New Features in BIND 8.2.2

Cricket Liu
Acme Byte & Wire
cricket@acmebw.com
www.acmebw.com

Name Resolution

- *Name resolution* is the process by which resolvers and name servers cooperate to find data in the name space
- To find information anywhere in the name space, a name server only needs the names and IP addresses of the name servers for the root zone (the “root name servers”)
  - The name space is an inverted tree
  - The root name servers know about the top-level zones and can tell name servers whom to contact for all TLDs

Name Resolution

- A DNS query has three parameters:
  - A domain name (e.g., *www.acmebw.com*),
    - Remember, every node has a domain name!
  - A class (e.g., *IN*), and
  - A type (e.g., *A*)
- A name server receiving a query from a resolver looks for the answer in its authoritative data and its cache

Name Resolution

- If the name server can't find the answer in its authoritative data or cache, it looks for the "closest enclosing" NS record(s) it has
- For example, say the query is the one we described in the last slide
  - The closest enclosing NS records would be NS records for *www.acmebw.com*
  - The next closest enclosing NS records would be NS records for *acmebw.com*, then *com*
  - The default enclosing NS records are the NS records for the root zone, which (nearly) all name servers have

Name Resolution

- Once the name server has found the closest enclosing NS records, it chooses one of the name servers from the RDATA and sends it a query
  - It sends the same query it received from the resolver (i.e., for *www.acmebw.com/IN/A*), rather than explicitly asking a root name server for the *com* NS records, for example
- The name server may get a referral (in the form of NS records) or the answer in response
- If the name server receives a referral, it follows it

The List of Internet Root Name Servers

```
.                       3600000  IN  NS    A.ROOT-SERVERS.NET.
A.ROOT-SERVERS.NET.     3600000      A     198.41.0.4
.                       3600000      NS    B.ROOT-SERVERS.NET.
B.ROOT-SERVERS.NET.     3600000      A     128.9.0.107
.                       3600000      NS    C.ROOT-SERVERS.NET.
C.ROOT-SERVERS.NET.     3600000      A     192.33.4.12
.                       3600000      NS    D.ROOT-SERVERS.NET.
D.ROOT-SERVERS.NET.     3600000      A     128.8.10.90
.                       3600000      NS    E.ROOT-SERVERS.NET.
E.ROOT-SERVERS.NET.     3600000      A     192.203.230.10
.                       3600000      NS    F.ROOT-SERVERS.NET.
F.ROOT-SERVERS.NET.     3600000      A     192.5.5.241
.                       3600000      NS    G.ROOT-SERVERS.NET.
G.ROOT-SERVERS.NET.     3600000      A     192.112.36.4
.                       3600000      NS    H.ROOT-SERVERS.NET.
H.ROOT-SERVERS.NET.     3600000      A     128.63.2.53
.                       3600000      NS    I.ROOT-SERVERS.NET.
I.ROOT-SERVERS.NET.     3600000      A     192.36.148.17
.                       3600000      NS    J.ROOT-SERVERS.NET.
J.ROOT-SERVERS.NET.     3600000      A     198.41.0.10
.                       3600000      NS    K.ROOT-SERVERS.NET.
K.ROOT-SERVERS.NET.     3600000      A     198.41.0.11
.                       3600000      NS    L.ROOT-SERVERS.NET.
L.ROOT-SERVERS.NET.     3600000      A     198.32.64.12
.                       3600000      NS    M.ROOT-SERVERS.NET.
M.ROOT-SERVERS.NET.     3600000      A     198.32.65.12
```


The Resolution Process

- Let's look at the resolution process step-by-step:

annie.west.acmebw.com

```
ping www.acmebw.com.
```


The Resolution Process

- The workstation *annie* asks its configured name server, *dakota,* for *www.acmebw.com's* address

dakota.west.acmebw.com

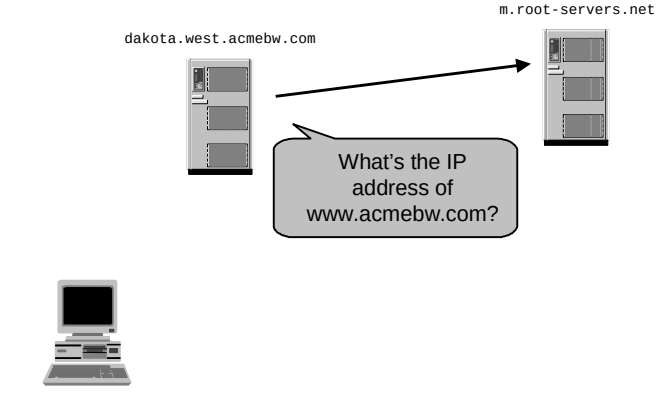
What's the IP address of www.acmebw.com?

