A High-Availability Architecture for the Dynamic Domain Name System

Geoffrey G. Filippi

Thesis submitted to the faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the
degree of

Master of Science
in
Computer Engineering

Dr. Scott F. Midkiff, Chair
Dr. Luiz A. DaSilva
Dr. Mohamed Eltoweissy

May 2, 2008
Arlington, Virginia

Keywords: DNS, DDNS, BGP, anycast, DHCP, replication, LDAP, multi-master, high-availability, reliability

A High-Availability Architecture for the Dynamic Domain Name System

Geoffrey G. Filippi

Dr. Scott F. Midkiff, Chair

Computer Engineering

Abstract

The Domain Name System (DNS) provides a mapping between host names and Internet Protocol (IP) addresses. Hosts that are configured using the Dynamic Host Configuration Protocol (DHCP) can have their assigned IP addresses updated in a Dynamic DNS (DDNS). DNS and DDNS are critical components of the Internet. Most applications use host names rather than IP addresses, allowing the underlying operating system (OS) to translate these host names to IP addresses on behalf of the application. When the DDNS service is unavailable, applications that use DNS cannot contact the hosts served by that DDNS server. Unfortunately, the current DDNS implementation cannot continue to operate under failure of a master DNS server. Although a slave DNS server can continue to translate names to addresses, new IP addresses or changes to existing IP addresses cannot be added. Therefore, those new hosts cannot be reached by the DDNS.

A new architecture is presented that eliminates this single point of failure. In this design, instead of storing resource records in a flat text file, all name servers connect to a Lightweight Directory Access Protocol (LDAP) directory to store and retrieve resource records. These directory servers replicate all resource records across each other using a multi-master replication mechanism. The DHCP servers can add records to any of the functioning DNS servers in event of an outage.

In this scheme, all DNS servers use the anycast Border Gateway Protocol (BGP). This allows any of the DNS servers to answer queries sent to a single IP address. The DNS clients always use the same IP address to send queries. The routing system removes routes to non-functional name

servers and delivers the request to the closest (according to network metrics) available DNS server.

This thesis also describes a concrete implementation of this system that was created to demonstrate the viability of this solution. A reference implementation was built in a laboratory to represent an Internet Service Provider (ISP) with three identical regions. This implementation was built using Quagga as the BGP routing software running on a set of core routers and on each of the DNS servers. The Berkeley Internet Name Daemon (BIND) was used as an implementation of the DNS. The BIND Simplified Database Backend (SDB) interface was used to allow the DNS server to store and retrieve resource records in an LDAP directory. The Fedora Directory Server was used as a multi-master LDAP directory. DHCP service was provided by the Internet Systems Consortium's (ISC) DHCP server.

The objectives for the design were high-availability, scalability and consistency. These properties were analyzed using the metrics of downtime during failover, replication overhead, and latency of replication. The downtime during failover was less than one second. The precision of this metric was limited by the synchronization provided by the Network Time Protocol (NTP) implementation used in the laboratory. The network traffic overhead for a three-way replication was shown to be only 3.5 times non-replicated network traffic. The latency of replication was also shown to be less than one second. The results show the viability of this approach and indicate that this solution should be usable over a wide area network, serving a large number of clients.

Contents

Figures

Tables

Chapter 1. Introduction

1.1. Thesis Statement

The Domain Name System (DNS) is a critical architectural component of network systems. DNS must be functional at all times for systems that depend on it to be available. Static DNS is traditionally important for giving names to servers in the client-server model of computing. When an email or web server changes its IP address, clients can continue to reach the server using the server's domain name. The Dynamic DNS (DDNS) is just as important for systems that rely on dynamically-assigned Internet Protocol (IP) addresses. In applications like Voice over Internet Protocol and file sharing, the peer-to-peer model is becoming more important. In the peer-to-peer model, hosts are both clients and servers. In nomadic and mobile peer-to-peer applications in particular, it can be helpful to assign a dynamic DNS name when one peer needs to connect to another. Some applications require the flexibility to allow peer-to-peer hosts to change IP addresses frequently. This can be simplified if hosts use dynamic DNS names to contact each other. To ensure that voice applications are available in critical situations, dynamic DNS must also be available. In applications that cannot tolerate the loss of DNS service, it is not acceptable for DDNS to have a single point of failure. Unfortunately, there is a notable single point of failure in the DDNS system as implemented by the Berkeley Internet Name Daemon (BIND). If the master DNS server is down, the updates that would normally be sent to it are lost forever. When the Dynamic Host Control Protocol (DHCP) assigns new addresses to a device during an outage, that device will not be reachable by their host name. Even when the master server is restored to operation, these devices will still not be reachable using their host name, because DNS was never updated. In this sense, a single critical failure on one server can cause a service outage for many devices. This problem is described in detail in Chapter 3. As our dependence on network services increases, we become less and less tolerant of an outage of those services. The purpose of this research is to eliminate this single point of failure to achieve the level of reliability necessary for this critical network.

