What to Look for in a Quality DNS Service

Table of Contents

Introduction 2
Primary and Secondary DNS Services 2
- API 2
- Web Client 3
- DNS Zone Transfer 3
- Dynamic Updates (DDNS) 4

Transport 5
- Anycast 5
- Unicast 5
- Peering Diversity 5
- Resilience against DDoS attacks 5
- Query latency 5

Diversity 5
DNSSEC 6
- Enabling DNSSEC 6
- Updating DS record 7
- Keeping DS records up to date 7

DNS Protocol Specific Recommendations 7
- RFC Compliance and the risk of non-compliance 7
- TTL 8
- SOA Timers 8
- DNSSEC Parameters 8
- TSIG 8
- NTP 8

Customer Rights 9
Support, SLA and Price 9
- Support 9
- Service Level Agreement 10
- Price 10

Glossary 10
About Internetstiftelsen 12
About Netnod 12

Introduction

This document is a joint work between Internetstiftelsen and Netnod. The goal is to outline some of the critical elements you should look for when setting up or procuring a DNS service for an enterprise.

Our intention in providing this outline is to share the knowledge we have gained over many years working with DNS services, customers and protocols.

If you would like to discuss this in more detail, you can always arrange a discussion with one of our DNS experts by contacting us here:

Internetstiftelsen: info@internetstiftelsen.se
Netnod: info@netnod.se

Primary and Secondary DNS Services

All domains need a primary DNS server which holds the resource records for that domain and can answer DNS queries for it. To ensure faster load times, better load balancing and redundancy, many domains also use secondary DNS servers. Secondary DNS servers contain copies of the relevant information held in the primary DNS server. This data is regularly copied across from the primary to the secondary DNS servers through zone transfers (AXFR or IXFR).

When assessing your options for both primary and secondary DNS servers, there are some important elements to consider. The sections below outline your various choices.

You have a number of options for how you manage the data in your zones and the way that this data is stored, updated, authenticated and copied between various primary and secondary DNS servers.

API

There are basically three possible mechanisms for managing zone content in a primary DNS service:

- bulk import of zone content edited and managed elsewhere
- web-based editing of zone details
- fine-grained API access to manage details of the zone contents

In the end, all of these fall back on an API to manage updates to zone data in the service backend (the web editor will have to send results back via the API and the bulk import would use API endpoints for bulk import).

Today, the usual case is a RESTful API using JSON as the transport encoding. There are also older XML-based APIs still in use, but those are not recommended. Good APIs map as closely as possible to the underlying DNS data structures, i.e. using Fully-Qualified Domain Names (FQDNs), including a terminating dot and provide support for managing data in DNS Resource Record (RR) sets.

You should consider whether the API allows write access to all parts of the zone data or if some parts (that are perhaps automatically managed by the provider, like the SOA serial) are unavailable. In the latter case, it is helpful if all data has at least read access.

A final point to consider is the security model used. In simpler APIs a static token is used, which is not sufficient today. Slightly better is a dynamic token valid for a limited time before a new token has to be acquired. The authentication token may be augmented by some form of 2-factor authentication (2FA), like a restriction on from which IP-addresses the API is accessible. Note, however, that more common forms of 2FA (including third party Authenticator service like Google Authenticator, Yubikey or similar) may be complex to use in practice if the API is to be accessed by software systems independent of a person. In such cases, the IP address restriction may be sufficient.

Web Client

In practice, every provider of a DNS primary service has “web access” as one of the available mechanisms (typically in addition to a management API). The web client access is typically used for low-volume management, i.e. for customers that have only a small number of zones and only an infrequent need to change any content. High-volume management is in practice only possible via API access.

But, in the cases where web-based management is convenient, it is important that the interface is as “usable” as possible. For instance, while only using FQDNs in the API makes sense, it typically does not make sense to have to type the full FQDN for “www.company.com.” when editing the company.com zone. We already know which zone is being edited, so the “company.com.” part is mostly redundant and including it only increases the risk of introducing errors. Hence, a good web interface will minimise typing by filling in everything that can be done automatically. It will also give an immediate sanity check to provide feedback on input that doesn’t make sense in the context of the zone being edited.

However, even so, it is sometimes difficult to understand the exact consequences of different changes in the web editor even if the customer fully understands DNS (which is not always the case). As most zone files are quite small, a good complement is therefore a preview mechanism where the complete, fully spelled out zone file is presented for inspection prior to committing it for publication.