annie.west.acmebw.com

```
ping www.acmebw.com.
```

The Resolution Process

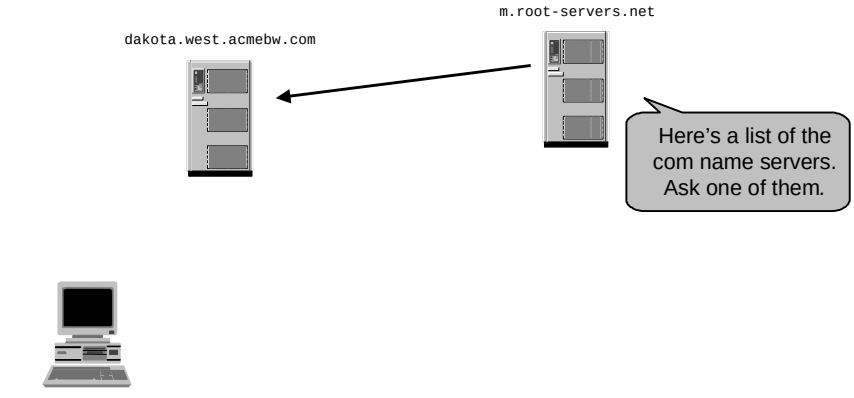
- The name server *dakota* asks a root name server, *m*, for *www.acmebw.com's* address




```
ping www.acmebw.com.
```


The Resolution Process

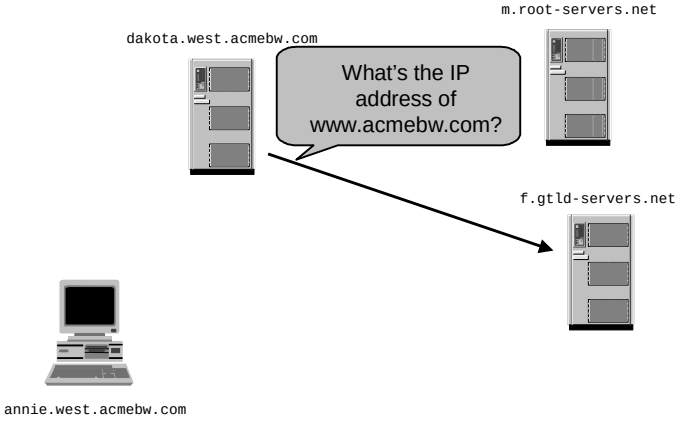
- The root server *m* refers *dakota* to the *com* name servers
- This type of response is called a "referral"




```
ping www.acmebw.com.
```

The Resolution Process

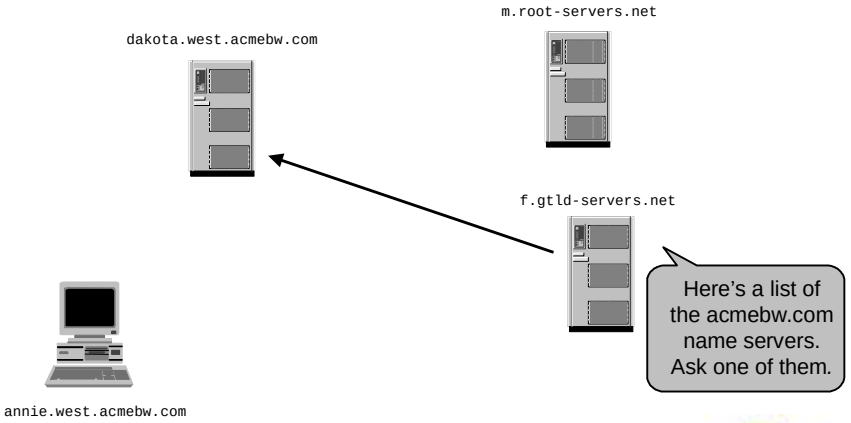
- The name server *dakota* asks a *com* name server, *f*, for *www.acmebw.com's* address




```
ping www.acmebw.com.
```


