Comments* (RFC) Series has been the main publication channel for Internet standards and related documents for more than 40 years. The RFC Editor function is in the process of being restructured and moved from its original home at the *University of Southern California Information Sciences Institute* (USC/ISI). Leslie Daigle describes the history and future of the RFC Editor mechanism.

If you are reading this online and did not receive the March 2010 edition of IPJ, it may be because your subscription has expired. You can still renew your subscription by visiting the "Subscriber Services" section of our webpage at `www.cisco.com/ipj`. Enter your subscription ID and e-mail address to gain access to your subscription record. If you don't know your subscription ID or have changed e-mail address recently, just send a message to `ipj@cisco.com` and we will take care of the renewal and update for you.

—Ole J. Jacobsen, Editor and Publisher
`ole@cisco.com`

You can download IPJ back issues and find subscription information at:
`www.cisco.com/ipj`

ISSN 1944-1134

Rolling Over DNSSEC Keys

by George Michaelson, APNIC, Patrick Wallström, .SE, Roy Arends, Nominet, Geoff Huston, APNIC

As we are constantly reminded, the Internet can be a very hostile place, and public services are placed under constant pressure from a stream of probe traffic, attempting to exploit any one of numerous vulnerabilities that may be present at the server. In addition, there is the threat of *Denial of Service* (DoS)[1] attacks, where a service is subjected to an abnormally high traffic load that attempts to saturate and take it down. This story starts with the detection of a possible hostile DoS attack on *Domain Name System* (DNS) servers, and narrates the investigation as to the cause of the incident, and the wider implications of what was found in this investigation.

Detecting the Problem

The traffic signature in Figure 1 is a typical signature of an attempted DoS attack on a server, where the server is subjected to a sudden surge in queries. In this case the traffic log is from a secondary DNS Name Server that is authoritative for a number of subdomains of the **`in-addr.arpa`** zone; the traffic surge shown here commenced on December 16, 2009. The traffic pattern shifted from a steady state of some 12 Mbps to a new steady state of more than 20 Mbps, peaking at 30 Mbps.

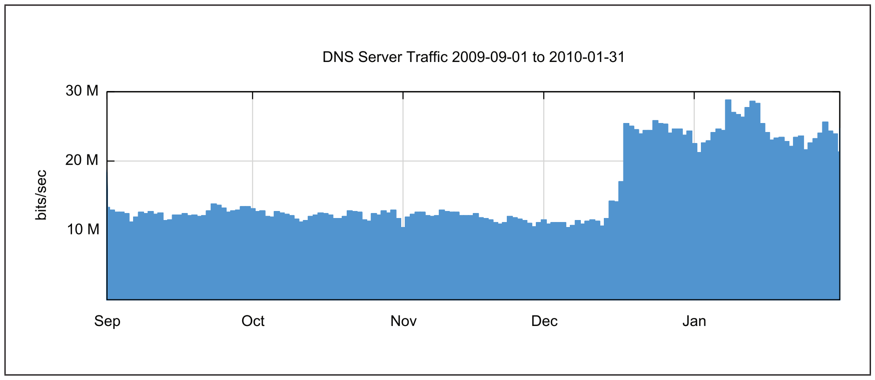


Figure 1: Traffic Load for ***`in-addr.arpa`*** *Server (provided by George Michaelson)*

Because the traffic shown in Figure 1 is traffic passed to and from a Name Server, the next step is to examine the DNS traffic on the Name Server, and in particular look at the rate of DNS queries that are being sent to the Name Server (Figure 2). The bulk of the additional query load is for DNSKEY *Resource Records* (RRs), which are queried as part of the operation of *Domain Name System Security Extensions* (DNSSEC)[2].

Because this zone is a DNSSEC signed zone, DNSKEY queries will cause the server to respond with a DNSKEY RR and the related RRSIG RR in response to each query. This pair of RRs generates a response that is 1,188 bytes in this case. At a peak query rate of some 3,000 DNS queries per second, a traffic response from the server in excess of 35 Mbps will be generated.

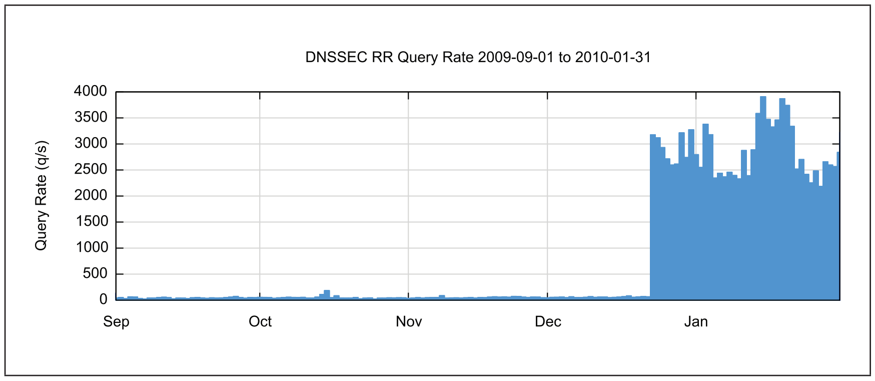


Figure 2: Query Rate for ***`in-addr.arpa`*** *Server (provided by George Michaelson)*

There are many possibilities as to what is going on here:

- This problem could be caused by a DoS attack directed at the server, with the attacker attempting to saturate the server by flooding it with short queries that generate a large response.
- This problem could be caused by a DNS reflection DoS attack, where the attacker is placing the address of the intended victim or victims in the source address of the DNS queries and attempting to overwhelm the victim with this DNS response traffic.

Although it is good to be suspicious, it is also useful to remember the old adage that we should be careful not to ascribe to malice what could equally be explained by incompetence, so numerous other explanations should also be considered, including:

- This problem could be a DNS resolver problem, where the resolver is not correctly caching the response, and some local event is triggering repeated queries.
- This problem could be a bug in an application where the application has managed to wedge itself in a state of rapid-fire queries for DNSKEY RRs.

The next step is to examine some of these queries more closely, and, in particular, look at the distribution of query source addresses to see if this load can be attributed to a small number of resolvers that are making a large number of queries, or if the load is spread across a much larger set of resolvers. The server in question typically sees on the order of 500,000 to 1,000,000 distinct query sources per day.

Closer inspection of the query logs indicates that the additional load is coming from a relatively small subset of resolvers, on the order of 1,000 distinct source addresses, with around 100 "heavy hitters." In other words, all this DNS traffic is being generated by some 0.01% of the DNS clients. The sequence of queries from one such resolver that is typical of the load being imposed on the server is shown in Figure 3.

Figure 3: DNS Query Sequence Packet Capture

1. Client requests the Delegation Signer (DS) records for the `211.89.in-addr.arpa` zone.
2. Reply says "no such delegation," and sends *DNSSEC Signature* (RRSIG) and *Next-Secure record* (NSEC) from the parent zone, and surrounding records.
3. Client requests DNSKEY from the parent zone.
4. Server sends DNSKEY and RRSIG set for the parent zone.

Having established an initial query state and the DNSKEY and signature set over the original request, the client then paradoxically repeatedly re-queries the parent-zone DNSKEY state. This process is elided as follows because the query and response do not differ during this exchange:

5. Client repeats the DNSKEY request.
6. Server repeats the DNSKEY and RRSIG response.
7. Client repeats the DNSKEY request.
8. Server repeats the DNSKEY and RRSIG response.

This exchange of DNSKEY request and DNSKEY and RRSIG response is repeated a further 6 times.

If this additional query load had appeared at the server over an extended period of time, it would be possible to ascribe this problem to a faulty implementation of a DNS resolver, or a faulty client application. However, the sudden onset of the additional load tends to suggest that something else is happening. The most likely explanation is that some external "trigger" event exacerbated a latent behavioral bug in a set of DNS resolver clients. And the most likely external trigger event is a change of the contents of the zones being served.

So we can now refine our set of possible causes to concentrate consideration on the possibility that:

- Something changed in the zones being served by this secondary server that triggered a pathological query response from a set of resolvers.

And indeed the contents of the zones did change on the day when the traffic profile changed, with a key change being implemented on that day.

DNSSEC Key Management

It is considered good operational practice to treat cryptographic keys with a healthy level of respect. As RFC 4641[3] states: "The longer a key is in use, the greater the probability that it will have been compromised through carelessness, accident, espionage, or cryptanalysis." Even though the risk is considered slight if you have chosen to use a decent key length, RFC 4641 recommends, as good operational practice, that you "roll" your key at regular intervals. Evidently it is a popular view that fresh keys are better keys.

The standard practice for a "staged" key rollover is to generate a new key pair, and then have the two public keys coexist at the publication point for a period of time. This practice allows relying parties, or clients, some period of time to pick up the new public key. Where possible during this period, signing is performed twice, once with each key, so that the validation test can be performed using either key. After an appropriate interval of parallel operation, the old key pair can be deprecated and the new key can be used exclusively for signing.

This key rollover process should be a routine procedure, without any intended side effects. Resolvers that are using DNSSEC should refresh their local cache of zone keys in synchronization with a published schedule of key rollover, and ensure that they load a copy of the new key within the period when the two keys coexist. In this way when the old key is deprecated, responses from the zone servers can be locally validated using the new key.

The question here is why did this particular key rollover for the signed zone cause the traffic load at the server to spike? And why is the elevated query rate sustained for weeks after the key rollover event? The key had changed 6 months earlier and yet the query load prior to this most recent key change was extremely low.

DNSSEC DNS Resolver Behavior with Outdated Trust Keys

It is possible to formulate a theory as to what is going on from this collection of information. It could be that one or more DNS resolver clients has been using a local *Trust Anchor* that has been manually downloaded from the zone administrator prior to the most recent key rollover, but has not been updated since. When the key rollover occurred in December 2009, these clients could no longer validate the response with their locally stored Trust Anchors.

Upon detecting an invalid signature in the response, the client appears to have reacted as if there were a "man-in-the middle" injection attempt, and immediately repeated the request in an effort to circumvent the supposed attack by rapidly repeating the query. If this instance were really a man-in-the-middle injection attack, this response would be plausible, because there is the hope that the query will still reach the authoritative server and the client will receive a genuine response that can be locally validated.

Why does the client really perform this repeated query pattern? In this case the contributory factor is the use of multiple name servers in the DNS. When the DNS client performs a key validation, it performs a bottom-up search to establish the trust chain from the initial received query to a configured Trust Anchor.

