RIPE Database Update

Shane Kerr, RIPE NCC
shane@ripe.net

Database Statistics

The contents of the RIPE Database change as the information about the resources that it tracks is updated. For instance, new number resources are allocated, contact details are updated, and so on. You can use the Database Consistency and Statistics page to get full detailed information about the current contents of the database:

http://www.ripe.net/db/dbconstat/index.html

Database Contents

The RIPE Database contains about 48% INETNUM objects, 41% contact data (PERSON and ROLE objects), and 6% DOMAIN objects.

The database has had steady growth in the number of objects since the RIPE 48 meeting, adding about 200000 new objects, for about a 9% increase. This contrasts with a 7% increase between RIPE 47 and RIPE 48.

The number of INET6NUM objects continues a large growth, with an annual growth rate of more than 100%. These objects store IPv6 network allocation and assignment information. Over 85% of these objects are assignments, with approximately equal numbers of /48 and /64 networks. Discussion about the appropriate information to record about IPv6 networks is occurring in the Address Policy Working Group.

Queries

The RIPE Database answers queries from anyone on the Internet, using the WHOIS protocol. The server receives an average of about 24 queries per second. This is a continued decline in query rate from the high point of about 30 queries per second in 2003.

The server has three features to reduce data mining. The connecting IP address is used to identify the client. The features are:

- Maximum number of queries an IP can issue at the same time is limited.
- The server will temporarily block an IP that has received a large amount of personal information.
- Being repeatedly blocked temporarily will result in a permanent block.

While still a minority of queries, the number of queries that are blocked has been increasing over time.

The response time to queries has been slowing, especially during peak hours (these are from about 08:00 to about 18:00 Central European Time). Average response time was over 1 second for several weeks, with the worst-case time being over 10 seconds. In *Illustration 1- Whois response time to queries for random IP addresses, and 1-week moving average*, you see a graph showing the server response time to a query for information about a random IP address from the RIPE NCC service region over the past 3 months. The Y axis (height) of the graph indicates the time, and is logarithmic. The slowdown and speed-up are described below in the section on Server Performance.