The Resolution Process

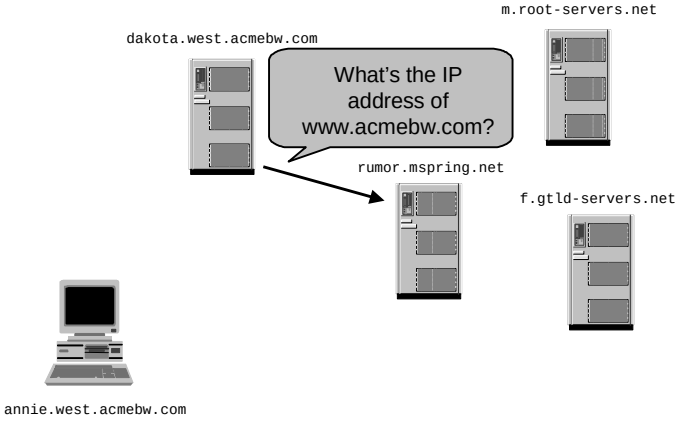
- The *com* name server *f* refers *dakota* to the *acmebw.com* name servers




```
ping www.acmebw.com.
```

The Resolution Process

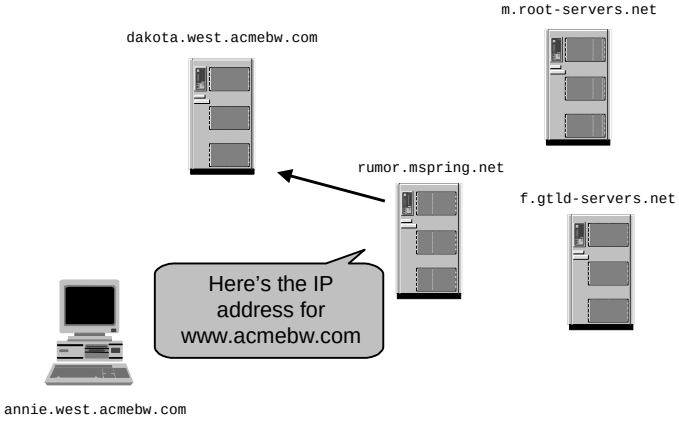
- The name server *dakota* asks an *acmebw.com* name server, *rumor*, for *www.acmebw.com's* address




```
ping www.acmebw.com.
```


The Resolution Process

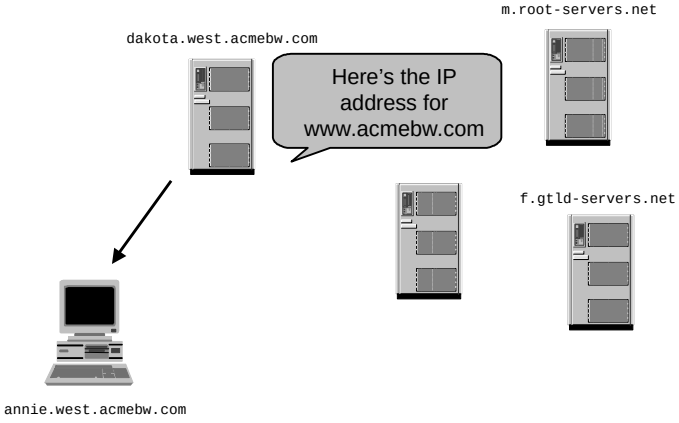
- The *acmebw.com* name server *rumor* responds with *www.acmebw.com's* address




```
ping www.acmebw.com.
```

The Resolution Process

- The name server *dakota* responds to *annie* with *www.acmebw.com's* address




```
ping www.acmebw.com.
```


Outline

- DNSSEC support
- IXFR support
- Enhanced forwarding
- Slave behavior
- *ndc/controls*
- Sortlist
- RRset order
- Negative caching support
- Miscellaneous name server stuff (lame server TTL, blackhole networks, dialup zones)
- Resolver enhancements

DNSSEC

- *DNSSEC* is the DNS Security Extensions
  - DNSSEC provides
    - cryptographic authentication of the origin of DNS data
    - cryptographic checking of the integrity of DNS data
    - all as described in our presentation *DNS Security Extensions*
- BIND 8.2 was the first BIND release to support DNSSEC
  - But BIND 8.2+'s DNSSEC support isn't complete
  - BIND 9 will (eventually) have complete DNSSEC support