DNS Zone Transfer

The interface between the customer and the DNS provider can also be implemented in the form of DNS zone transfers. This is a classic, very precise and well-defined interface, which sees a lot of implementation, but which requires a full DNS setup on the customer side.

AXFR/IXFR (Asynchronous/Incremental Transfer) has been used for decades and is a very stable mode of operation.

In this scenario the customer operates a full DNS server which acts as a hidden primary server. The DNS provider runs a secondary server that uses the DNS’s classic zone transfer mechanisms (AXFR) to obtain the zone information. Modern software adds the option of incremental zone transfers (IXFR) which just transfer changes to the zone between the two, rather than the entire zone, each time an update is made. This saves bandwidth and effort for large zones, but has less effect on smaller zones. DNSSEC-enabled zones have properties

that may affect how useful IXFR is primarily because the signature records have timers that sometimes necessitate the transfer of a lot of data.

The AXFR/IXFR model is designed such that the secondary server pulls the zone data from the primary server at regular intervals typically in the range of several hours. This can cause significant delay in propagation of updates. All modern DNS software therefore uses the updated standard which includes NOTIFY messages from the primary to the secondary to trigger more or less immediate updates. The propagation delay is therefore negligible for zones of moderate size.

The transfers are often secured by DNS transaction signatures (TSIG) which require a manual key exchange between the two parties beforehand. TSIG uses symmetric keys, which means that the same key is shared between the two parties. A consequence of this is that each customer must have its own key and key management becomes an issue for a DNS provider with many customers.

The main drawback of this approach is that the customer needs to operate a full-blown DNS service, and therefore needs to have the knowledge to do that. In addition, the configuration is based on IP addresses so the addresses on both sides must be static. The exchange of TSIG keys is also a complicating factor.

Dynamic Updates (DDNS)

Updating DNS data using Dynamic Updates over the DNS protocol is a well-defined interface and quite reliable. It is available in various implementations “under the hood” on Macintosh and Windows machines, for system-related information, but less so in relations between domain holders and DNS providers. The probable reason for this is that HTTP(S) traffic is more likely to traverse firewalls unscathed and that there are fewer code libraries to rely on. Knowledge about HTTP transactions is also much more widespread than that of DNS transactions.

DDNS uses the DNS protocol’s built-in “update” transaction to send updates from a client to a primary DNS server which then distributes it to other authoritative servers for the zone. The updates are typically secured by DNS transaction signatures (TSIG) which are widely implemented, stable and reliable. The drawback is that the updating client and the receiving primary server share the same key. This means that each client needs to have a separate key, and the key management at the primary server becomes a scaling issue. One alternative – transaction keys (TKEY) – uses asymmetric keys which mitigates this problem, but which doesn’t see the same levels of implementation.

While it’s theoretically possible to have the DDNS client maintain DNSSEC information and send it in its updates, it’s not a practical way to do it especially with a busy zone that receives lots of updates. Much effort is saved by having the primary server take care of the DNSSEC pieces, and having it sign the zone on-line. With TSIG/TKEY-signed updates and a primary server that does on-line signing, security can be maintained at a quite satisfactory level.

Using DDNS means that the receiving server doesn’t have to rely on other protocols or mechanisms. The entire process can be implemented in the same piece of software. As an added bonus, DNS updates are small and quick transactions which don’t consume a lot of computing resources.

Transport

When assessing the networking aspects of a DNS service one of the most important considerations is whether the provider has good connectivity both over IPv4 and IPv6.

You must also be sure that the service can ensure diversity and here there are a number of considerations that include: anycast, unicast, peering diversity, query latency and resilience against DDoS attack.

Anycast

You should aim for one IPv4+IPv6 address pair served from multiple locations running independently. Each anycast site can exchange traffic with multiple peers over BGP and each client can send its queries to the closest anycast site. By using a provider running a good anycast service, you can achieve both geographic and network diversity.

Unicast

You should aim for one IPv4+IPv6 pair served from one location (still potentially with local redundancy). By using multiple unicast services, you can still achieve network diversity as long as the unicast nodes are connected to different providers. This puts more responsibility on you to make sure that the providers are diverse enough: for example, that they use different upstream providers.

Peering Diversity

When choosing between using only unicast or going for an anycast provider, you should consider peering diversity. Using multiple unicast servers located in multiple locations and with different upstream providers could provide sufficient diversity for your service. Using an anycast provider would give you a lot of these things automatically.