Example DNSSEC Validation

As a hypothetical example, assume a TXT RRset for **`test.example.com`** in a signed **`example.com`** zone. The zone **`example.com`** resides on two Name Server addresses. The **`example.com`** zone has a *Key Signing Key* (KSK), which is referred to by the DS record in the **`.com`** zone. The **`.com`** zone is signed, and it resides on 14 addresses (11 IPv4 and 3 IPv6). The **`.com`** zone has a KSK, which is referred to by a Trust Anchor in the local configuration of the resolver (Figure 4).

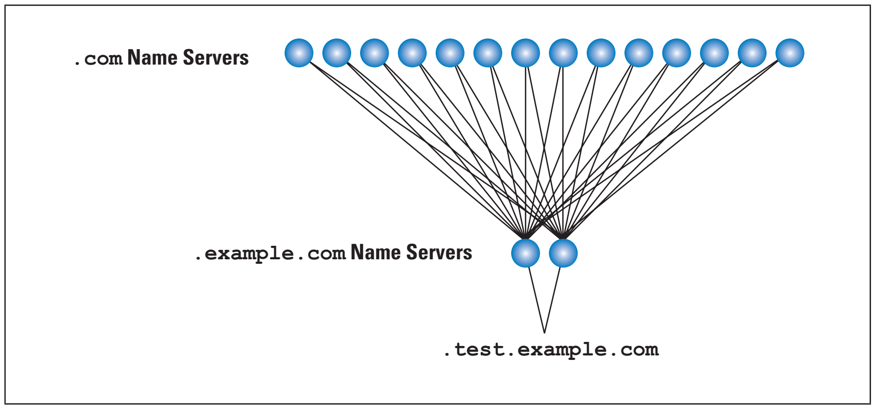


Figure 4: Example Configuration

Assume that the locally held Trust Anchor for **`.com`** in the resolver has become stale. That is, the DS record for **`.com`** in the root zone validates, but there are no DNSKEYs in **`.com`** that match the DS record in the root zone.

When a client is resolving a query relating to **`test.example.com`**, the following search occurs:

- *Berkeley Internet Name Domain* (BIND)[9] resolves the **`test.example.com`** RRset. It attempts to validate it. To do so, it needs the **`example.com`** DNSKEY RRset.
- It resolves the DNSKEY RRset for **`example.com`** from a Name Server of **`example.com`**. It attempts to validate it. To do so, it needs the **`example.com`** DS RRset.
- It resolves the DS RRset for **`example.com`** from a Name Server of **`.com`**. It attempts to validate it. To do so, it needs the **`.com`** DNSKEY RRset.
- It resolves the DNSKEY RRset for **`.com`** from a Name Server of **`.com`**. It attempts to validate it with the locally configured Trust Anchor.

However, the resolver cannot validate the **`.com`** DNSKEY RRset because it does not have the proper Trust Anchor for it. It queries all remaining 13 **`.com`** servers for the DNSKEY RRset for **`.com`**. Then the resolver still does not have the proper **`.com`** DNSKEY, and tracks back one level:

- It resolves the DS RRset for `example.com` from the next authoritative Name Server. It attempts to validate it. To do so, it needs the `.com` DNSKEY RRset. The search goes forward again.
- It resolves the DNSKEY RRset for `.com`. It attempts to validate it with the locally configured Trust Anchor.

Because the DNSKEY RRset for `.com` has not changed, this attempt will fail as well.

The complete in-depth first search consists of:

- TXT records on 2 `example.com` servers, signed by:
- DNSKEY records on 2 `example.com` servers, referred to by:
- DS records on 14 `.com` servers, signed by:
- DNSKEY records on 14 `.com` servers.

When all possible paths are exhausted, the client will have sent the following:

- 784 (2 × 2 × 14 × 14) `.com` DNSKEY requests to 14 `.com` Name Servers
- 56 (2 × 2 × 14) `example.com` DS requests to 14 `.com` Name Servers
- 4 (2 × 2) `example.com` DNSKEY requests to 2 `example.com` Name Servers

In other words, in this example scenario with stale Trust Anchor keys in a local client's resolver, a single attempt to validate a single DNS response will cause the client to send a further 844 queries, and each `.com` Name Server to receive 56 DNSKEY RR queries and 4 DS RR queries.

The breadth and level of the search is important here, because the longer the validation chain and the more the number of authoritative Name Servers for those zones that lie on the validation chain path, the more queries that will be sent in an effort to validate a single initial response. In this example, the level of search is three deep, and terminates at `.com`. If the `.com` zone were signed by the root Name Servers and the client were using a stale root zone key, then the 20 distinct root zone server addresses (13 in IPv4 and 7 IPv6 addresses) would also be queried:

- 313,600 (2 × 2 × 14 × 14 × 20 × 20) root DNSKEY requests to 20 root Name Servers
- 15,680 (2 × 2 × 14 × 14 × 20) `.com` DS requests to 20 root Name Servers

It is worthwhile noting in this context that reverse trees and enum trees in the `.arpa` zone are longer on average. Though delegations in those subtrees might span several labels, it is not uncommon to delegate per label. Note also that the entire effort is done per incoming query—the entire search is repeated for each query.

Though this example shows an enormous query load, there are a few ceilings. In commonly used validating resolvers, such as BIND 9.7rc2, every search is performed in serial, and each search is halted after 30 seconds.

The *Unbound* client[4] also appears to have a similar request behavior, although it is not as intense because of the cache management in this implementation. Unbound will "remember" the query outcome for a further 60 seconds, so repeated queries for the same name will revert to the cache. But the DNSSEC key validation failure is per zone, and further queries for other names in the same zone will still exercise this re-query behavior. In effect, for a zone that has sufficient "traffic" of DNS load in subzones or instances inside that zone, the chain of repeated queries is constantly renewed and kept alive.

If one such client failed to update its local trusted key set, then the imposed server load on DNSSEC key rollover would be slight. However, if a larger number of clients were to be caught out in this manner, then the load signature of the server would look a lot like Figure 2. The additional load imposed on the server comes from the size of the DNSKEY and RRSIG responses, which are 1,188 bytes per response in the specific failure case that triggered this investigation.

So far we've been concentrating attention on the `in-addr.arpa` zone, where the operational data was originally gathered. However, it appears that this problem could happen to any DNSSEC signed domain where the zone keys are published so as to allow clients to manually load them as trust points, and where the keys are rolled on a regular basis.

It is likely that one possible cause for this situation is in the way in which some DNSSEC distributions are packaged with operating systems. For example, the *Fedora*[5] Linux distribution has bundled numerous trust keys with its packaging of a DNS resolver client and local Trust Anchor key set. When the keys associated with sub zones of `in-addr.arpa` rolled over in December 2009, users of this version of the Fedora Linux distribution would have been caught with stale trust keys.

So there appears to be a combination of three factors that are causing this situation:

- The use of prepackaged DNSSEC distributions that included preloaded keys in the distribution
- The use of regular key rollover procedures by the zone administrator
- Some implementations of DNS resolvers that react aggressively when there is a key validation failure by performing a rapid sequence of repeat queries, with either a very slow, or in some cases no apparent back-off in query load

This combination of circumstances makes the next scheduled key rollover for **`in-addr.arpa`**, scheduled for June 2010, appear to be quite an "interesting" event. If there is the same level of increase in use of DNSSEC with manually managed trust keys over this current 6-month interval as we've seen in the previous 6 months, and if the same proportion of clients fails to perform a manual update prior to the next scheduled key rollover event, then the increase in the query load imposed on **`in-addr.arpa`** servers at the time of key rollover promises to be truly biblical in volume.

Signing the DNS Root

There is an end in sight for this situation for the subzones of **`in-addr.arpa`**, and for all other such subzones that currently have to resort to various forms of distribution of their zone keys. The *Internet Corporation for Assigned Names and Numbers* (ICANN) has announced that on July 1, 2010, a signed root zone for the DNS will be fully deployed[6]. Assuming that the **`.arpa`** and **`in-addr.arpa`** zones will be DNSSEC-signed in a similar time frame, the situation of escalating loads being imposed on the servers for delegated subdomains of **`in-addr.arpa`** at each successive key rollover event will be curtailed. It would then be possible to configure the client with a single trust key, the public key signing key for the root zone, and allow the client to perform all signature validation without the need to manually manage other local trust keys.

There are two potential problems with this scenario.

The first is that for those clients that fail to remove the local Trust Anchor key set, these repeated queries may not go away. When there are multiple possible chains of trust, the resolver will attempt to validate using the shortest validation chain. As an example, if a client has configured the DNSKEY for, say, **`test.example.com`** into its local Trust Anchor key set, and it then subsequently adds the DNSKEY for **`example.com`**, the resolver client will attempt to validate all queries in **`test.example.com`** and its subzones using the **`test.example.com`** DNSKEY.

A more likely scenario is where an operator has already added local Trust Anchor keys for, say, **`.org`** or **`.se`**. When the root of the DNS is signed, the operator may also add the keys for the root to the local Trust Anchor set. If the operator fails to remove the local copies of the **`.org`** and **`.se`** Trust Anchor keys, in the belief that this root key value will override the **`.org`** and **`.se`** local keys, then the same validation failure behavior will occur. In such a case, when the local keys for these second-level domains become stale, their resolver will exhibit the same re-query behavior, even when they maintain a valid local root Trust Anchor key.

As a side note, the same behavior may occur when *DNSSEC Lookaside Validation* (DLV) is used. If the zone key management procedures fall out of tight synchronization with the DLV repository, it is possible to open a window where the old key remains in the DLV repository, but is no longer in the zone file. This situation can lead to a window of vulnerability where the keys in the DLV repository are unable to validate the signed information in the zone file, a situation that, in turn, introduces the same problem with re-query.

The second potential problem lies with the phase-in approach of signing the root. The staged rollout of DNSSEC for the root zone envisages a sequenced deployment of DNSSEC across the root server clusters, and through this sequence the root will be signed with a key that has no valid published public part, creating a *Deliberately Unvalidatable Root Zone* (DURZ).

What happens when a client installs this key in its local Trust Anchor set and performs a query into the root zone?

As an experiment, this DURZ key was installed into an instance of BIND 9.7.0rc2, with a single upstream root, pointing at the "L" root, the only instance of the 13 authoritative root servers enabled with DNSSEC signed data in February 2010. On startup the client made 13 consecutive DNSKEY requests, one to each of the root zone server addresses. When the client started its first query in a subzone, the client issued a further 156 DNSKEY queries in a period of 19 seconds, making 12 queries to each of the 13 root zone server addresses.