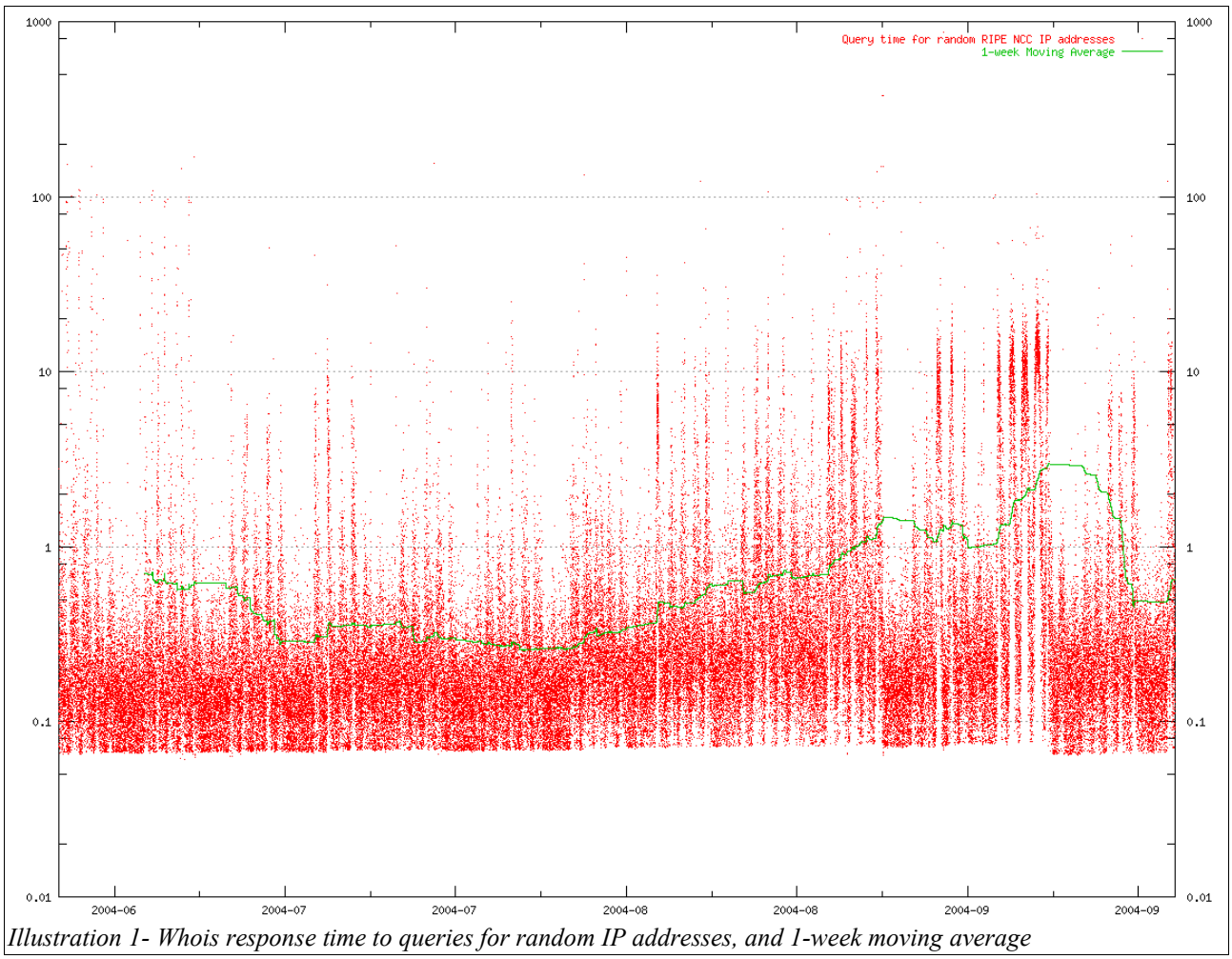

Illustration 1- Whois response time to queries for random IP addresses, and 1-week moving average

Synchronous Update Use

Historically updates to the database have been performed via e-mail. In the last two years, it has been possible to use synchronous updates (syncupdates). In this method, updates are sent to the server and performed immediately. This is useful for building interactive systems. The RIPE Web Updates interface uses syncupdates when updates are actually performed.

There has been a steady growth in the portion of updates that use synchronous updates versus e-mail updates. This is pictured in *Illustration 2- Daily percent of synchronous updates, with trend line.*

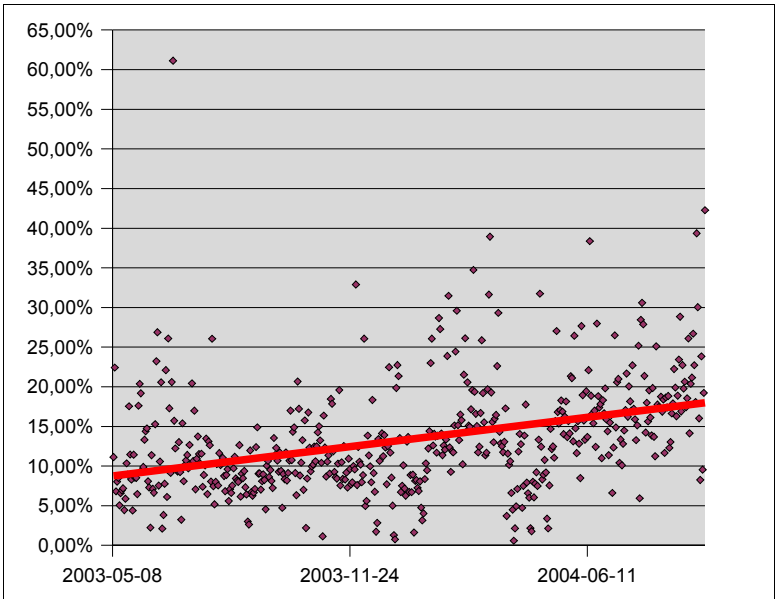

Illustration 2- Daily percent of synchronous updates, with trend line

Database Operations

Hanging Connections

The RIPE server software has a limit on the number of queries that it will process at the same time. Around June 2004, the server started to hit that number regularly. Investigation showed that this was partially due to a large number of clients that would connect to the server normally, but then not submit a query. These "hanging" connections came from a large set of IP addresses, showing no pattern in terms of network or physical location.