Resilience against DDoS attacks

Using an anycast provider also automatically gives better resilience against DDoS attacks. While it is more likely for a DDoS attack to overwhelm a single unicast server, an anycast network is more easily able to deal with malicious traffic and ensure legitimate DNS queries are rerouted to the best available location.

Query latency

You can reduce query latency by having name servers as close as possible to the users. This is easiest to achieve by using an anycast provider with sites distributed all over the world.

Diversity

Some zones will see queries from all over the world; others from a much more limited geographic area. But regardless of where the queries are coming from, it is of crucial importance that they are able to reach the DNS service (i.e. the name servers providing the service) and that the servers are able to send the response back. To reduce the risk of queries not getting responses, avoid “single points of failure”. This, however, is no trivial task, but

a good DNS service provider should have good answers to how they achieve geographic diversity, Network topology diversity, provider diversity and software diversity.

Geographic diversity

The goal is to be able to cater to queries from resolvers far apart without channelling them all through basically the same network pipes in the end.

Network topology diversity

This is very similar to the previous issue. The goal is to avoid overdependence on a small number of network components (regardless of whether the component is a particular fibre, router or even ISP).

Provider diversity

The DNS service provider is also a potential single point of failure. If the provider has a catastrophic failure in their central systems, that may affect all of their name servers regardless of how well diversified the actual servers are. On the other hand, quality providers design for minimising the consequences of such central failures. In that case, they should be able to describe how such risks have been mitigated.

Software diversity

In theory, all the involved software should work as intended. In practice, it doesn't. All software has bugs and the typical bug is the one that has yet to be discovered and where, therefore, the impact is unknown. The most sensitive part of the software in a DNS service is obviously the name server. However, other systems are also important (the software providing the management API, the software providing statistics, etc). Some providers are able to offer in-house software diversity by serving the customer zones from a mix of name server implementations. In other cases, it is possible to achieve a higher degree of software diversity by using multiple DNS providers that are using different software platforms. The latter alternative, however, requires that the DNS service providers are willing to disclose the software platform they use.

DNSSEC

Regardless of whether you currently sign your zones, the DNS service implemented or provided must provide DNSSEC. This will ensure that changing needs or requirements can be quickly implemented. Signing your zone also enables the secure use of DNS-based application security such as DANE, DKIM, DMARC and SPF.

The DNS Service should follow best current practices concerning algorithm support and other DNSSEC Parameters.

Enabling DNSSEC

The procedure for enabling DNSSEC should be simple and clearly documented.

Updating DS record

Responsibility for updating the Delegation Signer (DS) record in the parent zone should be well defined and clear in the process description.

Keeping DS records up to date

Responsibility for keeping the DS record updated in the parent zone in regards to DNSKEY rollovers and algorithm changes should be clearly documented.

DNS Protocol Specific Recommendations

RFC Compliance and the risk of non-compliance

The DNS specification is quite old. Many additions, clarifications, and extensions have been "retrofitted" onto the base specification over the years. This means that there is no "single source of truth" for the DNS protocol. One has to wade through dozens of documents to get the whole picture, and, as always, the devil is in the details. The basic parts of the DNS are quite well understood, but as new threats appear from various miscreants, the ways the DNS is operated varies by place and over time. Hence, "RFC compliance" is a moving target.

That said, a lot of effort has been put in by conscious standards developers and software vendors to make the central pieces of the global DNS system interoperable and stable.

When it comes to the DNS protocol itself, there are standards, and they are normally rather well defined. This means that it is possible to build software that follows the specifications. When things are less clear, there are knowledgeable and willing people to turn to for advice. The DNS community is active and competent. This means that it is possible to develop DNS software that is compliant and therefore interoperable. The problem is the plethora of specifications that need to be found and followed. This complexity may tempt less serious implementers to stray from the path or take shortcuts. Doing so will inevitably lead to interoperability issues.

There is also the problem of using other protocols than the DNS to carry DNS-related information, as, for example, using HTTP/HTML-based updates. For this, there is no formal standard so interoperability becomes a much more complex task. Interacting with a variety of local implementations will inevitably lead to complexity in the central end, whether that is the client interacting with different server implementations or a server interacting with a multitude of different clients. The $N(N-1)/2$ problem soon becomes very apparent.

When designing a DNS service there are a number of trade-offs. One overarching trade-off is that of simplicity vs. flexibility. Customers come in different flavours: some are ignorant of DNS and just want it to work; others want to have full control over the zone content and how it is served. The best approach is probably to provide reasonable default values that can be modified by the customer upon request.