1.2. Research Objectives

The reliability problems observed with current DDNS systems can be resolved with an appropriate architecture. In general, there are many ways to improve system reliability. For the purpose of this thesis, the following design goals were pursued:

- Provide redundant servers that can accept DNS updates (masters)
- Allow a master can receive updates and propagate them to the others
- Ensure that properly functioning masters automatically queue updates until a failed master returns to service
- Enable automatic reinitialization of a replacement master by existing masters
- Allow rapid failover and recovery

The following tasks were done to demonstrate feasibility and evaluate performance:

- Implement a laboratory prototype
- Conduct performance experiments and analyze results
- Demonstrate confidence in performance and identify potential performance issues

1.3. Research Findings

A design is presented that satisfies the research objectives. This design was realized as a reference implementation. The performance of this implementation is examined. The performance, as measured, shows that the implementation functions as designed. Research objectives are addressed with a specific set of design objectives for the system. To evaluate the performance, metrics related to the design objectives are identified.

1.4. Thesis Organization

This thesis contains seven chapters.

Chapter 2 covers the underlying technologies that are the building blocks of this architecture. The focus is on DNS, DDNS and the use of the Border Gateway Protocol (BGP) anycast in the root DNS system.

Chapter 3 begins by explaining the importance of high-availability DNS and DDNS systems. Next, recent problems with DNS availability in general are discussed. Finally, this chapter outlines the single point of failure within current DDNS architectures.

Chapter 4 describes the proposed architecture for a highly available DDNS system. The components of this solution are described in detail. The coverage includes anycast BGP, the Lightweight Directory Access Protocol (LDAP) and multi-master replication, and the BIND Simplified Database Backend (SDB) interface.

Chapter 5 discusses the specific implementation of this architecture that serves as the reference in this research.

Chapter 6 analyzes the performance and results of the reference implementation. Performance, reliability, replication overhead, latency, scalability, and fault recovery issues are presented.

Chapter 7 provides an interpretation of the results. The discussion includes opportunities for future work.

Chapter 2. Background

To appreciate the research presented herein, a working knowledge of certain fundamental Internet technologies is required. This chapter discusses key concepts and highlights those areas that are material to the problem presented in the next chapter. The chapter first explains the DNS, the system that maps domain names to IP addresses. It then discusses DHCP, the protocol used to automatically configure network settings on Internet hosts. Next, the chapter introduces DDNS, the extension of DNS that supports dynamic updates. The chapter then presents anycast and related addressing schemes. Anycast is an essential part of the design presented in chapter four. Finally, this chapter discusses prior work, including related implementations.

2.1. Domain Name System (DNS)

The DNS is a worldwide distributed database that maps domain names to the IP addresses of each host on the internet. This can be compared to the way a phone book maps telephone numbers to people. As an illustration of this concept, suppose we have two individuals, Al and John. Suppose Al wants to call John. Assuming Al has not memorized John's phone number, he refers to a phone book. Figure 2.1 shows this example.

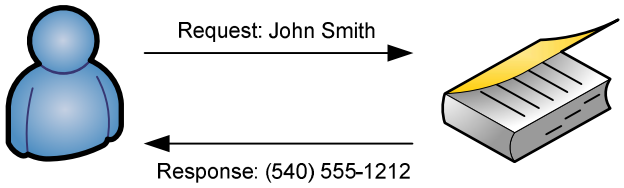


Figure 2.1. How a phone book works.

Now suppose we have two internet hosts. The client wants to contact the server "www.example.com" but does not know the IP address. The client refers to a DNS server to find the IP address that corresponds to the domain name. This scenario is shown in Figure 2.2. Since DNS is based on an Internet standard, most of the discussion that follows is based on the respective Request For Comments (RFC) documents. The intent here is to summarize and highlight key aspects of the standards that are the most relevant to the problem at hand.

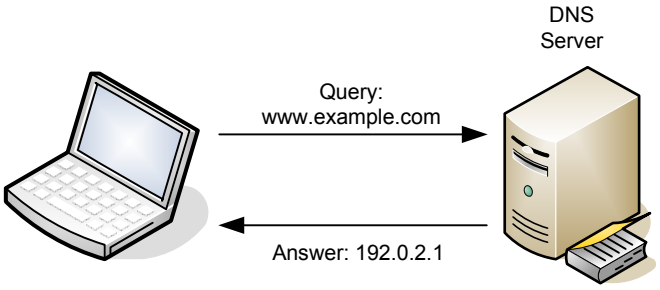


Figure 2.2. How DNS works.