Looking at the network at the RIPE NCC showed no configuration that would cause such problems (for example firewall or router issues). Further research showed that there were several patterns among these queries, including clients that would occasionally submit a proper query.

One theory was that a worm was scanning for target hosts and keeping these ports open, however we did not see a similar set of queries on our secondary machine, which answers exactly the same, but is not configured to appear in the DNS.

Another theory was that there was a poorly written client that was causing the problem. The RIPE NCC contacted a few network operators who were using the IP addresses originating the queries. We were careful to select addresses that were not assigned to end-users, so that we could talk to the actual users of the machines. In the cases where we could talk to the users of the machine, there was no clear notion of the way that Whois was being queried; users did report "problems" with the machines in question. While not certain, it is possible that the machines have been exploited, and hackers are attempting to follow connections to compromise other hosts.

In the end, the solution was to double the number of allowed concurrent queries, and consider the hanging queries to be a nuisance that can be ignored if it is not affecting service to other users.

Server Performance

The RIPE Database server has been experiencing some performance problems. Beginning around the middle of August 2004, the server started using 100% of the CPU all of the time. This is expected sometimes, but this was a full-time condition.

One effect of this was a slow-down in query response time. A more important problem was that the mail software on the server, Sendmail, would stop processing e-mail when system load passed a certain limit. On the worst day, acknowledgements to updates took 15 hours to reach some users.

While the main problem was for update users, the source of the load was queries. Updates account for less than 10% of the CPU use on average, and no increase in updates was noticed in the logs. Since there were no changes to the software, operating system, or network around this time, the slow-down is probably related to some change in queries.

Research

Several theories were investigated to explain the slow-down:

- Named queries
  Queries for names, for example "Shane", are far more costly for the server than the more

common queries, which tend to be IP address lookups. However, looking in the logs did not reveal any change in the occurrence of these types of queries.

- Referral queries

  Queries for certain domain names can be referred to another Whois server. In this case, the RIPE Whois server acts as a proxy. However, looking in the logs shows no change to the frequency of these queries.

- HTTP queries

  We noticed a large number of queries that appear to be HTTP queries on the Whois port. At one point about 10% of queries were of this type. These may be a worm, or a poorly written client. However, because of the high rate of queries from a small number of IP addresses, the server had already blocked these hosts, so this had little effect on the server load.

Although the cause of increased load remains elusive, we are still attempting to determine it.

Solutions

The RIPE Database had a primary server and a secondary server, operating in hot-standby mode (see *Illustration 3- RIPE Database Server Configuration*). This was implemented by the primary server advertising the IP address for `whois.ripe.net` in OSPF with a high priority. The secondary server had the IP address configured statically. Should anything happen to the primary server, the address would no longer be advertised, and the secondary would begins to answer queries within a few seconds.