Various parameters in the DNS configurations influence the behaviour of the system, and how the service for a specific zone is provided. When looking to implement or procure a DNS service, sufficient attention must be given to these. Some of the important parameters are: Time-to-Live (TTL), Start of Authority (SOA) Timers, DNSSEC parameters, and DNS

transaction signatures (TSIG). It is also important to ensure that all involved servers are synchronised, for example, by using Network Time Protocol (NTP).

TTL

The TTL of DNS records is used to balance DNS query traffic against rate of data change. As the DNS provider has little control over how often the customer needs to change the data, the customer must be able to set the TTL to its preference—per each record set (as they can have vastly different properties). Thus, any update mechanisms used must be able to carry this information.

If the provisioning system puts limitations on the TTL values, it's very important that the effects of these limitations on the systems and customers are well understood.

SOA Timers

The timers in SOA records, which determine the behaviour of the secondary servers in their relation to the primary server, are likewise subject of discussion. Originally they were designed to set the tempo of updates between the primary server and its secondaries. Modern software sidesteps this with the use of NOTIFY messages, which shortcut these timers so they usually play a very small role in modern DNS deployments. However, setting them to unreasonable values will cause operational problems by, for example, putting unnecessary load on the servers. Therefore the use of very short time values (especially 0) should be strongly discouraged or disallowed.

DNSSEC Parameters

DNSSEC (security additions to the DNS protocol) is a way to add integrity and authenticity to the DNS protocol. With the use of DNSSEC the client can verify that the data received indeed comes from the expected source and that it hasn't been tampered with during its journey across the network. DNSSEC adds a fair bit of complexity to the system, but is an important security feature that also enables DNS to carry security-related information such as certificates and credentials for network devices and users.

DNSSEC is a field where much consideration must be given to the details. Informed choices have to be made regarding cryptographic algorithms, key lengths, lifetimes for keys and signatures (also taking TTLs into account), signing jitter, on-line or off-line signing, which algorithm to use for denial of existence, etc.

TSIG

TSIG is a good mechanism for securing various DNS transactions. It requires some attention to set up as establishing TSIG relationships can require some work. Any DNS service provided should include the provisioning of TSIG keys "out of band". A web API is probably the most common way to implement this. It should be possible for the customer to use different TSIG keys for different services. It must also be possible for the customer to initiate the switch from one (old) key to another (new) key.

NTP

All crypto-based DNS transactions (including TSIG, TKEY, and DNSSEC in all their shapes) have time as a factor. Timestamps are woven into signatures to prevent replay attacks. This means

that all machines involved in generating and/or validating signatures must operate with a correct understanding of the current time. Failing that, they may see old (invalid) signatures as valid, or correct and current signatures as invalid. This is a common source of interoperability problems.

As a consequence, system clocks on all involved systems must be synchronised. The Network Time Protocol (NTP) makes that easy and, with its recent security additions (NTS), also secure. With NTP, system clocks can be kept in sync to within a fraction of a second of UTC, which alleviates all time-based problems when dealing with security-related protocols.

Reliable public NTP (and NTS) time services are readily available on the Internet at no cost.

Customer Rights

There is always a compromise between a service provider (that would like all customers to be the same) and a customer (that wants the provider to do things exactly the way they want). However, there are certain things that are more important than others for the customer to retain control of and this is mostly connected to the zone content. This issue is, therefore, of less importance when discussing a pure secondary service when the zone contents are, by definition, fully under control of the customer when using their own primary setup.

But when procuring a primary service, the maintenance of the zone contents is very important. Among things to consider are:

- TTLs on individual records. Some services have “standards” that cannot be overridden. Sometimes for good reasons (for more see p.8).
- Does the service support the publication of CDS and CSYNC records for automatic maintenance of the delegation from the parent zone?
- Does the customer have the ability to control the individual parameters for DNSSEC signing of the zone, or are they shared between multiple customers?

Support, SLA and Price

Support

There will be moments when support is needed. It may be a technical question about your DNS setup or it could be that the service has issues or, in the worst case, is not working.

Multiple channels of communication can be important, such as web forms, email and phone contact to support, helpdesk or NOC staff. This goes both ways. Make sure you have the possibility to provide your contact details to the DNS provider.

Depending on the importance of the zone(s), 24/7 support could be crucial. You need to make sure the terms of this support are clearly understood.

Support in the local language and local time zone can be a factor that should also be considered.