2.1.1. Brief History

Before the Internet came into wide use, it was possible to keep the host name to IP address mapping of every Internet host in a single file stored on each node. This was known as the "hosts.txt" file or simply the "hosts" file. This file still exists on computers today, but typically only stores a small number of locally interesting hosts. Early in the use of the Internet, keeping this file current with a growing number of hosts became intractable. DNS was designed as a replacement for the practice of exchanging and merging the records in these files [1].

2.1.2. Components of DNS

Conceptually, there are three elements of the DNS: the domain name space and resource records, name servers, and resolvers. The domain name space and resource records are organized in a hierarchical structure. Name servers are responsible, or authoritative, for sub-trees of this hierarchy. This facilitates the distributed nature of the DNS, because administration of each subset of the namespace can be delegated to different entities or organizations. Resolvers are the components of the DNS that reside on each Internet host. The resolver is a program that is responsible for requesting host name-to-IP address lookups from DNS servers, and interpreting the result [2].

To understand the domain name space, consider a sample domain name. For example, "www.example.com" represents the name for an Internet host. Each string separated by a dot is called a label. The rightmost label "com" is known as a top level domain, or TLD. There are a small number of TLDs (e.g. "com," "edu," and "gov"), and this number is rarely changed. These are the nodes closest to the root of the tree comprising the domain name space. The next label to the left, "example" is a domain delegated to another entity. This label typically corresponds to a business (e.g. "google"), a government entity (e.g. "whitehouse"), or an academic institution (e.g.

"vt"). Finally, the label "www" in this example names a specific host on the entity's network. The administrator of "example.com" is free to delegate a subset of the domain to another entity. For a business or academic institution, this might be a specific department. For example, a web server responsible for sales might be located at "www.sales.example.com." Each domain name is known as a resource record. The resource record associated with the above examples is known as an "A" record, which corresponds to an IPv4 address. There are several other types of resource records defined (AAAA, MX, NAPTR, etc.). For the sake of simplicity, we can safely ignore these other types. Figure 2.3 shows a partial map of the DNS name space.

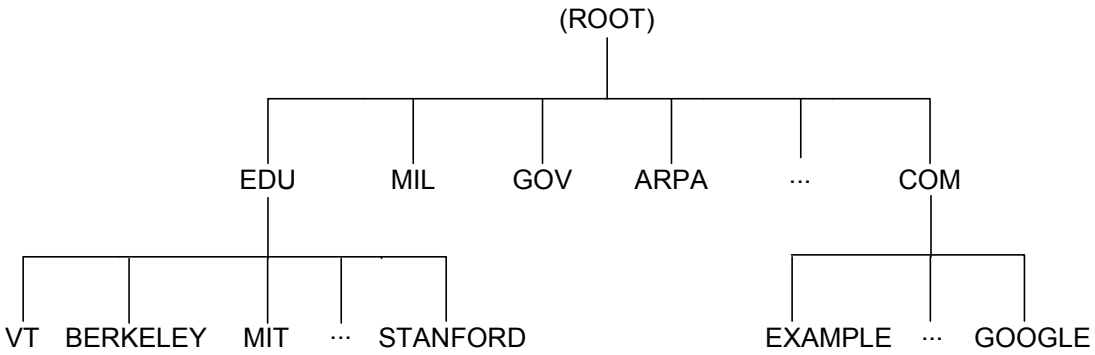


Figure 2.3. DNS name space (partial).

Name servers are the machines and programs that provide the translation between domain names and IP addresses. Name servers are configured with zone files. In our earlier example, the domain "example.com" is called a zone. A zone file contains all of the resource records, or domain names for that domain or zone. There are three main types of name servers. Primary servers are the ultimate source of the zone information. Slave servers contain a copy of the information stored on the primary servers. The slave servers are available to provide the domain to IP translation in the event that the primary server is unavailable. When the primary server gets a new record, it automatically updates the slave or secondary server as shown in Figure 2.4. The third type of DNS server is a cache. Cache servers retrieve resource records from other DNS servers and record or cache the result locally to provide a faster translation when the same record is requested in subsequent queries.

Resolvers take requests for domain name-to-IP address translations from client programs and query DNS servers by passing them the name to be translated. The resolver receives the response

from the server and provides the IP address to the client program. The resolver may contact more than one server, as required to find the server that has the desired resource record.

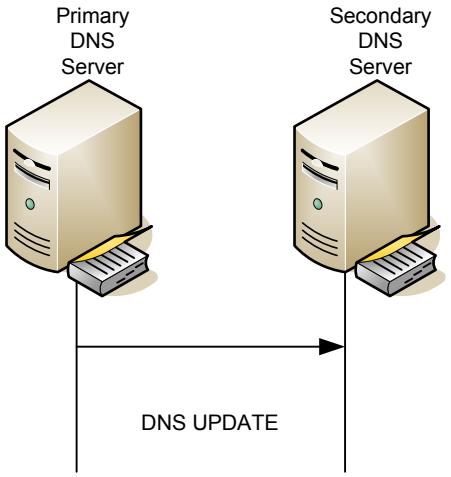


Figure 2.4. DNS update.

