DNS Operations

Advanced DNS
Operations & Security

Network Startup Resource Center

Goals

- Go beyond basic DNS administration, focus on service stability
- Identify common operational problems that plague authoritative nameserver administrators
- Identify pitfalls and errors to avoid when changing zones
- Define proper architectures
- Improve availability and reduce the chance of a breakdown of service using active monitoring

Overview

- Tools
  - using dig and interpreting the results
  - doc, dnstop

- Gotchas and common debugging problems
  - RFC1912, 2182, 2870
  - delegation and glue, keeping it up to date
  - inconsistent delegation between parent and child
  - cache effects
  - TTL policy

Overview

- Operations
  - logging using BIND channels
  - monitoring services and zone exports
  - active delegation checking
  - distributed hosting considerations
  - scripting and automation

Tools – using dig

- dig is the *domain information groper*.
- dig is used to query nameservers for information, usually for debugging purposes.
- dig gives you information, and can perfom queries, that most other tools usually used (nslookup, host) don't give you
- dig's output can be confusing the first time one sees it...

Tools – using dig

```
$ dig ns nsrc.org.

; <<>> DiG 9.4.1-P1 <<>> ns nsrc.org
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 40659
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 2

;; QUESTION SECTION:
;nsrc.org.          IN   NS

;; ANSWER SECTION:
nsrc.org.      132391    IN   NS   ARIZONA.EDU.
nsrc.org.      132391    IN   NS   RIP.PSG.COM.

;; ADDITIONAL SECTION:
ARIZONA.EDU.        104458    IN   A    128.196.128.233
RIP.PSG.COM.        89057     IN   A    147.28.0.39

;; Query time: 60 msec
;; SERVER: 212.38.128.2#53(212.38.128.2)
;; WHEN: Tue Nov 27 02:58:37 2007
;; MSG SIZE  rcvd: 108
```

Tools – using dig

- Pay particular attention to the flags and the answer section
- Use dig at the authority of the parent and child zones to control proper delegation
- Do the informations match ?
- Example for cctld.eu.org
  - Identify nameservers for EU.org

```
dig ns eu.org.
```

Tools – using dig

```
;; ANSWER SECTION:
eu.org.          23772    IN  NS  ns0.pasteur.fr.
eu.org.          23772    IN  NS  ns.eu.org.
eu.org.          23772    IN  NS  ns-slave.free.org.
eu.org.          23772    IN  NS  dns3.gandi.net.
eu.org.          23772    IN  NS  auth1.dns.elm.net.
eu.org.          23772    IN  NS  relay-1.ftel.co.uk.
eu.org.          23772    IN  NS  ns1.pasteur.fr.
```

- Ask one of the servers for the NS records for cctld.eu.org

```
dig @ns.eu.org NS cctld.eu.org.
```

Tools – using dig

```
;; AUTHORITY SECTION:
cctld.eu.org.        259200  IN  NS  NS1.CATPIPE.NET.
cctld.eu.org.        259200  IN  NS  NS2.CATPIPE.NET.
cctld.eu.org.        259200  IN  NS  NS1.cctld.eu.org.
```

- Notice the flags for the query, and the way the answers are presented
- Control that the servers for cctld.eu.org return the same information:

  ```
  dig @ns1.cctld.eu.org NS cctld.eu.org.
  dig @x0.dk NS cctld.eu.org.
  ```

- What do you notice ?

Tools – doc

- Checking delegations manually is error-prone and tiresome
- A tool to automatize this particular check exists: doc
- Doc can be installed as a port/package
- Usage:

```
doc [-p] domain.name
```

Tools – doc

- Try using doc – it should be installed.

  ```
  doc -p cctld.eu.org
  ```

Gotchas and common debugging problems

- Logging is the single most useful tool for troubleshooting a running nameserver – we'll see later how to set it up
- Check out RFC1912, 2182 and 2870
- Lame delegations and glue problems can be easy to overlook if the wrong tools are used
- Caching makes this more complicated – problems might appear later.
- Pick a reasonable TTL policy

Gotchas and common debugging problems: caching

- Cache effects
  - Changes can take a while to propagate – plan accordingly
- TTL and SOA policy
  - RIPE has a document for recommended SOA values: ftp://ftp.ripe.net/ripe/docs/ripe-203.pdf

```
example.com.  3600  SOA  dns.example.com. admin.example.com. (
                         1999022301   ; serial YYYYMMDDnn
                         86400        ; refresh (  24 hours)
                         7200         ; retry   (   2 hours)
                         3600000      ; expire  (1000 hours)
                         3600 )       ; neg ttl (   2 days)
```

Gotchas and common debugging problems: caching

- It's common to misinterpret/forget the negative value of the SOA
- ”negative” means ”how long can remember that the record for this query does *NOT* exist”

Operations

- remember to turn off recursion!
- logging
- monitoring service (availability and data)
- active delegation checking
- hosting and architecture considerations

Logging

- Using BIND channels, categories and severities (chap 7.5 of DNS & Bind)
  - The idea is to define *channels* (file, syslog, ...) and the assign *categories* to these channels:

```
logging {
   channel transfers {
      file ”log/transfers” versions 5 size 100M;
      print-time yes;
   };
   category xfer-out {
      transfers;
   };
   category default {
      default_syslog;
      default_debug;
   };
};
```