DNSSEC Support in BIND 8.2.2

- BIND 8.2.2 provides
  - Support for KEY, SIG and NXT record types
  - Ancillary programs to generate keys (*dnskeygen*) and sign zones (*dns_signer*)
  - Logic in the name server to verify responses cryptographically

New Record Types

- KEY records store public keys associated with zones, hosts or users
- SIG records store digital signatures for an RRset
- NXT records tell a querier which record is lexicographically “next” in the zone
  - These are used to produce authenticated negative responses

Examples of KEY, SIG and NXT Records

```
acmebw.com.  86400 IN    SOA   speakeasy.mspring.net. dnshelp.mindspring.com. (
                               1999021701 ; serial
                               1H  ; refresh
                               10M  ; retry
                               4w2d  ; expiry
                               1D )  ; minimum
             SIG   SOA 3 86400 19990320224141 19990217224141 49292 acmebw.com. (
                   AnnwKotKk32dqs8m0SGczcquji9HJw9ysUveQ4qJNpP+ppyr
                   +J/BrX0= )
acmebw.com.  KEY   0x4101 3 3 (
                   AvqyXgKk/uguxkJF/hbRpYzxZFG3x8EfNX389l7GX6w7rlLy
                   BJ14TqvrDvXr84XsShg+OFcUJafNr84U4ER2dg6NrlRAmZA1
                   jFfV0UpWDWcHBR2jJnvgV9zJB2ULMGJheDHeyztM1KGd2oGk
                   Aensm74NlfUqKzy/3KZ9KnQmEpj/EEBr48vAsgAT9kMjN+V3
                   NgAwfoqgS0dwj5OiRJoIR4+cdRt+s32OUKsclAODFZTdtxRn
                   vXF3qYV0S8oewMbEwh3trXi1c7nDMQC3RmoY8RVGt5U6LMAQ
                   KITDyHU3VmRJ36vn77QqSzbeUPz8zEnbpik8kHPykJZFkcyj
                   jZoHT1xkJ1tk )
             SIG   KEY 3 86400 19990321010705 19990218010705 4932 com. (
                   Am3tWJzEDzfU1xwg7hzkiJ0+8UQaPtlJhUpQx1snKpDUqZxm
                   igMZEVk= )
; ...
acmebw.com.  NXT   ftp.acmebw.com. A NS SOA MX SIG KEY NXT
             SIG   NXT 3 86400 19990320224141 19990217224141 49292 acmebw.com. (
                   AtXkJHUbHfGVFofiIw3qaEZubWGUkVS8pTyWFhAwtDIUEohk
                   KoY2tzY= )
```


New Ancillary Programs

- *dnskeygen* generates a DSA/DSS, HMAC/MD5 or RSA key pair
  - Size specified by the user
- *dns_signer* uses a zone's private key to sign the zone

IXFR vs. AXFR

- Until now, all transfers of zone data have transferred the entire zone
  - Even if just a single record in the zone has changed
  - This form of zone transfer is called *AXFR,* after the query type that initiates the transfer
- Incremental zone transfer, documented in RFC 1995, allows slave name servers to transfer just the changes in a zone
  - Which records have been added
  - Which records have been deleted
  - This form of zone transfer is called *IXFR*

Why IXFR?

- IXFR is important for several reasons
  - It greatly reduces the size and length of zone transfers, on average
    - This is critical for large zones, like *com*
    - This is also critical for dynamic environments, where agents are constantly updating a zone

IXFR Support in BIND 8.2.2

- In BIND 8.2.2, IXFR is configured with three new substatements:
  - *maintain-ixfr-base,* an *options* substatement that tells a master name server to track changes between versions of a zone so that it can send IXFRs
  - *ixfr-base,* a (optional) *zone* substatement that tells a master name server the base filename to use for tracking zone changes
  - *support-ixfr,* a *server* substatement that tells a name server that a particular remote name server supports IXFR

Example IXFR Configuration: Master

```
options {
     directory “/var/named”;
     maintain-ixfr-base yes;
};

zone “acmebw.com” {
     type master;
     file “db.acmebw.com”;
     ixfr-base “db.acmebw.com.ixfr”;
};

server 206.168.119.178 {
     transfer-format many-answers;
};
```