This scenario should sound familiar, because it is precisely the same query pattern as happened with the **`in-addr.arpa`** servers and the **`.se`** servers, although the volume of repeated DNSKEY queries is somewhat alarming. When the client receives a response from a subdomain that needs to be validated against the root, and when the queries to the root are not validatable against the local trust key, the client goes into a sequence of repeated queries that explore each potential validation path. Anchoring the local resolver with a key state that invalidates the signatures of all authoritative servers of the zone—but authoritatively (absent DNSSEC) confirms them as valid servers of the zone—places the client instance in an unresolvable situation: no authoritative Name Server that it can query has a signature that the client can validate, but the root zone informs it that only these Name Servers can be used.

Further tests of this behavior show that the client does not cache the outcome that the DNSKEY cannot be validated for a zone, and the client reinitiates this spray of repeated queries against the zone Name Servers when a subsequent DNSSEC query is made in a subzone. Therefore the behavior is promiscuous in two distinct ways. First it is evident that any Name Server so queried is repeatedly queried. Second, it is evident that all Name Servers of a zone are queried. The other part of the client response is not to cache validation failure for the zone in case this repeated query phase does not provide the client with a locally validated key.

After all, the data is provably false, so caching it would be to retain something that has been "proven" to be wrong.

The emerging picture is that misconfigured local trust keys in a DNS resolver for a zone can cause large increases in the DNS query load to the authoritative Name Servers of that zone, where the responses to these additional queries are themselves large, of the order of 1,000 bytes in every response. This situation can occur for any DNSSEC signed zone.

The conditions for the client to revert to a rapid re-query behavior follow:

- The *DNSSEC OK* (DO) bit is honoured by the server.
- The DNS data appears to be signed.
- The signature check fails.
- The client does not cache the validation failure for this zone.

The conditions being set up for the DURZ approach for signing the root follow:

- The DO bit is honoured by the server.
- The DNS data appears to be signed.
- The signature check fails.
- The client does not cache the validation failure for this zone.

What is to stop the DNS root servers from being subjected to the same spike in the query load?

The appropriate client behavior for this period of DNSSEC deployment at the root is not to enable DNSSEC validation in the resolver. Although this advice is sound, it is also true that many resolvers have already enabled validation in their resolvers, and are probably not going to turn off for the next 6 months while the root servers gradually deploy DNSSEC using DURZ.

But what load will appear at the root servers if a subset of the client resolvers starts to believe that these unvalidatable root keys should be validated?

What If…?

The problem with key rollover and local management of trust keys appears to be found in around 1 in every 1,500 resolvers in the **`in-addr.arpa`** zones. With a current client population of some 1.5 million distinct resolver client addresses each day for these **`in-addr.arpa`** zones, there are some 1,000 resolvers who have lapsed into this repeated query mode following the most recent key rollover of December 2009. Each subzone of **`in-addr.arpa`** has six Name Server records, and all servers see this pathological re-query behavior following key rollover.

The root servers see a set of some 5 million distinct resolver addresses each day, and a comparable population of nonupdated resolvers would be on the order of some 3,000 resolvers querying 13 zone servers, where each zone server would see an incremental load of some 75 Mbps.

Because the re-query behavior is caused by the client's being forced to reject the supposedly authoritative response because of an invalid key, and because DURZ is by definition an invalid key, the risk window for this increased load is the period during which DURZ is enabled, which for the current state of the root signing deployment is from the present date until July 2010. Because not all root servers have DNSSEC content or respond to the DO bit—and therefore do not return the unvalidatable signatures—the risk is limited to the set of DNSSEC-enabled roots, which is increasing on a planned, staged rollout. It has been reported that a decision to delay deployment of the DNSSEC/DURZ sign state to the "A" root server instance was made because this root server receives a noted higher query load for the so-called "priming" queries, made when a resolver is reinitialized and uses the offline root "hints" file to bootstrap more current knowledge. It is therefore likely that the "A" root server would also see increased instances of this particular query model, if the priming query is implicated in this form of traffic.

Arguably, this situation is unlikely. For most patterns of DNS query, failure to validate is immediately apparent. After all, where previously you receive an answer, you now see your DNS queries time out and fail.

However, because the typical situation for a client host (including *Dynamic Host Configuration Protocol* [DHCP] initialized hosts in the customer network space, the back office, etc.) is to have more than one listed resolver, there is the possibility of a misconfiguration being unnoticed during the period of a rolling deployment of DNSSEC-enabled services. In this situation if only one of the resolver's "nserver" entries is DNSSEC-enabled, either it is not queried or it is queried, but then passed over by the resolver timeout setting. Users see slower DNS resolution, but can attribute it to network delay or other local problems.

A second argument is that installation of hand-trust material is not normal, so the servers in question will be immediately known because a nonstandard process has to be invoked. Unfortunately, this situation is demonstrably not true. For example, the *Fedora*[5] release of Linux has included a simple DNSSEC-enabling process including a preconfigured trust file covering the reverse-DNS ranges. Because a previous release of this software included now stale keys (which have since been withdrawn in subsequent releases), any instance of *Fedora* for this release state being enabled will not only be unable to process reverse-DNS, it may also invoke this re-query mode of operation that places the server under repeated load of DNSKEY requests.

Because reverse-DNS is the "infrastructure" DNS query that is typically logged, but not otherwise used, unless the server in question is configured to block service on failing reverse (unlikely, given than more than 40 percent of reverse-DNS delegations are not made for the currently allocated IP address ranges), the end user simply might never notice this behavior. The use of so-called "Live CDs" can exacerbate this problem of pre-primed software releases that include key material that falls out-of-date. Even when the primary release is patched, the continued use of older releases in the field is inevitable. So perhaps this second argument is not quite as robust as originally thought.

Lastly, distinct from hand-installed local trust is the use of DNSSEC look-aside validation, which is known as DLV. This DNS namespace is privately managed and has been using the ICANN-maintained *Interim Trust Anchor Repository*, or ITAR. The DLV service is configured to permit resolvers to query it, in place of the root, to establish trust over subzones that exist in a signed state, but cannot be seen as signed from the root downward before the deployment of a signed root. There is now evidence that part of this query space exists, covering zones of interest to this situation. The **`.se`** zone key, for instance, is in the ITAR, as are the **`in-addr.arpa`** spaces signed by the RIPE NCC. Evidence suggests that if the DLV chain is being used and a key rollover takes place, some variants of BIND resolver clients fail to reestablish trust over the new keys until the client is rebooted with a clean cache state. This theory is difficult to confirm because as each resolver is restarted, the stale trust state is wiped out and the local failure is immediately resolved.

Post DURZ

Of course this phase is transitory, and even if there are concerns in terms of DURZ and queries to the root servers, all will be resolved when the root key is rolled to a validatable key on July 1, 2010.

Yes? Maybe not.

The current plan is to roll the root zone Key Signing Key every 2 to 5 years. The implication is that sometime every 2 to 5 years all DNS resolvers will need to ensure that they have fetched a new root trust key and loaded it into their resolver's local trust key cache.

If this local update of the root trust key does not occur, then the priming query for such DNSSEC-enabled resolvers will encounter this problem of an invalid DNSKEY when attempting to validate the priming response from the root servers. The fail-safe option here for the resolver client is to enter a failure mode and shut down, but there is a strong likelihood that the resolver client will try as hard as it can to fetch a validatable DNSKEY for the root before taking the last resort of a shutdown, and in so doing will subject the root servers to this intense repeated query load that we are seeing on the **`in-addr.arpa`** zone.

A reasonable question to ask follows: “Are there any procedural methods to help prevent stale keys from being retained during key rollover?” Reassuringly, the answer is “Yes.” There is a relatively recent RFC, “Automated Updates of DNS Security (DNSSEC) Trust Anchors,” RFC 5011[7], which addresses this problem.

RFC 5011 provides a mechanism for both signaling that a key rollover needs to take place and forward declaring the use of keys to sign over the new trust set to permit in-band distribution of the new keys. Resolvers are required to be configured with additional keying, and a level of trust is placed on this mechanism to deal with normal key rollover. RFC 5011 does not solve initial key distribution problems, which of course must be made out of band, nor does it attempt to address multiple key failures. Cold standby equipment, or decisions to return to significantly older releases of systems (for example, if a major security compromise to an operating system release demands a rollback) could still potentially deploy resolvers with invalid, outdated keys. However, RFC 5011 will prevent the more usual process failures, and it provides an elegant in-band rekeying method that obviates a manual process of key management that all too often fails through neglect or ignorance of the appropriate maintenance procedures to follow.

It is unfortunate that RFC 5011-compliant systems are not widely deployed during the lifetime of the DURZ deployment of the root, because we are definitely going to see at least one key rollover at the end of the DURZ deployment, and we can expect a follow-up key rollover within a normal operations window. The alternative is that no significant testing of root trust rollover takes place until we are committed to validation as a normal operational activity—a situation that invites the prospect of production deployment across the entire root set while many production operational processes associated with key rollover remain untested. The evidence from past concerns in resolver behavior is that older deployments have a very long lifetime for any feature under consideration, and because BIND 9.5 and older prerelease BIND 9.7 systems can be expected to persist in the field in significant numbers for some years to come, it is likely a significant level of pathological resolver behavior in re-querying the root services by active resolvers will have to be tolerated for some time.

It is also concerning that aspects of the packet traces for the **`in-addr.arpa`** zone suggest that for all key rollovers, albeit at very low levels of query load, some of the resolvers have simply failed to account for the new keys—and may never do so. Therefore, with increasing deployment of key validation, it is possible that a substantial new traffic class that grows, peaks, and then declines, but always declines to a slightly higher value than before, has to be borne, and factored into deployment scaling and planning.

Because this traffic is large—generating a kilobyte of response per query and potentially generally prevalent—it has the capability to exceed the normal response requirements for "normal" DNS query loads by at least one, if not two orders of magnitude. This multiplication factor of load is defined by the size of the resolver space and the number of listed Name Servers for the affected zone.

Mitigation at the server side is possible if this problem becomes a major one. The pattern of re-query here (the sequence of repeated queries for DNSKEY RRs) appears a potential signature for this kind of problem. Given that for any individual server the client times its repeat queries on the reception of the response from the previous query, delaying the response of the server to the repeated query will further delay the client's making its repeated query to this server. If the server were in a position to delay such repeated responses, using a form of exponential increase in the delay timer or similar form of time penalty, then the worst effects of this form of client behavior in terms of threats to the integrity of the ability of the server to service the "legitimate" client load could be mitigated.

