ISC Software Guild

BIND Forum Newsletter

Making use of anycast for DNS services

Issue 3

July 2006

Inside this issue:

Making use of anycast for DNS services 1
IPv6 support in BIND 3
BIND Forum member annual meeting 5

In this article, we will be looking into a technique for deploying your name servers that is becoming increasingly popular as it can be used to both increase the number of nameserver for your zones as well as bring them closer to your pool of clients while not having to change configuration in the clients or the zone contents.

Most people are familiar with the notions of TCP/IP unicast, multicast and broadcast traffic. Anycast is just traditional unicast traffic with a twist regarding where packets are delivered, thanks to thoughtful use of both your internal and the Internet's routing systems.

The term anycast was first introduced in RFC 1546 [1] to define an IP service where the same IP address is given to several hosts delivering the same service and it is not of real importance which one delivers the service, though one usually hopes that it will be the *closest* one to deliver that service.

This technique relies on the routing system, either an interior gateway protocol (IGP) such as OSPF or IS-IS, or an exterior gateway protocol (BGP on the Internet) to exercise its usual routing decision process to deliver packets to a destination IP when several paths are available. The routing system will not know anything about whether the IP address represents one host present at several sites or several hosts, located in different places, that happen to be using the same IP address.

Having many servers respond with the same address has several potential benefits:

- It allows replication of servers and installing them close to pools of clients, improving network response times for the clients in a way that is transparent.
- It allows to bring down some servers, for instance during maintenance windows, and have the routing system automatically direct clients to other operational servers, therefore enabling maintenance tasks without service interruption
- It allows service continuation during unexpected server downtime. In fact, if a pool of clients has a server close-by that can provide them with full service, e.g. lookups for DNS records of a particular sub-domain, clients may continue to conduct local network interactions even in the even of a network partition event such as the failure of the external data link.

Deploying anycast for DNS

DNS is particularly suited to anycast as it runs over UDP, which, like IP is stateless, and makes use of mostly single packet exchanges between the client and the server, with each query

being independent of the previous ones. These characteristics make it immune to the single biggest caveat regarding the use of Anycast (see section on Pitfalls below).

How anycast is used in your DNS service will depend on where your servers are located.

We will start by assuming all the DNS servers that are going to be anycasted are inside the same autonomous system, in a network all kept working together by use of an IGP, for instance OSPF.

The simplest form of anycasting is done when all servers are behind a router or set of routers that have direct access to all servers with equal metric, e.g. all servers are connected through switches to the routers making a single broadcast domain.

The servers are configured with the anycast address, which is the service address, on one of its interfaces and a routing (OSPF speaker) process is configured on the same interface to advertise reachability for the service address to the OSPF routers it will peer with.

The routers will receive several of these announcements and, if done carefully, they will have the same metric. Therefore, the routers will consider all paths to be equal to reach the service IP address. Modern routers, when faced with several identical ways to reach an IP address will use Equal Cost Multi-path routing. There are two different strategies the routers can use to decide which path to choose, and in both all paths are used. Per-packet-load-balancing will see the router send each new incoming packet destined to the service IP follow a different path, as the router tries to balance the traffic among all paths.

The other, much better, option is to have flow-based-balancing, where the router will use the (origin-IP, origin-port, destination-IP, destination-port) tuple to generate a hash, which will determine the path to follow.

When used with a sufficiently large number of clients this last strategy results in load balancing as good as per-packet-balancing, with the advantage that traffic becomes deterministic and is therefore easier to rationalise and debug in case of service issues

Of course the servers will also need a regular unicast IP address on the same or other interface to enable remote administration, otherwise you will not be able to single out a specific server to talk to (that is the whole point of anycast, after all).

The above can be extended when servers are located in various remote areas of the network, except in this case not all paths will have equal cost. This will see routers deliver packets to the closest server from each of their points of view. It will also provide fail-over if a given server is taken down, with an alternate path taking over almost instantly.

If the servers are outside your network, or cross over one or more ISPs, for instance at remote branches, the technique used is similar but the routing protocol to use is BGP, possibly in combination with internal OSPF.

Routers are told to advertise reachability to their BGP peers for the service IP address from various points that have access to the various servers.

The process is similar to the previous one with a few special considerations:

- ISPs tend to filter out network prefixes that have fewer addresses than a /24 does. This means effective use of this technique may need the alloca-

tion of a /24 to this purpose.

- Various different services can be anycasted inside the same prefix but it would be unwise to have non-anycasted services within that prefix as the packets that ended up at locations where the service is not present could not be dealt with, creating a blackhole.

The advantage over Layer 4 Load balancers

Networks already have routers in them, so by using anycast no new hardware is inserted in the network, increasing reliability. In addition, anycast is a pure routing technique and no risky packet analysis takes place to determine where the service is provisioned.

Anycast is not a universal solution but reasoned use of anycast can yield more reliable network services.

Pitfalls

Given the dynamic nature of routing systems and their changes in response to evolving network conditions, it is possible that long-lived sessions (e.g. those over TCP) may be initiated to a given server and then, through routing changes has its packets delivered to a different server afterwards. This second server would have no information about the TCP connection and would therefore respond back with a TCP RESET, ending the client side's connection.

If you have a very dynamic routing system or the path between your clients and your servers changes frequently, using anycast for long lived sessions will not be an improvement for your clients as they may see an increased number of session resets, timeouts and other disruptive behaviour.