2.1.3. Protocol

The DNS protocol defines a single message format. DNS queries, responses, and other communications are all conveyed in these messages. For instance, in a query, the answer section is blank. In the response, that answer section is populated. DNS messages can be carried over Transmission Control Protocol (TCP) or User Datagram Protocol (UDP) on port 53. Most DNS messages are delivered in a single UDP datagram. The main use of TCP for DNS is in zone transfers.

2.1.4. Implementations

BIND is the most widely deployed name server on the Internet [1]. It was originally written at the University of California, Berkeley. BIND works by keeping resource records in text zone files. One feature that BIND supports is the SDB LDAP. This allows BIND to use LDAP to connect to a directory server to retrieve resource records when responding to queries. Using LDAP as a backend for BIND enables the zone data to benefit from the advanced replication techniques described in Sections 4.3. and 4.4.

2.2. Dynamic Host Control Protocol (DHCP)

DHCP provides IP addresses to hosts. This allows hosts to join the network without involving a network administrator. This is critical for the scaling of large local area networks (LANs). The network administrator defines IP address assignment policies and the DHCP server allocates addresses based on those policies.

2.2.1. Purpose

The primary purpose of DHCP is to provide automatic administration of IP addresses. This enables nomadic networking, where internet hosts can easily move between different LANs. Another benefit of DHCP is that it simplifies the reuse of IP addresses. This is especially important as the number of hosts in the global Internet exceeds the number of useable addresses available in the IP address space. DHCP simplifies this address reuse by using a pool of available IP addresses. These addresses are assigned based on the policy configured by the network administrator.

In addition to providing IP addresses, DHCP also provides other network configuration information. Most DHCP servers, for example, also provide the subnet mask and the default gateway. Another common DHCP option is to provide IP addresses of one or more DNS servers. This allows a DHCP client to join the LAN and begin resolving the IP addresses of Internet hosts using their DNS names. There are many other possible parameters that can be provided by DHCP, but they are outside the scope of this research.

2.2.2. Modes of Operation

DHCP supports three modes of operation: dynamic, automatic, and manual [3]. The most common mode of operation is dynamic. In the dynamic mode, IP addresses are temporarily assigned as leases, giving the client the right to use the address for a finite period of time. Before the lease expires, the client can request that the lease be renewed. This extends the lease for another period of time. In the automatic mode, the DHCP server permanently assigns an IP address to the client. In manual mode the IP address is assigned by the network administrator, although the DHCP server still delivers the assigned address to the client.

2.2.3. Protocol

DHCP servers listen on port 67 over UDP, while DHCP clients listen on port 68 over UDP. The DHCP uses a single message format. DHCP is historically related to the Bootstrap Protocol (BOOTP). The messages relevant to this research are described below. Figure 2.5 shows the typical life cycle of a dynamically assigned IP address from the perspective of the DHCP client and server. The interaction with DDNS is not shown.

Client: DHCPDISCOVER - Are any servers out there?

Server: DHCPOFFER – Server here. Would you like 192.0.2.1?

Client: DHCPREQUEST - OK. I'll take 192.0.2.1

Server: DHCPACK - OK. 192.0.2.1 is yours.

Client: DHCPRELEASE - I'm done with 192.0.2.1 now.

The DHCPDISCOVER message is broadcast by the client to the local network in search of an available DHCP server. The DHCP server responds with a DHCPOFFER message. The server also creates a lease and reserves the IP that was requested, if available, or another IP address if it is not. The contents of the offer message are the MAC address of the client, IP address offered, subnet mask, lease duration, and IP address of the server. The client responds with a DHCPREQUEST message that contains the acceptance of the offer. Other DHCP servers on the network, if they exist, will see this message and withdraw any offers that they sent. These other DHCP servers will also return other offers to the pool. Clients are only permitted to request one offer. The server then provides a DHCPACK to acknowledge the receipt of the request and that it has been confirmed. The acknowledgement also contains the lease duration and other optional information. This message marks the end of the configuration phase. An optional message, the DHCPREALEASE can be sent by the client to let the server know that it is done with the address. Usually, this request is not sent, because the DHCP client has no way of predicting when the user will disconnect from the network. This is not a problem, because the server will simply recover the IP address when the lease expires.

The DHCP server can also update a DDNS server if both are configured for this function. Once the DHCP server has acknowledged the request, the server can send an update to the DDNS server to let it know the host name and IP address of the new client. This interaction is covered in detail in the next section.