Conclusion

It is an inherent quality of the DNSSEC deployment that in seeking to prevent lies, an aspect of the stability of the DNS has been weakened. When a client falls out of synchronization with the current key state of DNSSEC, it will mistake the current truth for an attempt to insert a lie. The subsequent efforts of the client to perform a rapid search for what it believes to be a truthful response could reasonably be construed as a legitimate response, if indeed this instance was an attack on that particular client. Indeed, to do otherwise would be to permit the DNS to remain an untrustable source of information. However, in this situation of slippage of synchronized key state between client and server, the effect is both local failure and the generation of excess load on external servers—and if this situation is allowed to become a common state, it has the potential to broaden the failure state to a more general DNS service failure through load saturation of critical DNS servers.

This aspect of a qualitative change of the DNS is unavoidable, and it places a strong imperative on DNS operations and the community of the 5 million current and uncountable future DNS resolvers to understand that "set and forget" is not the intended mode of operation of DNSSEC-equipped clients.

For Futher Reading

[0] A longer version of this article can be found in our online companion publication, *The Internet Protocol Forum*, `http://www.ipjforum.org/?p=226#more-226`

[1] Charalampos Patrikakis, Michalis Masikos, and Olga Zouraraki, "Distributed Denial of Service Attacks," *The Internet Protocol Journal*, Volume 7, No. 4, December 2004.

[2] Miek Gieben, "DNSSEC: The Protocol, Deployment, and a Bit of Development," *The Internet Protocol Journal*, Volume 7, No. 2, June 2004.

[3] O. Kolkman and R. Gieben, "DNSSEC Operational Practices," RFC 4641, September 2006.

[4] `http://www.unbound.net`

[5] `http://fedoraproject.org`

[6] `http://www.root-dnssec.org`

[7] M. St. Johns, "Automated Updates of DNS Security (DNSSEC) Trust Anchors," RFC 5011, September 2007.

[8] Geoff Huston, "Resource Certification," *The Internet Protocol Journal*, Volume 12, No. 1, March 2009.

[9] `https://www.isc.org/software/bind`

[10] `https://www.isc.org/community/blog/201002/signed-root-coming-and-what-means-you`

GEORGE MICHAELSON has a B.Sc. from the University of York and is a research scientist at the *Asia Pacific Network Information Centre* (APNIC), the Regional Internet Registry serving the Asia Pacific region. George explores problems in Internet Number Resource management, Internet standards, and network measurement by collaborative research. George has more than 28 years experience in computer science, networking, ICT administration, and research conducted in Australia and the UK. He participates in standards development in the IETF and has been a working group chair as well as an RFC author. He is a member of the *British Computer Society*. E-Mail: `ggm@apnic.net`

PATRIK WALLSTRÖM is a senior researcher at .SE, the Internet registry for SE domain names, and has been with the registry for eight years developing registry systems and working with the deployment of DNSSEC. Patrik is currently working on the *OpenDNSSEC* project, producing tools for a wider deployment of the technology. At .SE he is also managing the *Healthcheck* project, a new open source platform for measuring the quality of DNS, E-mail, Web and IP within Sweden. Patrik is also a board member of the *Swedish Network Users' Society* (SNUS). E-mail: `pawal@iis.se`

ROY ARENDS is a senior researcher at Nominet UK, the Internet registry for UK domain names. He co-authored several IETF standards on DNSSEC, resides on the board of DNS-OARC, is a member of ICANN's *Security and Stability Advisory Committee*, and is part of IETF's DNS-Directorate. As an expert on DNS and DNSSEC, Roy has co-initiated several DNS-related open source projects, such as *Unbound* and *OpenDNSSEC*. In the past, Roy was a member of, and chaired, CERT-NL. E-mail: `roy@nominet.org.uk`

GEOFF HUSTON holds a B.Sc. and a M.Sc. from the Australian National University. He has been closely involved with the development of the Internet for many years, particularly within Australia, where he was responsible for the initial build of the Internet within the Australian academic and research sector. The author of numerous Internet-related books, he is currently the Chief Scientist at APNIC. He was a member of the *Internet Architecture Board* (IAB) from 1999 until 2005, and served on the Board of the Internet Society from 1992 until 2001. E-mail: `gih@apnic.net`

Extending Router Lifetime with Virtual Aggregation

by Paul Francis, Max Planck Institute for Software Systems, and Xiaohu Xu, Huawei Technologies

Biologists believe that human life is limited by the number of times cells can replicate; noncancerous cells have a kind of internal counter that prevents them from replicating forever. Even if humans are kept healthy in every respect, they will eventually die simply because their cells will cease to replicate. Internet routers also have a finite lifetime. They are built with a fixed amount of hardware memory for storing the forwarding table (the memory structure that tells the router where to forward any IP packet, also called the *Forwarding Information Base* [FIB]). As the Internet global routing table grows, it eventually overflows the FIB, and the router ceases to be able to hold the full routing table. Even if the router is healthy in every respect (all of its hardware components still operate), it can no longer function as a router in the Internet *Default-Free Zone* (DFZ), where no default routes can be used.

In the past, router vendors have been reasonably good at predicting how long FIBs will last because the growth of the global DFZ routing table has stayed fairly predictable. As a result, *Internet Service Providers* (ISPs) can plan their capital budgets, and where necessary use a set of tricks (discussed in the next section) to squeeze additional life out of routers even after their "FIB death." But there are two problems.

First, these tricks work only in limited situations, they require extra configuration, and they can lead to increased traffic loads. Second, and potentially much more serious, the rate of routing table growth may dramatically accelerate in the near future, thus shrinking the lifetime of the installed router base. This expected acceleration is due to the imminent exhaustion of IPv4 addresses. In the past, address authorities such as the *American Registry for Internet Numbers* (ARIN) could assign large contiguous blocks of addresses to ISPs, which in turn assigned smaller blocks to their customers. Therefore, routers in other ISPs' networks need only a single routing table entry—that of the large block—to route to destinations in the ISP. This approach to scaling is called *address aggregation.* There is a fear that, as IPv4 addresses become increasingly unavailable, ISPs will start buying and selling smaller and smaller blocks of IP addresses from each other in an effort to squeeze out as many addresses as possible. These small blocks will appear all over the Internet thus significantly increasing the size of the routing table.

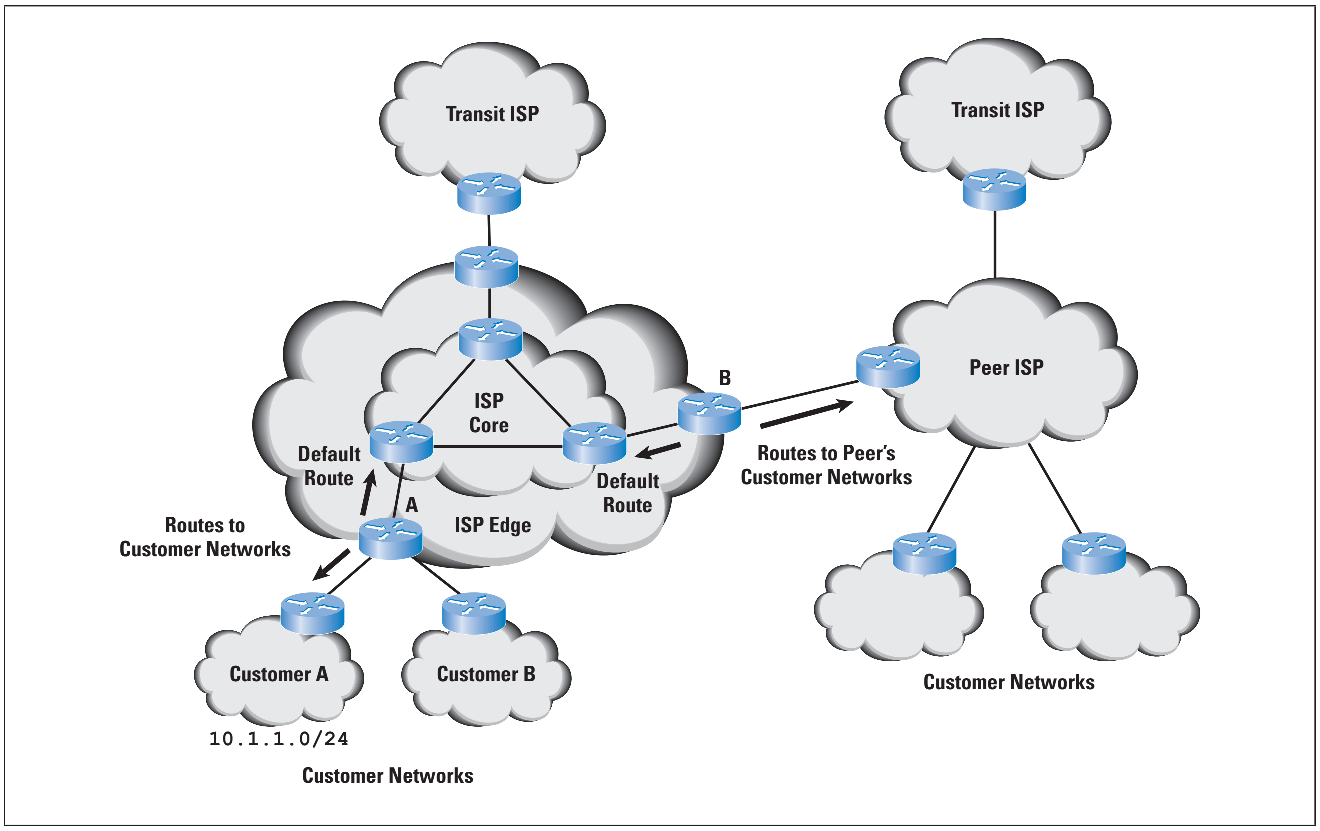
This article describes a new routing technology, called *Virtual Aggregation* (VA), which mitigates these problems. It makes extending the lifetime of old routers much easier, and makes it possible for existing routers to absorb a surge in the routing table size. Virtual Aggregation is a working item in the *Global Routing Operations Working Group* (GROW) working group of the IETF[7], and is documented in **`draft-ietf-grow-va`**[6] and related drafts.