Example IXFR Configuration: Slave

```
options {
     directory “/var/named”;
};

zone “acmebw.com” {
     type slave;
     masters { 205.158.42.225; };
     file “db.acmebw.com”;
};

server 205.158.42.225 {
     support-ixfr yes;
};
```


Enhanced Forwarding

- Before BIND 8.2.1, you configured forwarders as an ordered list of IP addresses
  - The name server simply forwarded a query to the first name server in the list, waited, and tried successive name servers after a static timeout
- As of BIND 8.2.1, forwarders are queried just like any set of authoritative name servers
  - Initially, your name server tries each name server in the list in order
  - But it keeps track of how quickly each name server responds, and favors the fastest name server

What Enhanced Forwarding Means

- You can list multiple forwarders and performance won't degrade (as much) if the first one fails
- You don't need to pay attention to the order in which you list your forwarders
- Fallback from one forwarder to another is faster than with static timeouts

Forward Zones

- In some corporate environments, you need to use one set of name servers to resolve certain domain names, and a different set to resolve other domain names
  - For example, HP may need to use a particular Intel name server to resolve *intel.com* domain names
- *Forward zones* allow an administrator to specify which name servers to forward queries in certain zones to

Configuring Forward Zones in BIND 8.2.2

- Example 1

```
zone “intel.com” {
    type forward;
    forward only;
    forwarders { 134.134.214.6; 143.183.152.22; };
};

zone “mot.com” {
    type forward;
    forward only;
    forwarders { 129.188.136.100; };
};
```


Configuring Forward Zones in BIND 8.2.2

- Example 2

```
options {
    directory “/var/named”;
    forward only;
    forwarders { 15.81.168.10; 15.81.176.10; };
};

zone “.” {
    type hint;
    file “db.cache”;
};

zone “hp.com” {
    type master;
    file “db.hp.com”;
    forwarders { };
};
```


Multiple Master Name Servers

- Before BIND 8.2.1, you could specify multiple master name servers in a slave’s configuration
  - E.g.,

```
zone “acmebw.com” {
  type slave;
  masters { 207.69.231.2; 165.121.2.31; }
};
```

  - BIND simply queried the master servers in order until it received an SOA record
- With BIND 8.2.1, a slave will query all of the master name servers and choose the one with the highest serial number in the SOA record

Masters on Alternate Ports

- In BIND 8.2, you can also specify that the master name server for a zone runs on a port other than 53:

```
zone “acmebw.com” {
    type slave;
    masters port 5300 {
          207.69.231.2;
          165.121.2.31;
    };
};
```


ndc, the Name Daemon Controller

- *ndc,* the name daemon controller, is used to send signals to the name server
  - For example, *ndc reload* sends *named* a SIGHUP
- In BIND 8.2, you can also use *ndc* remotely, across a network
  - The name server now listens for commands on a configurable port, like an FTP or SMTP server does
  - *ndc* can send the name server those commands
  - This is clearly very useful, and very dangerous

What You Can Do With *ndc*

- *status:* Get *named*'s status
- *dumpdb:* Dump *named*'s authoritative zones and cache
- *reload:* Tell *named* to reread *named.conf,* reload primary master zones that have changed and schedule maintenance on slave zones
- *reload <zone>:* Tell *named* to reload just the named zone
- *reconfig:* Tell *named* to check *named.conf* for newly added or deleted zones

What You Can Do With *ndc*

- *stats:* Tell *named* to dump statistics
- *[no]trace:* Tell *named* to log (or stop logging) debugging messages
- *querylog:* Tell *named* to log information about every query
- *start:* Start *named*
- *stop:* Stop *named*
- *restart:* Stop, then start *named*

“Signal Mode”

- To use *ndc*’s traditional mode of operation (i.e., sending signals), use the *-p* switch, followed by the pathname of the file that contains the name server’s process ID:

  % ndc -p /var/run/named.pid

“Command Mode”

- To use *ndc*’s command mode *locally* (i.e., to control the local name server), you don’t need to specify anything
  - *ndc* will use a UNIX domain socket, */var/run/ndc* by default, to communicate with *named*
- To use *ndc*’s command mode across a network, use the *-c* switch and an IP address and port

  % ndc -c 192.168.0.1/54

Restricting Command Mode

- To restrict command mode, you use a new statement in *named.conf, controls*
- Syntax

```
controls {
    [ inet ip_addr
    port ip_port
    allow { address_match_list; }; ]
    [ unix path_name
    perm number
    owner number
    group number; ]
};
```