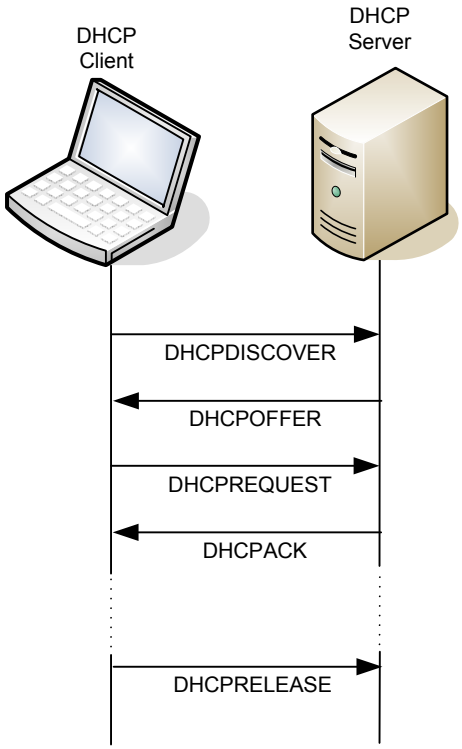


Figure 2.5. DHCP lease life cycle.

2.3. Dynamic DNS (DDNS)

DDNS is an extension to DNS that allows domain names to be inserted or removed from the DNS database dynamically [4]. When a DHCP server assigns a new address, it can also be configured to notify the DNS server that a new host has joined the network.

Figure 2.6 shows a typical DDNS update, including a Master (Primary) and Slave (Secondary) server. Only the DHCP update is shown here. First, the DHCP server sends an update to the Primary DNS server. Then, the Master DNS server sends the update to the Slave server.

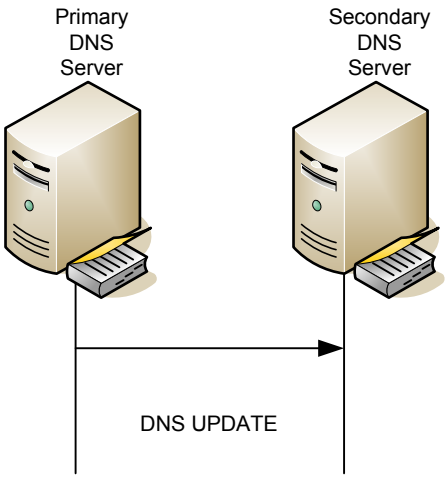


Figure 2.6. DDNS update.

Figure 2.7 shows more detail of the DDNS update, including a Master (Primary) and Slave (Secondary) server. Here the DHCP transaction is also shown. Then the Master DNS server sends the update to the Slave server.

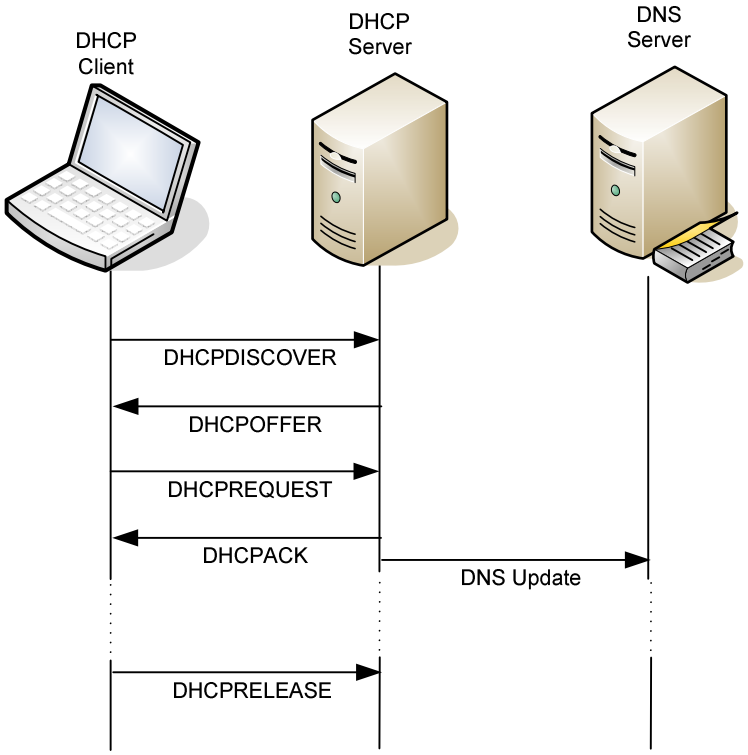


Figure 2.7. DDNS update from DHCP server.

With additional automation and complexity, there are some critical failure scenarios that can occur. Redundancy configurations and the problems with those systems are covered in detail in Chapter 3.

2.4. Anycast

The design presented in Chapter 4. utilizes anycast. Anycast is an IP address and routing scheme that is used when a group of servers all provide the same service and the clients using that service are indifferent as to which of those servers provides that service [5]. To further explain anycast, the comparison to other routing and addressing schemes is explored.