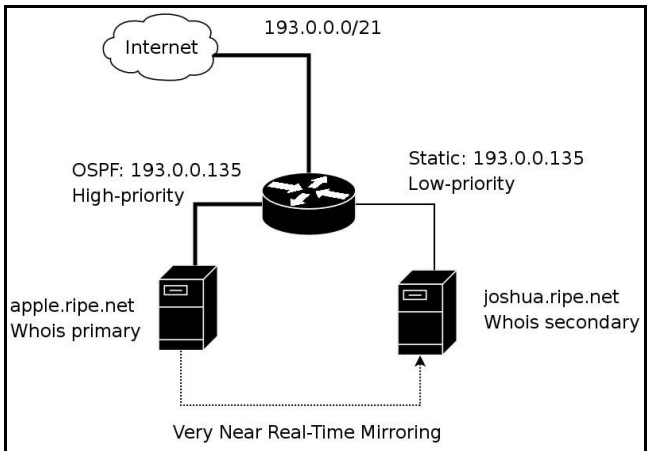

Illustration 3- RIPE Database Server Configuration

The rate of queries for the server is much higher than the rate of updates, occurring at least 300 times more often. This is the reason that query load can affect updates. To insure updates are handled quickly, the server setup was modified to split the primary servers tasks across two servers. In the current configuration, updates occur on one server and queries on another. In the event of a failure in queries, the update server handles both loads (see *Illustration 4- Modified RIPE Database Server Configuration*).

While this setup has fixed the problem with updates, it has introduced a new problem. Because the query server is heavily loaded, changes sometimes appear several minutes after they have completed on the update server. This causes confusion to users, who can get a report of a successful change to the database, but not see it when they look up their objects. Even worse, this can cause certain automatic scripts to fail. As a workaround, a new name has been added to DNS, `whois-update.ripe.net`, which can be queried directly if necessary.

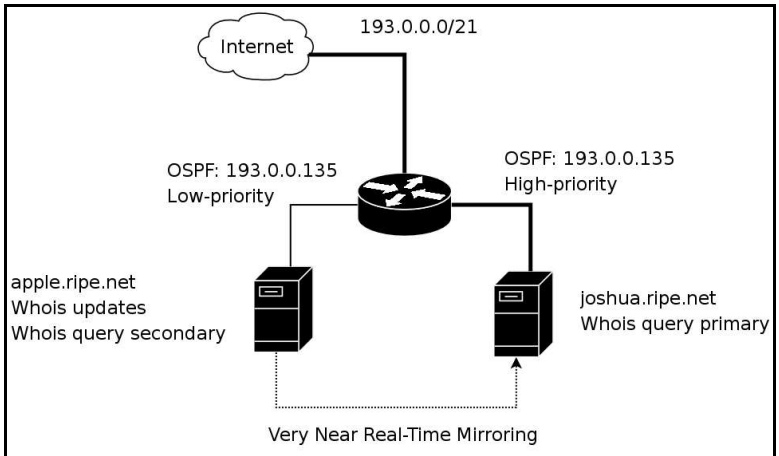

Illustration 4- Modified RIPE Database Server Configuration

While investigation into cause of increased load will help, the long term solution to load issues is to increase the processing power of the Whois server. An upgrade is currently in process. The current hardware is over three years old, and is being replaced with machines around five times faster. Also, the setup is being changed to a cluster (see *Illustration 5- Clustered RIPE Database Server*). This eases administration, and more importantly allows for additional query servers to be added as necessary.

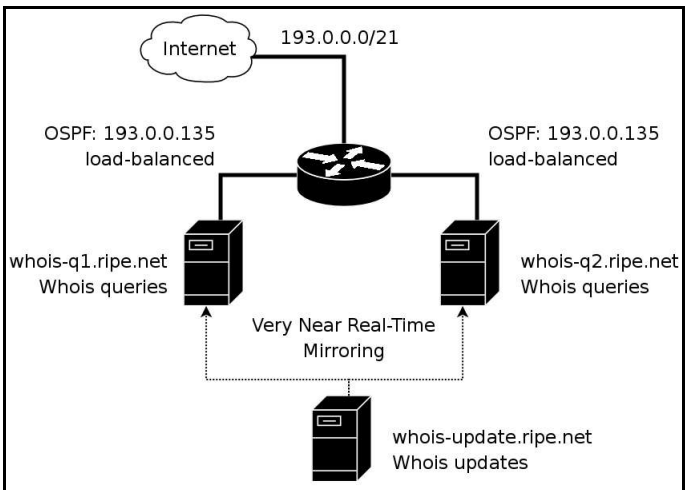

Illustration 5- Clustered RIPE Database Server

RIPE DBM Update

The RIPE DBM is the database support role at the RIPE NCC. It answers all e-mail to <ripe-dbm@ripe.net>. This includes requests for service, like recovering from lost passwords, or information, such as explaining the meaning of error replies from the Whois server.