Logging

- Categories of interest:
  - default
    - a good set of defaults – send it to your syslog
  - lame-servers
    - bad delegation
  - load
    - zone loading events
  - notify
    - zone change notifications
  - queries
    - logging of queries – can be huge!
  - response-checks
    - badly formed answerd, additional information, ...
  - xfer-in / xfer-out
    - events for incoming / outgoing zone transfers

Logging

- Add logging to /etc/namedb/named.conf, and restart named

  ```
  # rndc reconfig
  ```

- Do a zone transfer for a zone from one of your neighbors:

  ```
  # dig @ns.of.neighbor axfr zone.name
  ```

- Look at /etc/namedb/log/transfers

  ```
  # more /etc/namedb/log/transfers
  ```

Monitoring - services

- Monitoring services – why ?
  - make sure that your nameserver is answering correct data, in a timely manner
  - monitor secondaries
  - monitor infrastructure to deliver DNS service (network, servers, ...)
- Tools useful for monitoring:
  - echoping – check service latency and availability
  - SmokePing – graph of reponse times
  - Nagios – service and server monitoring
  - ... many others

Monitoring – zone exports

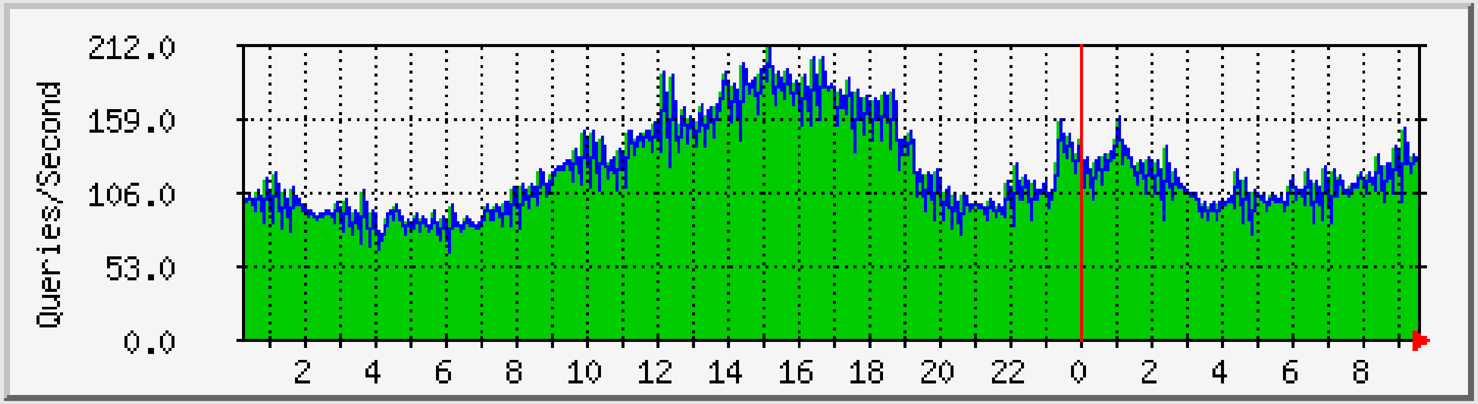
- Monitoring zone export – why ?
  - Avoid publishing incorrect information
  - Avoid publishing incomplete information (truncated zone)
  - Avoid disappearance of your zone! (undetected errors + expire of zone)
- Checks
  - zone change controls before *AND* after publication
    - named-checkzone
  - use EOD markers (data that your export script adds to the zone at the *end* of your zone export job
    - `zonevalid        TXT   ”exported at 20071126 09:54”`

Monitoring – zone exports

- Undetected errors
  - zone fails to load (invalid syntax or inconsistent – CNAME and other data for example)
  - no one notices
  - 2-4 weeks later, the zone expires on the secondaries
  - the zone has disappeared
  - difficult to correlate the problem with the exact cause (unless one has logs)
- Note that if ”rndc reload” is used, BIND will keep the old zone in memory if the new one fails validation

Monitoring - baseline

- Get to know your system
- Using tools such as dnstop, tcpdump, MRTG, establish a baseline for your platform when it is functioning normally
- Identify
  - average queries per second
  - memory usage for named



Monitoring - baseline

- Useful for capacity planning for future growth, and for handling attacks

Delegation checking

- Mostly a policy decision
- Proactive or reactive ?
  - check regularly every delegation
  - or check only when delegation changes
- But there are advantages
  - avoid to field problem reports that are Not Your Problem (”domain XYZ doesn't work!”)
- Some TLDs have a ”Name server registration” procedure.

Secondary considerations

- If you're not already doing it, then make sure your SOA server is a hidden master, not accessible from the rest of the network
- None of your public servers should serve any data that is unique/irreplaceable.
- Normally, all public servers are secondaries (but there are other methods, including secure copy)

Scripting and automation

- You should be try and be familiar with at least one scripting language (Shell, Perl, Python, ...)
- Automate as much as you can
- Run tools like doc, dig to control delegations for critical zones

Questions ?

?