Tricks for Keeping Old Routers Deployed

ISPs frequently want to extend the usefulness of a router beyond its "FIB death," and there are many tricks for doing just this. The most common is to structure the ISP in a core-edge arrangement. In this setup, a core of routers forms the backbone of the network. Edge routers connect to other networks and feed into the core. In many cases these edge routers do not need to know how to route to everything in the Internet. Rather, they often need to know only what addresses are reachable in their directly connected networks.

For instance, Figure 1 shows an ISP whose edge routers connect to three types of other networks: customer networks, peer ISP networks, and transit ISP networks. Each customer network has only one or a small number of address prefixes. The edge routers connecting customer networks must know what addresses are reachable in the customer networks, but everything else can be "default routed" to the core. Likewise, the routers connected to peer ISPs need to know how to route to the peers' customer addresses. Everything else can be defaulted to the core. The core routers and the edge routers that connect to transit ISPs, however, need to know how to route to everything.

Figure 1: With a core-edge style of deployment, some routers need to keep full routing tables, while others can keep partial routing tables and default route everything else to the ISP core.



A common practice is for ISPs to delegate FIB-dead routers to the customer or peer edges, and to have the core routers filter the routing information given to the edge routers. For instance, router A in Figure 1 learns the addresses reachable in customer network A (say, `20.1.1.0/24`) and conveys them to the ISP core, but the core tells router A only that "everything else" is reachable through it (`0.0.0.0/0`). But what if customer A itself wants the full DFZ routing table? For instance, customer A might be multihomed to some other ISP, and might want to know which Internet destinations are best reachable through each ISP. To do this, it needs to receive the whole routing table from each ISP, a situation that, of course, cannot happen if the core withholds routes from router A.

As another example, what if two peer ISPs later decide that they want to offer transit service to each other? Now additional routes need to be conveyed to the peer-connected edge routers (router B), and this process may not be possible with limited FIB.

Another way an ISP can shrink its routing table is to default route to its transit ISPs. For instance, routers keep track only of how to route to customers and peers, and everything else is defaulted to the transit ISPs. When this default routing is done, even an ISP's core routers do not need the full routing table. A simple approach is for an ISP to send all defaulted packets to the nearest transit ISP. This process, however, may result in many packets taking a longer Internet path than necessary. Reference [1] describes a more complicated approach where the ISP maintains "semidefaults" for different transit networks in order to improve its global routing while reducing routing table size by about half. This approach, however, can be hard to manage.

In addition, any form of ISP-level default (simple or complex) results in sending extra traffic to the transit ISPs. A substantial amount of Internet traffic is targeted to nonroutable prefixes. When an ISP has the full routing table, it can identify this traffic and drop it before sending it to its transit ISPs. When an ISP defaults, it sends this traffic to its transit ISPs, and pays for it.

To summarize, dealing with FIB-dead routers leads to more complex management, limitations in business arrangements with peers and customers, poor paths over the Internet, and increased traffic load.

The Idea of Virtual Aggregation

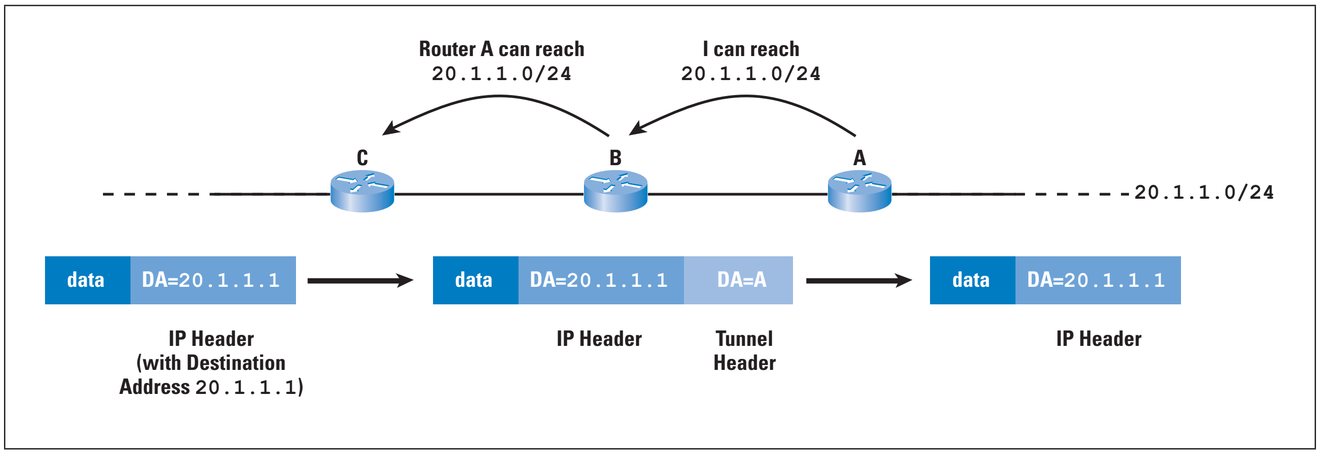
In its simplest form, Virtual Aggregation allows an ISP to use FIB-dead routers as edge routers, in any edge router position (neighbor is a transit provider, a peer, or a customer) without limiting what routing information is exchanged. Configuration requirements are minimal. In a more complex form, Virtual Aggregation allows all ISP routers (not just edge routers) to be FIB-dead routers, without requiring ISP-level default routing.

Virtual Aggregation uses two basic mechanisms, FIB suppression and tunneling. Before discussing FIB suppression, a small amount of background is needed. Internet routers have a "data plane" and a "control plane." The data plane is what forwards packets, and includes such functions as header parsing, FIB lookup, queuing, and packet transmission. The control plane operates the background protocols that gather much of the information needed by the data plane. Examples include routing protocols such as the *Border Gateway Protocol* (BGP) and *Open Shortest Path First* (OSPF), and tunnel establishment protocols such as the *Label Distribution Protocol* (LDP).

The idea of FIB suppression is that the control plane operates as normal, but that certain routing table entries are not loaded into the FIB. This idea exploits the fact that it is (data plane) FIB memory, not control plane routing table memory that is the more severe bottleneck. By allowing the control plane to operate as normal, no changes are required to routing protocols or, for the most part, the management of routing protocols.

Tunneling is used to pass packets through routers that have suppressed FIB entries. The principle is illustrated in Figure 2. Here router A tells router B that it can reach `20.1.1.0/24`. Router B in turn tells router C that router A can reach `20.1.1.0/24`. As a result, router C tunnels packets destined for `20.1.1.0/24` to router A through router B. In other words, it wraps the IP header in another IP or a *Multiprotocol Label Switching* (MPLS) header that first gets the packet to router A. Router A strips that header, and sends the packet toward the destination. Notice that router B can suppress the route to `20.1.1.0/24` from the FIB—it only needs to know how to route the packet to router A. In other words, even though router B fully participates in the control plane, it is able to shrink its FIB through FIB suppression and tunneling.

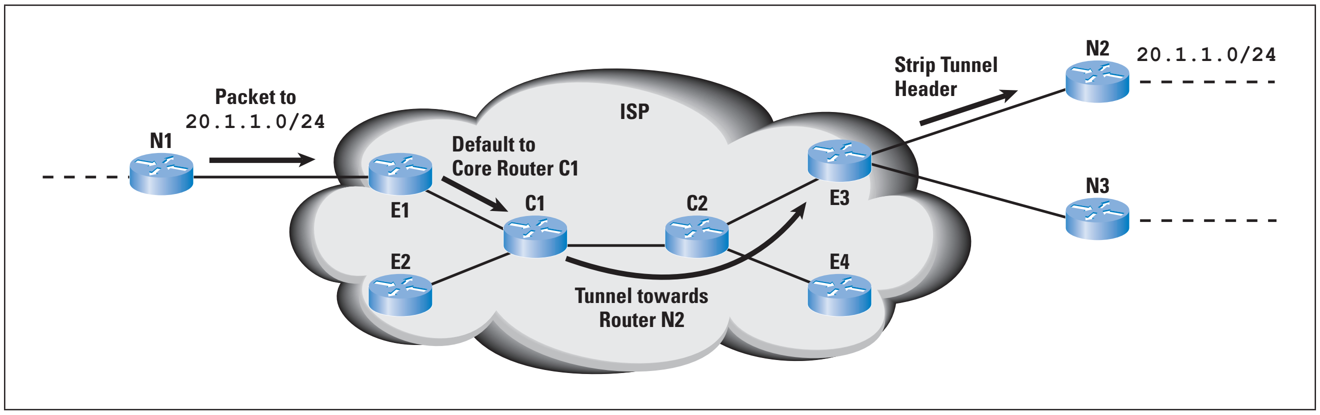
Figure 2: Because router C tunnels the packet to router A, router B does not need to know how to forward packets with addresses in `20.1.1.0/24`.



Virtual Aggregation in Practice, Simple Version

In the simplest version of Virtual Aggregation, a core-edge configuration is used. The core routers maintain full FIB tables. The edge routers FIB-install at least a default route to the core, and potentially additional routes if there is space in the FIB. This process is illustrated in Figure 3. Here there are two core routers, C1 and C2, and four edge routers, E1, E2, E3, and E4. The edge routers have external neighbors, N1, N2, and N3, as shown.

Figure 3: Packets can be delivered to **`20.1.1.0/24`** even if none of the edge routers has a FIB entry for **`20.1.1.0/24`**.



The operation is best explained by example. Suppose that N2 advertises a route to destination **`20.1.1.0/24`** to E3 using *External BGP* (eBGP) and giving itself as the next hop. E3 in turn advertises this route to the other internal routers using *Internal BGP* (iBGP), with the next hop still as N2. The core routers install this route in their FIBs, with an indication that packets matching the route should be tunneled to the next hop, N2. Assume for now that all edge routers FIB-suppress the entry. When a packet for say **`20.1.1.1`** arrives at E1 from N1, E1 does not find an entry for **`20.1.1.0/24`**, but does find the default route **`0/0`** telling it to forward the packet to its core router C1. C1 looks into its FIB and indeed finds an entry for **`20.1.1.0/24`** telling it to tunnel the packet to N2. C1 wraps the packet in another header, typically IP or MPLS, addressed to N2. When the packet reaches E3, however, E3 notes that the header directs it to send the packet to N2, strips off the outer header, and sends the packet to N2. E3 can do this without a FIB entry for **`20.1.1.0/24`**.