An Example *controls* Statement

```
controls {
      inet * port 54 allow { localnets; };
      unix “/var/run/ndc” perm 0600 owner 0 group 0;
};
```


Sortlists

- The *sortlist* allows the administrator of a name server to “prefer” certain networks
- The name server “pulls” addresses on those networks to the front of multiple address responses
- BIND 4 name servers supported a sortlist, but early BIND 8 name servers didn’t
  - The ISC’s position was that the name server had no business sorting addresses
- BIND 8.2 supports a sortlist (once more)
  - The BIND 8.2 sortlist lets you prefer different networks based on a query’s source

A Simple Example Sortlist

```
sortlist {
     { localhost; localnets; };
     { localnets; };
};
```


A More Complicated Example Sortlist

```
sortlist {
          { localhost;         // IF   the local host
            { localnets;       // THEN first fit on the
              192.168.1/24;    //      following nets
              { 192,168.2/24; 192.168.3/24; }; }; };
          { 192.168.1/24;      // IF   on class C 192.168.1
            { 192.168.1/24;    // THEN use .1, or .2 or .3
              { 192.168.2/24; 192.168.3/24; }; }; };
          { 192.168.2/24;      // IF   on class C 192.168.2
            { 192.168.2/24;    // THEN use .2, or .1 or .3
              { 192.168.1/24; 192.168.3/24; }; }; };
          { 192.168.3/24;      // IF   on class C 192.168.3
            { 192.168.3/24;    // THEN use .3, or .1 or .2
              { 192.168.1/24; 192.168.2/24; }; }; };
          { { 192.168.4/24; 192.168.5/24; }; // if .4 or .5, prefer that net
          };
};
```


RRset Order

- Before BIND 4.9.2, records in an RRset (records attached to the same domain name with the same class and type) always had the same order
- BIND 4.9.2 introduced round robin, which rotated RRset order in successive responses, e.g.,
  - 10.0.0.1, 10.0.0.2, 10.0.0.3
  - 10.0.0.2, 10.0.0.3, 10.0.0.1
  - 10.0.0.3, 10.0.0.1, 10.0.0.2

RRset Order in BIND 8.2

- BIND 8.2 introduces configurable RRset order
  - *fixed:* records are returned in the order in which they appear in the zone data file
  - *random:* records are returned in a random order
  - *cyclic:* records are returned in round robin order

Configuring RRset Order in BIND 8.2

- Syntax

```
rrset-order {
    class class type type name "FQDN" order ordering;
    […]
};
```

- Example

```
rrset-order {
    class IN type A name "www.acmebw.com" order fixed;
    class IN type A name "ftp.acmebw.com" order random;
    class IN type A name "*" order cyclic;
};
```


New Negative Caching Support

- Before BIND 4.9, name servers couldn't cache negative answers
  - NXDOMAIN (no such domain name)
  - NODATA (no data of the requested type available)
- BIND 4.9 introduced negative caching
  - With a fixed time to live (TTL) of ten minutes for negatively cached information
- RFC 2308 changed the semantics of various TTL fields

RFC 2308 Negative Caching

- With RFC 2308
  - The last (seventh) RDATA field in a zone's SOA record becomes the *negative caching* TTL
  - A record's explicit TTL becomes its "positive" caching TTL
  - The new *$TTL* control statement allows a zone's administrator to set a default TTL

An Example Zone

```
$ORIGIN acmebw.com.
$TTL 3d
@       IN      SOA     speakeasy.mspring.net.    dnshelp.mindspring.com.   (
                        1999040200       ; serial
                        1H               ; refresh
                        10M              ; retry
                        4w2d             ; expiry
                        1h )             ; negative caching TTL
        IN      NS      speakeasy.mspring.net.
        IN      NS      hearsay.mspring.net.
        IN      NS      rumor.mspring.net.
        IN      A       207.69.209.143
        IN      MX      0 @
        IN      MX      10 store-forward.mspring.net.
www     IN      CNAME   acmebw.com.
ftp     IN      CNAME   acmebw.com.
huskymo         1D      IN      A       206.168.119.178
littlemiss      1D      IN      A       206.168.119.179
walterb         1D      IN      A       206.168.119.180
```