2.4.1. Routing Schemes

To understand anycast, it helps to discuss more traditional routing schemes [6]. The predominant routing scheme used on the Internet is unicast. This is when one Internet host creates a packet destined for another Internet host. This can also be described as a one-to-one communication. An example of this is a web client requesting a page from a web server. This routing scheme is shown in Figure 2.8.

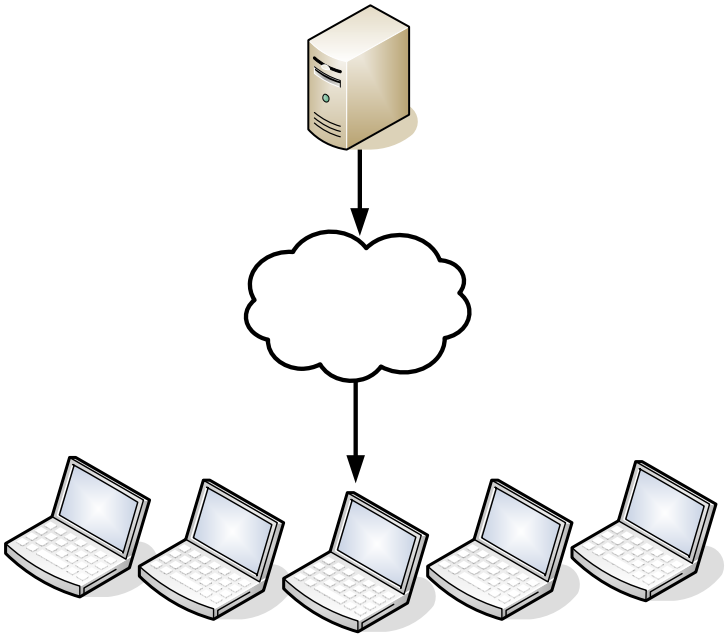


Figure 2.8. Unicast routing scheme.

Another scheme for routing is broadcast. This is when one Internet host sends a packet to "all" hosts. This is commonly used in a LAN when one host needs to find out which host has a particular IP address. Since the originator of the packet cannot know where the intended recipient is, it simply sends the packet to a special network address, known as the broadcast

address. All hosts listen to the broadcast address. This one-to-many relationship is illustrated in Figure 2.9.

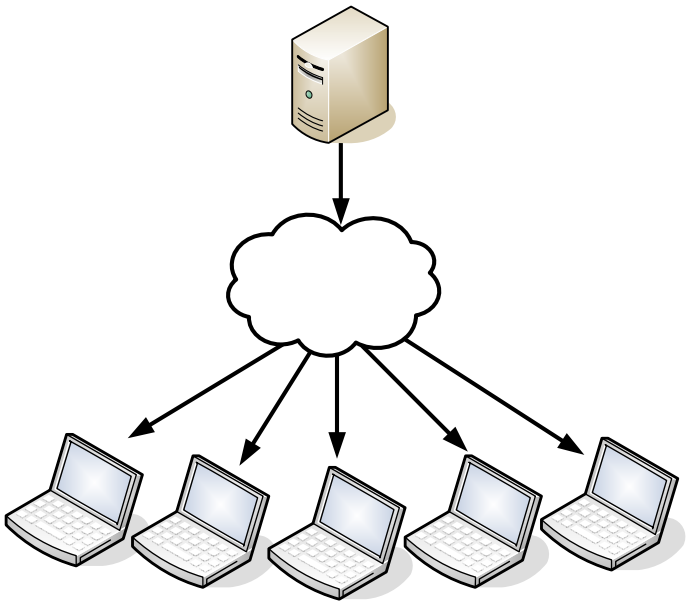


Figure 2.9. Broadcast routing scheme.