MPLS already has all the mechanisms needed to perform this packet forwarding. E3 can use LDP to signal a *Label Switched Path* (LSP) to N2, and *Penultimate Hop Popping* can be used to strip off the MPLS header before forwarding the packet to the external neighbor N2 (as described in section 4.1.4 of [4]).

Alternatively, stacked MPLS label technology can be used; for example, the inner label is signaled with BGP (see "Carrying Label Information in BGP-4"[3]) while the outer label is signaled with LDP. Here E3 sets itself as the next hop for all the routes learned from external neighbors (for example, `20.1.1.0/24`) when advertising them to its iBGP peers, and uses the inner label to identify the external neighbor (see section 4.3, "Label Stacks and Implicit Peering" of [4]). IP-in-IP tunneling can also be used, in this case signaled with softwires BGP attributes[5].

Now let's see what happens if a packet to `20.1.1.1` is received by E3 from external neighbor N3. If E3 has not FIB-installed the route for `20.1.1.0/24`, it uses its default entry and forwards the packet to C2. C2 finds its entry for `20.1.1.0/24`, which instructs it to tunnel the packet to N2. The packet is sent back to E3, which strips off the outer header and delivers the packet to E2. In this case, the packet has traveled an extra hop and back, a process that is not acceptable if done too much. As long as there is space in the FIB, however, routers are free to FIB-install additional routes. A good policy is to always install routes when external neighbors are the next hop. This policy avoids the longer path. In some cases, such as edge routers that connect to transit networks, there may not be enough FIB space to hold all routes from all external neighbors. In this case, the router may FIB-install the routes for which the most traffic is forwarded. Studies have shown that a small number of routes account for majority of the traffic, making Virtual Aggregation a very efficient solution[2].

Note that this simple form of Virtual Aggregation is very easy to configure. Essentially all that is needed is to tell the routers that they are using simple Virtual Aggregation, and to tell them if they are a core or an edge router. The routers can automatically configure everything else. Virtual Aggregation requires configuration of tunnels from every router to every other router, but these configurations also can be automatic. In any event, increasingly these tunnels are created anyway for the purpose of traffic engineering.

Simple Virtual Aggregation solves most of the problems described earlier. It can save FIB on any edge router without having to compromise BGP service to customers or flexibility in using peer networks for some transit. It also allows FIB-dead routers to be used as edge routers with transit ISPs. Finally, it prevents the need for ISP-level default routing to transits, thus avoiding unnecessarily sending unroutable traffic to the transit. And it does all this with much less configuration than is required to operate with FIB-dead routers today.

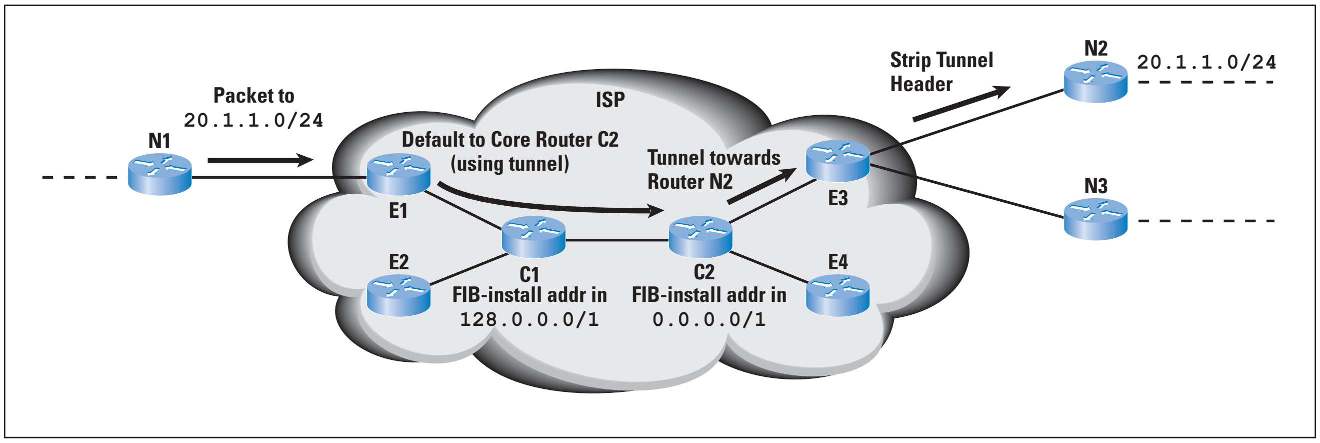
Virtual Aggregation in Practice, Complex Version

The simple version of Virtual Aggregation is satisfactory for edge routers, but it does nothing to reduce FIB size on core routers. What if an ISP wishes to also extend the lifetime of its core routers? Or wants to move away from a core-edge model, and rather connect all edge routers directly through a Layer 2 substrate like MPLS?

What if indeed there is a surge in routing table growth, thus causing ISPs all over the world to suddenly find themselves FIB-starved? There is a version of Virtual Aggregation that allows for FIB reduction in any and all routers in an ISP network.

The basic idea is to divide the address space so that different routers maintain full routes within different parts of the address space. So for instance, rather than have all core routers responsible for all of the address space, you could have half of the core routers responsible for the lower half of the address space, and the other half of the core routers responsible for the upper half of the address space. Figure 4 shows how this setup would look for the simple topology of Figure 3, keeping in mind that this example is rather simplistic.

Figure 4: In a complex version of Virtual Aggregation, even core routers do not need to hold the full routing table.



Assume that C2 FIB-installs only the lower half of the address space (`0.0.0.0/1`) and C1 FIB-installs the upper half (`128.0.0.0/1`). With this arrangement, the edge routers have two defaults instead of one. Packets to addresses in `0.0.0.0/1` are defaulted, through a tunnel, to C2, and packets to addresses in `128.0.0.0/1` are defaulted to C1. These defaults are learned simply by having C1 and C2 advertize their respective default routes with themselves as the next hop in iBGP.

As with the previous example, assume that router N2 advertises a route to `20.1.1.0/24`, with itself as the next hop, to E3. E3 advertises this route to all other routers using iBGP. Only C2, however, FIB-installs this route—C1 suppresses it. When a packet to `20.1.1.1` arrives at E1, it looks in its FIB, finds a matching route to `0.0.0.0/1`, and so tunnels the packet to C2. C2 terminates the tunnel, finds its FIB entry for `24.1.1.0/24`, and tunnels the packet toward N2. E3 uses the tunnel information to know to forward the packet to N2, strips away the tunnel header, and forwards the packet to N2.

Now suppose that a packet for `20.1.1.1` arrives at E3 from N3. Ideally E3 has already automatically FIB-installed the route for `24.1.1.0/24` either because its external neighbor provides the next hop, or because the route is a high-volume destination. In this case, of course, the packet is directly forwarded to N2. However, if E3 has not FIB-installed the route, then its best match is the default to `0.0.0.0/1`, and it tunnels the packet to C2. C2 in turns tunnels the packet back toward N2 through E3 as described before. Worse, if C1 rather than C2 FIB-installed the lower half of the address space, the packet would have detoured all the way to C1. Clearly these routes are not optimal, and so we must ask how nonoptimal would the complex version of Virtual Aggregation be in real ISPs.

The USENIX NSDI paper[2] answers this question for one large transit ISP. In this study, both the topology and the traffic matrix of the ISP are considered. The deployment strategy is substantially more complex than the simplistic example given previously. An upper limit is placed on the maximum increase in latency (5 ms) for any path through the ISP. There is a requirement that within a *Point of Presence* (PoP) at least two routers must cover the same address space. The number and size of address partitions are engineered to spread FIB load evenly. The "additional" routes installed in the FIB are designed to cover high-traffic destinations to the extent possible.

With these requirements in mind, this study found that FIB size could be reduced in all routers by at least an order of magnitude with a negligible increase (1–2%) in overall traffic load due to the occasional extra hops from the detours. This result ultimately translated into an increased router lifetime of easily 10 years.

The management requirements for the complex version are substantially greater than those for the simple version. The address partitions must be chosen, the routers assigned to address partitions must be chosen, and possibly some strategy for deciding what "additional" routes should be FIB-installed is needed. Whether this added configuration and the associated difficulties due to, for instance, misconfiguration are worth the cost savings for extending router lifetime is up to each ISP. Virtual Aggregation at least provides an option that was not previously available.

Status

Virtual Aggregation is a working-group item in the *Global Routing Operations Working Group* (GROW) in the IETF. The primary draft is **`draft-ietf-grow-va`**[6]. This draft has gone through several revisions, and is very close to its final form. Huawei is currently implementing Virtual Aggregation. A second open-source implementation has been built by Paul Francis' research group for the *Quagga* open-source routing platform, and is still being enhanced.

Acknowledgements

The authors would like to thank the co-authors of the Virtual Aggregation drafts, Hitesh Ballani, Dan Jen, Robert Raszuk, and Lixia Zhang. In particular, it was Robert who suggested the simple version of Virtual Aggregation.

References

[1] Andre Chapuis, "BGP Filtering," Presentation from SWINOG7, `www.swinog.ch/meetings/swinog7/BGP_filtering-swinog.ppt`

[2] Hitesh Ballani, Paul Francis, Tuan Cao, and Jia Wang, "Making Routers Last Longer with ViAggre," USENIX NSDI 2009, April 2009.

[3] Y. Rekhter and E. Rosen, "Carrying Label Information in BGP-4," RFC 3107, May 2001.

[4] E. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol Label Switching Architecture," RFC 3031, January 2001.

[5] P. Mohapatra and E. Rosen, "BGP Encapsulation SAFI and BGP Tunnel Encapsulation Attribute," RFC 5512, April 2009.

[6] P. Francis, X. Xu, H. Ballani, D. Jen, R. Raszuk, and L. Zhang, "FIB Suppression with Virtual Aggregation," October 2009, `draft-ietf-grow-va-01`.