Beginning in June 2004, the software engineering department has had a full-time User Support Specialist, Vasco Asturiano. Before this, the software developers performed this task on a rotating basis. Having a dedicated RIPE DBM has improved the consistency and quality of support.

Maintainer Creation Change

A significant amount of RIPE DBM time was spent handling requests for MNTNER object creation. Manual intervention was required to insure that the RIPE NCC had some relationship with the people responsible for maintaining data in the database. The restriction was intended to help prevent “hijacking” of unmaintained objects, or inappropriate creation of delegated objects.

Improvements in the database security model have reduced these risks. These changes include:

- Removal of the less secure MAIL-FROM and NONE authentication schemes.
- Protect delegations by using “mnt-by:” attributes if no “mnt-lower:” attribute is present.

In order to avoid delays for database users and to reduce the workload on RIPE DBM, the RIPE Database software was changed to allow users to create maintainers without administrator review. Prior to this, the RIPE DBM would check to see if the user was listed as a contact for some “operational” objects in the database (INETNUM, INET6NUM, or AUT-NUM object).

Projects

RPSLng

RPSL is a standard way to define information about routing policy. RPSLng is an IETF effort to extend RPSL so IPv6 and multicast policies may be defined. For several years, work has been done to complete and refine the draft document, and put it forward as a standards-track RFC. The draft document was approved on 2004-08-10, and is currently waiting for final editing and publishing. The latest draft may be found here:

http://www.radb.net/rpslng.html

The RIPE NCC has updated the RPSLng prototype to meet the latest draft, and is planning on putting RPSLng into production when the RFC is published. The IRRToolSet has also been updated, and was discussed in a separate presentation at the Rouoting Working Group..

rERX

rERX is the "Rapid ERX", or "Rapid Early-Registration eXchange". It is the final stage of an on-going project to transfer Internet number resources that were allocated before the RIR system was established. The status of this project will be discussed in a separate presentation.

Afritrans

A new RIR is being set up for the African region. Part of the current RIPE NCC service region is in Africa, so LIRs and their resources will be transferred from the RIPE Database to the AfriNIC database. The status of AfriNIC will be discussed further in the plenary.

Further KEY-CERT and MNTNER PKI Integration into the LIR Portal

Illustration 6- KEY-CERT automatic creation

It has been possible to use X.509 authentication with the RIPE Database since February 2004. However, to do this users had to create the KEY-CERT objects manually, involving a difficult process of saving certificate information and then copying it into a KEY-CERT object before submitting to the database.

To make the X.509 functionality more accessible for users, some features were added or changed on the LIR Portal. These automate parts of the database interaction. The changes are:

- "Generate certificate" has a check box added to allow the creation of KEY-CERT objects automatically (see *Illustration 6- KEY-CERT automatic creation*).
- "Replace certificate" added, to allow users to revoke an old certificate and generate a new one to replace it in a single operation. This simplifies database interactions, because the same KEY-CERT object can be used with new authentication information. This avoids problems of having to update MNTNER objects that reference X.509 KEY-CERT objects.
- "Add authentication to maintainers" added, which allows users to add their own or other users certificates to maintainers. This makes actually using LIR Portal certificates with the database easier.

Other Database Changes

Several other minor changes were made to the database:

- References with "AUTO-" keyword now followed fully between objects.
  In earlier versions of the code, PERSON and ROLE objects could be created and then referenced in the same update, but it was possible to submit updates with objects in some orders that failed . This has been fixed, so these will work unless there is a circular reference between objects. For details, see Appendix A.