Tuning the Negative Cache

- As an administrator, you can also control how long your BIND 8.2 name server caches negative information with *max-ncache-ttl*
  - Default is three hours
  - Maximum value is seven days
- Example

```
options {
    max-ncache-ttl 3600;
};
```


Lame Delegation TTL

- Modern name servers protect themselves against *lame delegation*
  - Lame delegation is when a zone is delegated to a name server that is not authoritative for that zone
  - Following lame delegation lengthens resolution time
  - Consequently, once a name server discovers that another name server is lame for a given zone, it won't query that name server for data in the zone for a period of time
  - In BIND 8.2, that period is configurable

Configuring the Lame Delegation TTL in BIND 8.2

- Example

```
options {
    lame-ttl 1800;
};
```

- Default is 600 seconds
- Maximum is 1800 seconds (30 minutes)
- Value 0 disables lame delegation caching

Blackhole Networks

- With the *allow-query* and *allow-transfer* substatements, an administrator can restrict queries and zone transfers to a particular set of IP addresses
  - Queriers on disallowed networks get a response indicating that their query was refused (RCODE=REFUSED)
- Sometimes an administrator may not want his name server to respond at all (not even with REFUSED) to certain queries and zone transfer requests
  - That's what "blackholing" a network does

Blackholing a Network with BIND 8.2

- Example

```
options {
    blackhole { 10/8; };
};
```


Dialup Zones

- Occasionally, your name server will run as a master or slave to a name server on the far end of a dial-on-demand link
- With *dialup* zones, an administrator can specify that, for a particular zone, the master for the zone or one or more of the zone's slaves are behind dial-on-demand links
  - For master zones, this tells the name server to send NOTIFY messages to the slaves
    - The NOTIFY message should bring up the dial-on-demand link, and the slave should initiate a zone transfer while the link is up

Dialup Zones

  - For slave zones, this tells the name server to check the zone's master name server only once during the *heartbeat interval*
    - Normal zone refreshes are suppressed

Configuring Dialup Zones

- On the master

```
options { heartbeat-interval 60; };

zone “acmebw.com” {
    type master;
    file “db.acmebw.com”;
    dialup yes;
};
```

- On a slave

```
options { heartbeat-interval 60; };

zone “acmebw.com” {
    type slave;
    masters { 207.69.231.2; };
    file “db.acmebw.com”;
    dialup yes;
};
```


Generating Zone Data

- BIND 8.2 supports a new master file control statement, $GENERATE
- $GENERATE allows you to generate multiple resource records that differ only by a number
- Syntax:

```
$GENERATE <range> <lhs> <type> <rhs> [<comment>]
```

- Example:

```
$GENERATE 1-31 $  CNAME  $.0/26.0.168.192.in-addr.arpa.
```

generates

```
1    CNAME  1.0/26.0.168.192.in-addr.arpa.
2    CNAME  2.0/26.0.168.192.in-addr.arpa.
…
31   CNAME  31.0/26.0.168.192.in-addr.arpa.
```


Little Things

- Configurable version
  - Tailors how the name server responds to CLASS=CHAOS, TYPE=TXT queries for *version.bind*

```
options { version “None of your business”; };
```

- Host statistics
  - Turns on statistics tracking on a per-host basis

```
options { host-statistics yes; };
```

- Randomize message IDs
  - Tells the name server to randomize message IDs

```
options { use-id-pool yes; };
```


More Little Things

- Treat carriage returns in zone data files as spaces

```
options { treat-cr-as-space yes; };
```

- Selectively restrict recursion

```
options { allow-recursion { address_match_list; };
```

- Configure the minimum acceptable number of root name servers

```
options { min-roots 2; };
```

- Configure the maximum number of outstanding serial number queries

```
options { serial-queries 4; };
```


Resolver Enhancements

- Configurable name checking
  - Allows administrators to turn name checking off
    `options no-check-names`
- Configurable number of resends
  - Allows administrators to choose how many times the resolver will query the set of name servers in *resolv.conf* before giving up
  - Default is 2
    `options attempts:NN`

Resolver Enhancements

- Configurable retransmission timeout
  - Allows administrators to choose how long the resolver will wait for a response before trying the next name server in *resolv.conf*
  - Default is 5
    `options timeout:NN`
- Name server “rotation”
  - Spreads the resolver’s query load across the name servers listed in *resolv.conf* in round robin fashion
    `options rotate`