[7] IETF Global Routing Operations Working Group (GROW), `http://www.ietf.org/dyn/wg/charter/grow-charter.html`

PAUL FRANCIS is a faculty member at the Max Planck Institute for Software Systems in Germany. He has been active periodically in the IETF for nearly 20 years. Dr. Francis has held research positions at Cornell University, ACIRI, NTT Labs, and Bellcore, and was Chief Scientist at Fast Forward Networks and Tahoe Networks.
E-mail: `francis@mpi-sws.org`

XIAOHU XU graduated from Beijing University of Posts and Telecoms in 2000. He has been working in the telecom industry for about 10 years and now is a research engineer with IP Advanced Technology Research Department of Huawei Technologies. Before joining Huawei at the end of 2004, he was the chief engineer of the Technical Support Department for Harbour Networks.
E-mail: `xuxh@huawei.com`

RFC Editor in Transition: Past, Present, and Future

by Leslie Daigle, ISOC

In April 2009, the *Request For Comments* (RFC) Editor published RFC 5540[1], "40 Years of RFCs," which summarized the publication history of the RFC Series. The series has been the technical publication series for Internet technology since long before there was an *Internet Engineering Task Force* (IETF). Although the RFC Series is the publication vehicle for the IETF, it has been, and remains, scoped more broadly than that (refer to RFC 4844[2], "The RFC Series and RFC Editor").The RFC Series is the archival series dedicated to documenting Internet technical specifications, including general contributions from the Internet research and engineering community as well as standards documents.

For the past three of the four decades of the history of the series, the RFC Editor work has been carried out at the *University of Southern California Information Sciences Institute* (USC/ISI). The RFC Editor role now faces another evolutionary step: The work involved in managing the overall series is being split up to recognize the different components of the editing, production, and archiving activities and to lay the groundwork to ensure its continued success, as outlined in RFC 5620[3], "RFC Editor Model (Version 1)."

At the IETF 76 plenary in Hiroshima, Japan, in November 2009, USC/ISI and the role it has played in supporting the RFC Editor over the past 30 years were given special recognition. Some members of the team will move from USC/ISI to the RFC Editor's new home, where they will continue their work. We took the opportunity to talk with current and future RFC Editor staff and advisory board members, including current RFC Editor staff members Bob Braden, Sandy Ginoza, and Alice Hagens, as well as Bob Hinden, who is a member of the RFC Editor advisory board.

The People Behind the RFC Editor

Jon Postel was the first RFC Editor, starting the position in 1969 as an activity to keep track of RFC Series documents. Bob Braden, who was then part of the *Advanced Research Project Agency Network* (ARPANET) research program, told how he got started with the RFC Series: "I wrote my first RFC in the early 1970s, when it was somewhere around RFC 100. I was at that point manager of programming for the Computing Center at the *University of California, Los Angeles* (UCLA), and *Advanced Research Projects Agency* (ARPA) wanted to connect it to ARPANET as a resource." This was all pre-TCP/IP, and Bob's staff had to implement file transfer and Telnet. At the same time, Jon was a graduate student at UCLA, and Bob worked with him as a colleague. It was before Jon got his Ph.D. and moved to SRI in 1973–1974. In 1980, Jon moved to USC/ISI, taking the RFC editorship with him. Joyce Reynolds went to work for Jon at USC/ISI. She did much of the actual editing and became an important part of making the RFC Editor activity viable.

Jon was responsible for quality control, running the operation, and generally being the series editor. When Jon died suddenly in 1998, Bob, who joined USC/ISI in 1986, and Joyce both felt a keen sense of loss. "Jon was a very remarkable guy in many ways," Bob said. "We knew how much the RFC Series meant to Jon, and we volunteered to carry it on."

Sandy Ginoza joined USC/ISI to work on the RFC Editor activity in 1999, just after Jon passed away. Alice Hagens came onboard in 2005, taking on more of the computer-oriented aspects of the work.

RFC Series

Although we tend to reference and read individual RFC documents, it is important to understand that there is significant value in the collection of published RFCs as a *series*. On the importance of the RFC Series, Bob Hinden said, "This community is IETF-focused, but to the larger world not centered around the IETF; it's really the RFCs that are how you build the Internet. One of the things that made the Internet possible was the RFC Series: that you could build things and deploy things without coming to IETF meetings was valuable." Bob went on to outline his own experiences, such as meeting engineers in Taipei, for whom it was the first time they had ever met anyone who had written an RFC. Even the notion of going to an IETF meeting was in another dimension. "The RFC Series is what enables people to build products, networks, and the Internet," he said.

And it is quite an active series. Currently, some 300 documents (10,000 pages) are published every year, and although it might be interesting to review the material to detect trends or arcs of work in the Internet technical community, that type of activity is beyond the current scope of the RFC Editor. Focusing on consistency of the series, Bob Braden wondered, "Will we eventually have good enough statistics from the errata system to gauge our error rate?"

The intent of the RFC Series is to serve the broader Internet community; it is not just for or by the IETF. Sandy's perspective on the value of the *Independent Stream* of RFCs is that "it offers an alternate view than what happens in the IETF and what working groups have decided to take on as part of their chartered activities. It's good to document that work was done, results were generated, lessons learned, etc. 'We tried it; don't do it this way.' We often get asked why it's called RFC when we're not really requesting comments anymore, but that is the genesis, and the Independent Stream keeps some of that alive."

Bob Braden offered his own perspective on the Independent Stream.

"Historically, the RFC Series is supposed to be larger than the IETF, and while Jon was alive, the editor did whatever he thought he ought to do; the community didn't question it much."

However, in the absence of Jon as an authority figure, the community began to ask questions and build its own set of beliefs, eventually coming to believe that RFCs were only for the IETF. That matter was resolved with RFC 4846[4], which explained that there is a separate set of independent submissions that do not come through the IETF.

"It's not a big stream, not a lot of documents, but it is important philosophically," Bob added. "The Internet community is bigger than the IETF."

The RFC Series is, nevertheless, entwined with the IETF and its activities. For instance, the discussion of (IETF) *Intellectual Property Rights* (IPR) has led to an impasse in assigning boilerplate to RFCs that allow the continued publication of the Independent Stream documents. That subject is being worked on and resolved, but it offers an example of some of the complexities—and frustrations—that can arise as part of the RFC Editor process. "The current situation—that the independent submissions cannot be published because we don't know what the boilerplate is—is just terrible," said Bob Braden.

Bob Hinden, who has been tracking the IPR work from the IETF side, agreed and elaborated on some important lessons learned: "The IETF created a process in the IPR working group that focused on trying to provide a solution to what they perceived as a problem. But they lost sight of the complexity and cost of implementing that solution compared with the actual risk of something bad happening. We have learned a lot about doing this in the future. This isn't like a protocol spec where you fix a bug in the finite state machine. This has a real effect on people doing stuff. When you ask for legal opinions you get the answer about how to solve the problem, but that's not the end of the process. You need to balance the cost of solving the problem with the risk of what you're trying to avoid. Lawyers are supposed to give you the lowest-risk answer. You need to follow through with questions about likelihood and consequences. This is all great hindsight, and I hope we can apply it in the future." Hinden also said he believes the current impasse could have been avoided if the new procedure had specified that it go into effect when appropriate supporting conditions were met, instead of on a specific flag day, such as the date of publication of the RFC.

The effects of entwining the RFC Series and the IETF go both ways. For example, the RFC Series recognizes three levels of standards documents: *Proposed, Draft,* and *Full*. The expectation, documented in the IETF standards process, is that standards-track specifications should be published as Proposed and then advanced to Draft and Full as the specification gets tested commercially and acknowledged as appropriately mature to move to the next stage.

In reality, as observed at the IETF 76 plenary, many of the important specifications that form the basis of the operating Internet are still published only as Proposed Standard. Bob Braden explained the history of the standards-track RFC maturity system this way: "Labels were invented whole cloth by the original *Internet Architecture Board* (IAB), who were a bunch of academics. At that point the Internet had not been commercialized—there were no commercial pressures—so we imagined that it made sense to step through progressions in a theoretical world. In the real world, companies are putting out products. There is no financial incentive for people to spend time advancing documents. Plus, the IETF is so large and there are so many working groups that we try to dispatch them as fast as we can; there is no one around to advance a document." There have been, and will continue to be, proposals for moving important, current standards (such as the *Border Gateway Protocol* [BGP]) forward in maturity or for collapsing the maturity scale and labeling system.

On the fun side of the RFC Series, there remains a tradition of "April 1st" RFCs. "That people want to participate in that is cool," said Sandy. "And we get to see the runners-up and the really-not-so-good ideas!"

Alice agreed, adding that "there are high standards for straight-faced satire."

RFC Editor

Traditionally, the RFC Editor has not only populated the series with new (approved) documents but also kept all the threads together in the RFC Series. Describing the origins of the role, Bob Braden pointed out that "Originally, Jon was prince of his kingdom. As RFC Editor, he was an honorary member of the IAB informally called the *Protocol Czar*. He used the RFC Editor position to actively prevent bad ideas from getting pushed. Jon imposed a consistency of style on the document series. You pick up RFC 1001 and compare it with 2001, and they look very similar." Jon believed, and the RFC Editor continues to believe today, that consistency was a worthwhile attribute, promoting stability in the series.

Reflecting back, Bob Braden said, "In discussions over the last five years, people have expressed the view that we don't need an RFC Editor—just take an Internet Draft and publish it. That notion drives me crazy. The implication is that it doesn't matter whether it is good English, correctly referenced, consistent, etc. I can't stand that view." One of the arguments for such an approach to IETF document publishing is that editing can inadvertently alter, and thereby introduce errors to, text. But the RFC Editors understand that.

Alice said changes to text can be problematic, "partly because of the technical content and partly because it is a group process. It's agreed-upon text. The idea is how precious the text is and how a slight change can make a large difference."

Sandy agreed, adding that "for as many changes that get pushed back upon, there are many that make it through the process: for as many people as look at the document before it gets to us, there are things that escape them; there is often missing text, missing words." According to Alice, with working group documents, people often focus on getting the technical ideas right, but nobody has read the text from beginning to end. In addition, many in the community are not native English speakers. It all comes back to the consistency and professionalism of the output of the series.

RFC Editing Process

As the RFC Series has grown, achieving consistency has required the creation and refining of processes. "When Joyce and I took over," said Bob Braden, "we built the website and regularized a lot of things, and the community began to ask, 'Why do you do it that way?'" In response, the editors started publicizing the *Style Manuals* they used. Joyce and Bob generated a lot of rules that have become institutionalized.

Of course, there is continuing evolution. Bob Braden noted that the addition of errata was his idea, although "it has turned out to be a much, much bigger deal than ever imagined, as is often the case," he said, laughing. "Now we're talking about adding image files to solve the problem of incorporating graphics in an ASCII RFC. John Klensin and I generated a plausible solution for that, and we hope to get it installed soon."

It is important to note that there are some edits the RFC Editor will not make. According to Sandy, the RFC Editor tries to ensure consistency of terminology and to make recommendations that improve consistency within a document, both in a technical sense and within the series. "We don't change the active/passive voice," she said.