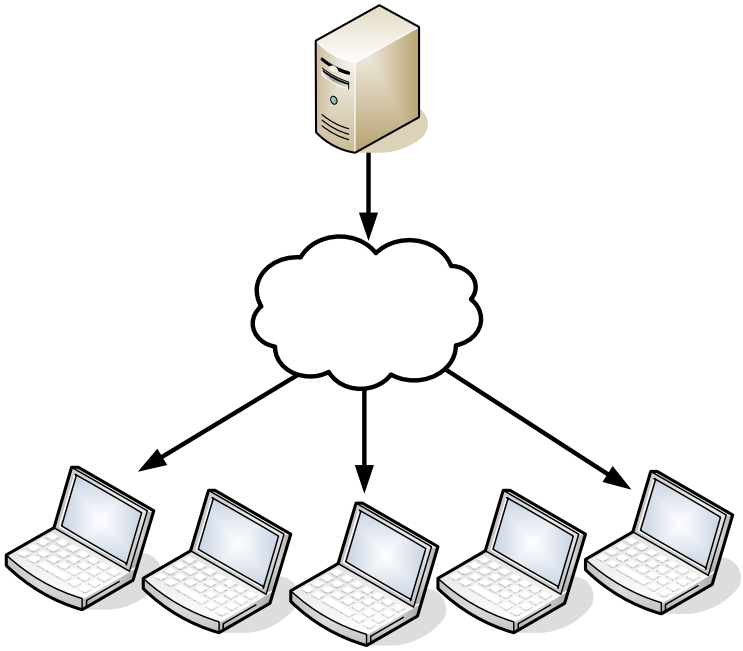


Figure 2.10. Multicast routing scheme.

Another one-to-many routing scheme is multicast. This is a subscription-based routing scheme. Individual Internet hosts join a multicast group by using a multicast IP address. If they are connected to a router (and network) that supports multicast routing, the sender can send a single packet. The multicast network will provide copies of that packet to all hosts that are configured

to use that multicast IP address. An example of this would be an internet radio broadcast or webcast. Figure 2.10 shows the multicast routing scheme.

Anycast is a one-to-any routing scheme. In anycast, several Internet hosts share the same IP address. The routing system computes the route to the closest (according to network routing metrics) host with that IP address and delivers the packet only to that host. Anycast routing is shown in Figure 2.11.

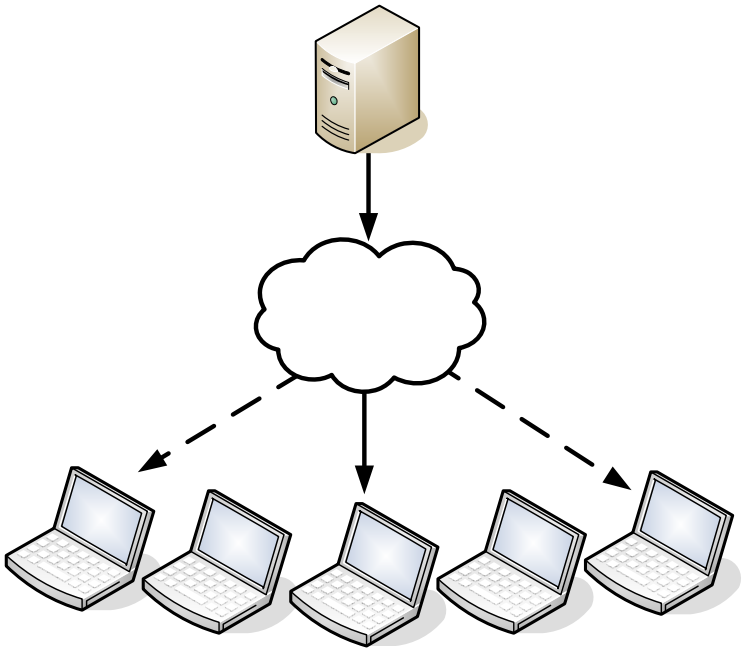


Figure 2.11. Anycast routing scheme.

2.4.2. Border Gateway Protocol (BGP) and Anycast

BGP is the primary inter-AS (Autonomous System) routing protocol used on the Internet today. Its widespread use, and the success of the Internet itself, can be attributed to BGP's efficient aggregation of route advertisements. BGP is a policy-based routing protocol. BGP uses a distance vector routing algorithm.

In BGP, public AS numbers are assigned by the Internet Assigned Numbers Authority (IANA) to networks that are maintained by independent organizations [7]. A BGP router uses this AS number as an identifier. Two BGP routers connected to each other are known as neighbors. Changes to the state of network availability are announced to their neighbors via update

messages. One of the most important features for the scope of this problem is support for anycast. BGP supports many advanced features that are not relevant to the research here.

OSPF is a popular routing protocol for intra-AS routing. OSPF is based on a link-state algorithm. OSPF networks are administered as areas. OSPF also supports anycast. OSPF was not chosen for this research. The use of BGP anycast for DNS is recommended by the Internet Engineering Task Force (IETF) [8], but no reason for this decision is provided. However, since BGP is the primary inter-AS routing protocol used on the Internet, using BGP anycast facilitates the world-wide deployment of BGP anycast for DNS.

Figure 2.12 shows the normal behavior of an example BGP anycast network. In this case, the client attached to the first router, rtr-1, sends out packets intended for the anycast IP address. The first router, rtr-1, delivers those packets to the "closest" name server, ns-1. If ns-1 fails for any reason, rtr-1 detects this and begins using the "next closest" name server, ns-2. This is the failover scenario shown in Figure 2.13.