However, if your routing system is stable, then the occasional path change will not be a bother and the benefits mentioned earlier, may outweigh the drawbacks for long-lived sessions.

References

[1] C. Partrdige, T. Mendez & W. Milliken, RFC 1546. Host Anycasting Service

BIND IPv6 features

In this article, we will be looking at IPv6 support in BIND (8 and 9) from both a protocol and a server administration point of view.

Both the current versions of BIND 8 and BIND 9 have support for DNS records relevant to IPv6 as well as configuration controls to allow server administrators to control the software's behaviour.

Protocol

Both BIND 8 and BIND9 support the current DNS standards defining support for IPv6 forward and reverse queries.

With the deprecation of A6 Resource Records (RRs) by RFC 3363 [1] forward queries are only performed through use of AAAA (quad-A) RRs and that is what BIND currently supports.

A typical AAAA RR looks like

```
$ORIGIN example.com.                \
host   3600 IN    AAAA 2001:db8::1
```

As for reverse queries, BIND supports the traditional *nibble* format used in the ip6.arpa domain, as well as the older, now deprecated ip6.int domain. BIND 9, but not 8, formerly

supported the *binary label* (also known as *bitstring*) format. The support of binary labels, however, is now completely removed according to the changes in RFC 3363 [1].

Applications in BIND 9 do not understand the format any more, and will return an error if given such data. In particular, an authoritative BIND 9 name server rejects to load a zone file containing binary labels.

A reverse DNS data entry for the example above would look like

```
$ORIGIN 0.0.0.0.0.0.0.8.b.d.0.1.0.
0.2.ip6.arpa.

1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0      \
14400 IN PTR host.example.com.
```

Each number in the label corresponds to a nibble (4 bits) in the IPv6 address.

Transport and configuration

BIND 9 configuration files define IPv6 addresses for use in multiple configuration clauses. Configuration clauses that accept an IP address, will accept both an IPv4 or an IPv6 address.

The degree of IPv6 support in BIND 9 has grown since the first release to what it is today, where IPv6 and IPv4 can be used indistinctively and support for both is identical.

BIND 8 has followed a differente path, getting full IPv6 support with the release of BIND 8.4, thanks to the support of the WIDE project. [2]

Both support IPv6 addresses in any address_match_list, as used in acls, etc.

For some time BIND 9 had a problem with the listen-on-v6 clause only being able to take the value "any", meaning it would either list on all IPv6 interfaces or on none of them. This feature now works properly an allows administrators to have the same level of control as they are used to with IPv4.

When { any; } is specified as the address match list for the listen-on-v6 option, the server does not bind a separate socket to each IPv6 interface address as it does for IPv4 if the operating system has enough API support for IPv6 (specifically if it conforms to RFC 3493 and RFC 3542). Instead, it listens on the IPv6 wildcard address. If the system only has incomplete API support for IPv6, however, the behaviour is the same as that for IPv4. A list of particular IPv6 addresses can also be specified, in which case the server listens on a separate socket for each specified address, regardless of whether the desired API is supported by the system.

In the absence of specific clauses, a BIND server running on a dual stack host will use both IP stacks. A server administrator can choose to select whether to provide service over only one stack, either IPv6 or IPv4, using

```
listen-on-v6 { none; };
```

or

```
listen-on { none; };
```

respectively.

Other clauses allowing fine grained control include:

- query-source-v6
- transfer-source-v6
- notify-source-v6

One item of note is the match-mapped-addresses clause, particularly useful with some Linux kernels, where the mapping of incoming IPv4 connections to use IPv6 mapped addresses could lead to the failure of an address_match_list, leading to

unexpected behaviour.

Both rndc (BIND 9) and ndc (BIND 8) are also totally capable of talking to the server processes over IPv6.

Finally, other utilities that ship with BIND, such as `dig` and `queryperf` have also seen IPv6 support added.

To summarise, BIND is today perfectly tuned to the provision of IPv6 service, be it in a dual stack environment or in an IPv6 only environment, with no need to rely on any IPv4 service to carry out its job.

References

[1] Representing Internet Protocol version 6 (IPv6) Addresses in the Domain Name System (DNS), RFC 3363, R. Bush, A. Durand, B. Fink, O. Gudmundsson, T. Hain

[2] WIDE project. http://www.wide.ad.jp

[3] Basic Socket Interface Extensions for IPv6, RFC 3493, R. Gilligan, S. Thomson, J. Bound, J. McCann, W. Stevens

[4] Advanced Sockets Application Program Interface (API) for IPv6, RFC 3542, W. Stevens, M. Thomas, E. Nordmark, T. Jinmei

Note from the editor

The article on BIND 9.4 performance due to be published in this issue is now scheduled for the next issue and in its place we have published an article on IPv6 support in BIND.

BIND Forum member meeting

The 2006 BIND Forum member meeting will take place on Sunday July 9th, 6:15pm to 7:45 pm.

Venue information

Palais des Congres de Montreal
201, Avenue Viger Ouest
Montreal, Quebec, Canada

Meeting room: 519B

Agenda

a) Welcome
b) Review of current BIND version
c) Report on current ongoing work in BIND
d) What gets into BIND, how and when.
An explanation of ISC's BIND development process
e) Presentation and discussion of the BIND roadmap
f) Any Other Business

A current draft of the BIND roadmap was recently posted to the forum_members@isc.org mailing list, to which all BIND Forum members are subscribed.