Service Level Agreement

If the DNS service does not provide a Service Level Agreement (SLA), you are on thin ice. So, what should you look for in an SLA for a DNS service? The following elements are crucial:

- Uptime
- Accessibility to respond to DNS queries
- Number of sites (and number of sites from different regions)
- Minimum number of sites up and running
- Propagation time for the zone file or zone changes to sites
- Maximum time to act upon a fault
- SLA claims in the agreement (i.e. what are the consequences if the provider does not fulfil the SLA)

Price

There are several price models and service descriptions. Some providers charge based on Query per second (QPS), some by the number of zones or delegations etc. Make sure the price model is scalable and transparant.

Glossary

Anycast is a network addressing and routing methodology in which a single destination IP address is shared by devices (generally servers) in multiple locations.

An Authoritative DNS server is a name server that holds all records for a given zone. Using that data it acts as the publication point for that zone, and can authoritatively give out DNS data that the zone owner wants to publish. Both primary and secondary DNS servers are considered authoritative.

AXFR (asynchronous zone transfer) refers to a data transfer of all records in a zone. The typical use case is a secondary server synchronising with its primary server.

CDS is a modern record type that allows a zone owner to report new crypto key information for the zone in question. This allows the registry operator to automatically pick up this information and modify the information in the registry accordingly.

CSYNC is a modern record type that allows a zone owner to report new delegation information for the zone in question. This allows the registry operator to automatically pick up this information and modify the information in the registry accordingly.

DNSSEC refers to DNS Security Extensions which use public key cryptography to authenticate that the data sent in response to a DNS query. This means that the “answers” sent from authoritative DNS servers in response to DNS queries can be validated as genuine.

DNS-based application security such as DANE, DKIM, DMARC and SPF leverage the authentication provided by DNSSEC to provide enhanced security measures for other protocols such as email.

DNS Transaction Signatures (TSIG) enable the DNS to authenticate lookups and updates to a DNS database. They are most commonly used to dynamically update the DNS or between primary and secondary DNS servers when synchronising zone data.

Hidden Primary Server – this server is not listed at the registry or as an NS record in the DNS zone. This way, the primary DNS server is more protected as it's not publicly visible.

IXFR (incremental zone transfer) refers to a partial data transfer of records in a zone. The typical use case is a secondary server synchronising with its primary server.

Network Time Protocol (NTP) is the most commonly used protocol for synchronising the time on computer systems.

Primary DNS server for a zone is the server that stores the definitive versions of all records in that zone. It is identified in the Start of Authority (SOA) resource record.

Resource Records (RRs) are the data elements that build up DNS zones. There are many types of RRs which carry different types of information.

RFC documents contain technical specifications and organisational notes for the Internet.

Secondary DNS server for a zone uses an automatic updating mechanism to maintain an identical copy of the primary server's database for a zone.

Start of Authority (SOA) is a type of resource record that carries important administrative and technical data for a zone.

Time-to-Live (TTL) is a field in every DNS record that tells DNS clients how long the information in that record is valid and, thus, how long it can be stored in the DNS client's cache.

Unicast DNS refers to the classic method of addressing and routing on the Internet, where an address is reachable at exactly one location. For an alternate method, see anycast.

Zone transfer is the process of copying the contents of the zone from its primary DNS server to a secondary DNS server.

About Internetstiftelsen

The Swedish Internet Foundation is an independent, private foundation that works for the positive development of the internet. We are responsible for the Swedish top-level domain .se and the operation of the top-level domain .nu, and our vision is that everyone in Sweden wants to, dares to and is able to use the internet.

Contact: info@internetstiftelsen.se

About Netnod

Netnod provides critical infrastructure support ranging from interconnection services and Internet Exchanges to DNS services, root server operations and activities for the good of the Internet. As innovators at the core of the Internet with a worldwide reputation for our services and the expertise of our staff, we ensure a stable and secure Internet for the Nordics and beyond.

Netnod's range of activities include:

- running interconnection services and the largest Internet Exchange in the Nordics
- providing secondary DNS services to partners, enterprises and some of the largest TLDs in the world
- operating I-root, one of the world's 13 root name servers
- providing Time and Frequency (NTP, NTS and PTP) services for Sweden

Established in 1996 as a neutral and independent Internet infrastructure organisation, Netnod is based in Sweden and fully owned by the non-profit foundation TU-stiftelsen (Stiftelsen för Telematikens utveckling).

Contact: info@netnod.se