"We might suggest it, but we are concerned that it would affect the author's intent." Being conservative is critical. Sandy said she was surprised by how "simple grammatical changes can have a serious technical effect; placement of a comma can make a big difference in how people read the document and what they implement."

Working with authors is an important part of making the editing process successful. Innovations such as having the *RFC Editor Help Desk* at IETF meetings and making the AUTH48[5] (final check of the RFC Editor's edits) more of an interactive dialogue have helped build community and create awareness of how to build a better document that conveys the meaning as intended. "It is extremely useful to get discussions started earlier, which lessens problems during AUTH48," said Alice. She added that it has also been useful to have face time with the developers of community-created tools, such as *xml2rfc*[6] and the *Augmented Backus–Naur Form* (ABNF) checker, which have been instrumental in improving RFC production. Office hours, building relationships, and face time "all help make it about working together," said Sandy.

Looking forward, Sandy said she would like to see the RFC editing process (and series) "grow and continue to be more consistent, with better community relations and more transparency so authors can look at our site and better understand the process, instead of thinking their document has gone into a black box."

On the Verge of Major Change

As this article is written, the RFC world is on the brink of major structural change. Following IAB-led community discussion, there is a new model for recognizing the components of activity that make up the RFC activities. ISI is handing off the RFC Editor activity, which will be taken up by separate organizations working together. In February 2010, the IAB appointed Nevil Brownlee as the *Independent Submissions Editor* (ISE) and Glenn Kowack as the Transitional *RFC Series Editor* (RSE). In October 2009, *Association Management Solutions* (AMS) was awarded 2-year contracts to manage the RFC Production Center and the RFC Publisher.

Sandy will be joining AMS as RFC Production Center director and Alice will be joining as senior editor and information technology development project manager. To the question of whether the current RFC advisory board will carry forward in the current format or will change, Bob Braden answered, "The current board serves two functions: It provides a supply of experienced people who review independent submissions, but it also gives the RFC Editor advice on policy matters. Some members of the advisory board are very strong members of the IETF in terms of policy advice. In forming the board, I tended to identify a subset of people within the IETF who have long IETF and publishing experience. In the new world there will be an *RFC Series Advisory Group* (RSAG), which will take over the policy discussions that are currently being conducted by the editorial board. In practice it will be the same people, at least for a while, but with separate duties. That separation is useful."

In considering the change of organizations, Sandy said the biggest thing in moving to AMS is that it is a more service-oriented environment. "In the new model," she said, "it is important that the ISE and RSE be respected individuals who are granted some of the independence the RFC Editor had at ISI."

Alice added that the institutional memory of the RFC Editor function will not be lost with the move to AMS. "Sandy has worked side by side with Bob Braden for 10 years, and much of the process is written down in the document series. I'm confident that the continuity of the series won't be lost by the move to AMS."

Bob Hinden offered another perspective. "I think one of the positive things that has come out of the new model that has gotten lost is this: A lot of people in the IETF didn't understand where the series had come from, or why the IETF chose to use it," he said.

"It is the formalization that there are different streams that have different rules. Before, this was confused with the IETF standards process. Going forward we'll have the opportunity to use the RFC Series for other relevant Internet publication streams that have not been part of IETF. Now we have a framework that would allow that."

Although it is on the verge of major changes, the RFC Series and RFC Editor functions are clearly continuing what has been a long process of constant evolution and change. This transition is just a new chapter in the history of the series.

[Ed.: This article is composed of interviews conducted by Leslie Daigle and Lucy Lynch, and notes compiled by Mat Ford. The original version was published in *The IETF Journal,* Volume 5, Issue 3, January 2010 and has been been updated for use in IPJ. *The IETF Journal* can be obtained from `http://isoc.org/ietfjournal/`]

For Further Reading

[1] RFC Editor, "40 Years of RFCs," RFC 5540, April 2009.

[2] L. Daigle, Ed., Internet Architecture Board, "The RFC Series and RFC Editor," RFC 4844, July 2007.

[3] O. Kolkman, Ed., IAB, "RFC Editor Model (Version 1)," RFC 5620, August 2009.

[4] J. Klensin and D. Thaler, Eds., "Independent Submissions to the RFC Editor," RFC 4846, July 2007.

[5] `http://www.rfc-editor.org/pubprocess.html`

[6] Marshall T. Rose and Carl Malamud, "Writing Internet Drafts and RFCs Using XML," *The Internet Protocol Journal,* Volume 10, No. 1, March 2007.

Alice Hagens, Bob Braden, and Sandy Ginoza are recognized at IETF 76 for their work with the RFC Editor. (Photo: Internet Society)

Fragments

IETF Outcomes Wiki Launched

As an organization, the *Internet Engineering Task Force* (IETF) measures its success by its publication of RFCs (see previous article). It does not explicitly ask itself whether published work is adopted and used by the greater Internet community. The IETF's dialogue about success started to change with the production of RFC 5218, "What Makes for a Successful Protocol?"[1] which documented case studies and empirical data about some of the factors that appear to correlate with success, in terms of community uptake for IETF work.

Taking a different approach in assessing long-term IETF impact, another tool is now available: A wiki that lets community participants list the success or failure of significant standards. The *Outcomes Wiki*[2] divides listings according to the "areas" used for managing technical work in the IETF, such as Applications or Transport. Outcomes are rated according to a 6-point scale, ranging from "complete failure" to "massive adoption, plus extensive derivative work."

The wiki began in June 2009, as an independent effort among a small set of IETF participants, to test its feasibility and evolve its design. For example, it quickly became clear that the single attribute of success vs. failure needed to be qualified by another attribute that indicates who the work is intended for, called "Target Segment." Work that is intended to support the internal operations of an *Internet Service Provider* (ISP) is not necessarily visible to the billions of Internet users and will, at best, be part of only a few thousand organizations. In terms of Internet scale, that is considered minuscule. However wide adoption of a tool among ISPs can have substantial benefit, and thereby qualify as "massive adoption."

The wiki can serve both as a means of recording the IETF's track record of successes and failures, as well as providing a means of encouraging community dialogue about the quality of different IETF efforts. In addition, it can provide a window onto completed IETF work for the broader Internet community.

[1] D. Thaler and B. Aboba, "What Makes for a Successful Protocol?" RFC 5218, July 2008.

[2] `http://trac.tools.ietf.org/misc/outcomes/`

Final Phase of Four-byte AS Number Policy Begins in APNIC Region

From 1 January 2010, the *Asia Pacific Network Information Centre* (APNIC) ceased to make a distinction between four-byte only and two-byte only *Autonomous System* (AS) numbers. Instead, all AS numbers are now considered to be four-byte AS numbers.

This change marks the third phase of the transition to four-byte AS numbers. For more information on the implementation phases of the four-byte AS number policy, please see "Policies for Autonomous System number management in the Asia Pacific region," section 6.3, "Timetable for moving from two-byte only AS numbers to four-byte AS numbers," available from:
`http://www.apnic.net/policy/asn-policy.html#6.3`

To learn more about how the transition to four-byte AS numbers may affect your network, see: `http://icons.apnic.net/asn`

Charting the Course for Future Internet Leaders

As the importance of the Internet grows in all aspects of modern life, so too do the challenges of those in positions of leadership and responsibility.

Responding to the need for well-qualified leadership, the *Internet Society* (ISOC) is now accepting applications from people seeking to join the new generation of Internet leaders to address the critical technology, policy, business, and education challenges that lie ahead.

Successful candidates in ISOC's *Next Generation Leaders Program* will gain a wide range of skills in a variety of disciplines, as well as the ability and experience to work with people at all levels of society.

This program, under the patronage of the European Commission, blends course work and practical experience to help prepare young professionals (aged from 20 to 40) from around the world to become the next generation of Internet technology, policy, and business leaders.

"The Internet Society's Next Generation Leaders Program is a unique opportunity to identify potential Internet leaders and help them accelerate their careers," said Bill Graham, responsible for strategic global engagement at ISOC.

The key to the Internet's success lies in the Internet Model of decentralized architecture and distributed responsibility for development, operation, and management. That model also creates important leadership opportunities, especially in those spaces where technology, policy, and business intersect.

"We have designed the Next Generation Leaders Program to prepare young professionals for leadership, bridging the boundaries between business, technical development, policy, and governance on local, regional, and international levels," said Graham.

Full details of the Next Generation Leaders Program are available at:
`http://www.isoc.org/leaders/`

Call for Papers

The Internet Protocol Journal (IPJ) is published quarterly by Cisco Systems. The journal is not intended to promote any specific products or services, but rather is intended to serve as an informational and educational resource for engineering professionals involved in the design, development, and operation of public and private internets and intranets. The journal carries tutorial articles ("What is...?"), as well as implementation/operation articles ("How to..."). It provides readers with technology and standardization updates for all levels of the protocol stack and serves as a forum for discussion of all aspects of internetworking.

Topics include, but are not limited to:

- Access and infrastructure technologies such as: ISDN, Gigabit Ethernet, SONET, ATM, xDSL, cable, fiber optics, satellite, wireless, and dial systems
- Transport and interconnection functions such as: switching, routing, tunneling, protocol transition, multicast, and performance
- Network management, administration, and security issues, including: authentication, privacy, encryption, monitoring, firewalls, troubleshooting, and mapping
- Value-added systems and services such as: Virtual Private Networks, resource location, caching, client/server systems, distributed systems, network computing, and Quality of Service
- Application and end-user issues such as: e-mail, Web authoring, server technologies and systems, electronic commerce, and application management
- Legal, policy, and regulatory topics such as: copyright, content control, content liability, settlement charges, "modem tax," and trademark disputes in the context of internetworking

In addition to feature-length articles, IPJ contains standardization updates, overviews of leading and bleeding-edge technologies, book reviews, announcements, opinion columns, and letters to the Editor.

Cisco will pay a stipend of US$1000 for published, feature-length articles. Author guidelines are available from Ole Jacobsen, the Editor and Publisher of IPJ, reachable via e-mail at **`ole@cisco.com`**

The Internet Protocol Journal, Cisco Systems
170 West Tasman Drive
San Jose, CA 95134-1706
USA

ADDRESS SERVICE REQUESTED

PRSRT STD
U.S. Postage
PAID
PERMIT No. 5187
SAN JOSE, CA

The Internet Protocol Journal is published quarterly by the Chief Technology Office, Cisco Systems, Inc.
www.cisco.com
Tel: +1 408 526-4000
E-mail: ipj@cisco.com