- "auth:" attribute can now be inverse queried.
  You can now use an inverse query, "-i auth", to find references to a KEY-CERT object. This is useful to find where a PGP key is used, for example.
- "fingerpr:" attribute now searchable.
  This is necessary to find X.509 KEY-CERT objects, as the primary key is an arbitrary value.

Future Work

- The CRISP standards are approaching maturity, and the RIPE NCC will be producing a prototype server. CRISP is covered in a separate presentation.
- Any changes necessary to improve abuse reporting will be implemented.
- A new release of the server code has been started, with primary goals being to include all changes since the last release, simplify the installation and administration, and providing an external CVS view of the software.

Appendix A: Details of AUTO- "nic-hdl:" Ordering

The "AUTO-" mechanism provides a way for users to let the database pick a unique identifier for PERSON objects when creating or updating objects in the database. In most cases, it does not matter which "nic-hdl:" is used for contact data, and the database can automate the task of picking one. It can be referenced from other objects in the same update. An example is shown in *Example 1- Typical use of "AUTO-"*.

```
inetnum:   10.0.1.0 - 10.0.1.255
status:    ASSIGNED PA
admin-c:   AUTO-1                  # refers to the object below
tech-c:    AUTO-1                  # refers to the object below
  ...
person:    Contact Person
nic-hdl:   AUTO-1                  # converted by database
  ...
```

Example 1- Typical use of "AUTO-"

Normally the database software processes objects in the order they appear in the update. However, in order to allow updates with "AUTO-" to work, any object that references an object that is identified by an "AUTO-" attribute must be updated *after* that object is created. This is because the database has no way to know what identifier will be used until it is created.

In the past, the method used for this was to scan the objects, and put anything that had a reference to an "AUTO-" object at the end. In *Example 1- Typical use of "AUTO-"*, the PERSON object would be created first, and then the INETNUM would be created, using the "nic-hdl:" that the person got.

There are cases where this algorithm can fail. This can occur any time there is a reference from an "AUTO-" object to another "AUTO-" object. This is pictured in *Example 2- "AUTO-" reference to "AUTO-"* . The reason the algorithm would fail is that the AUT-NUM and ROLE objects would be moved to the end, but the AUT-NUM would still be processed before the ROLE object. Since the AUT-NUM object references the ROLE object, this is an error.

```
aut-num:  AS3333
admin-c:  AUTO-1      # refers to "nic-hdl:" in ROLE
tech-c:   AUTO-2      # refers to "nic-hdl:" in PERSON
  ...

role:     New Role
nic-hdl:  AUTO-1
admin-c:  AUTO-2      # refers to "nic-hdl:" in PERSON
  ...

person:   New Person
nic-hdl:  AUTO-2      # request automatic creation
  ...
```

Example 2- "AUTO-" reference to "AUTO-"

In practice, updates with this pattern were rare. This changed with the introduction of the ORGANISATION object type. Users would submit updates in the same order that the database returns them when queried, usually INETNUM, ORGANISATION, then PERSON objects. In this case, the creation of ORGANISATION objects would fail because the ORGANISATION objects would often reference the PERSON objects.

The solution was rather than reorder the objects, to perform multiple passes on the objects in each update. The actual process is shown in *Listing 1- Pseudo-code for object processing*.

```
def process_objects(object_list):
    number_processed = 0
    while (object_list is not empty) and (number_processed > 0):
        number_processed = 0
        for each object in object_list:
            if object has AUTO- references:
                change all AUTO- references already created
            if object has no AUTO-references:
                process object
                delete object from object_list
                number_processed = number_processed + 1
```

Listing 1- Pseudo-code for object processing

Using multiple passes will work no matter what ordering of objects is used, and will process the list in an order as close to the original order as possible. However, it will still fail in the case where there are circular references. This is possible, for example, if a ROLE object refers to itself using "AUTO-", or two ORGANISATION objects refer to each other and are using "AUTO-". This occurs very rarely in practice, and users can create these objects by performing a series of updates that do not have the circular reference.