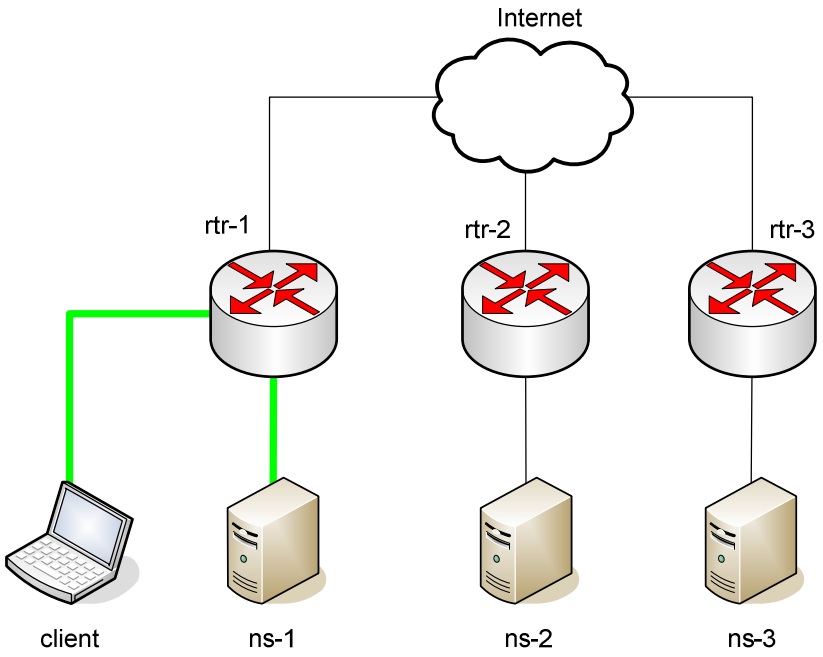


Figure 2.12. Anycast normal case.

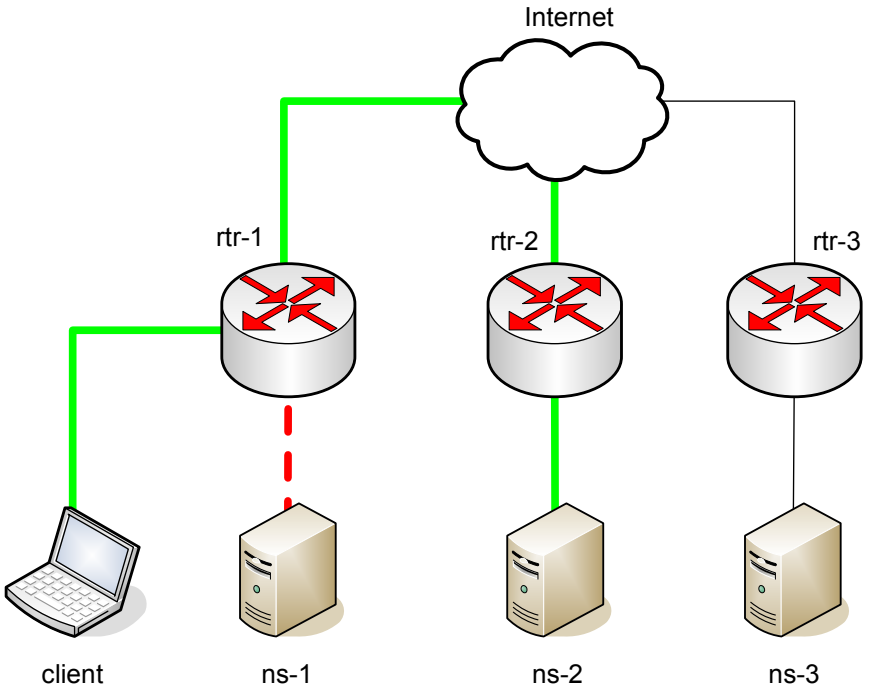


Figure 2.13. Anycast failover case.

2.4.3. Scope

The scope of the anycast address is the degree to which that address is propagated through the Internet [9]. The global anycast scope refers to anycast addresses that are available throughout the entire Internet. Local anycast scope addresses are limited to a subset of the Internet. The techniques presented herein are applicable to either scope. Local and global scoped addresses can be applied together to optimize these techniques presented to relieve local network traffic problems. However, these optimization techniques are beyond the scope of this research.

2.5. Prior Related Work

This research builds on the work of the Internet community to improve the reliability of static DNS using anycast [8]. Recent research on the application of static DNS using anycast is promising. Sarat showed that the root servers anycast implementation of DNS reduced the number of outages as compared to those root name servers that do not use anycast [10]. Sarat also showed that query latency was reduced by this approach. They were also able to conclude that the closest DNS server is frequently chosen. The tradeoff observed in this research is that outages can be extended by slow BGP convergence time.

Sarat's findings support the anycast architecture that is currently in use on the root name servers. However, Sarat does not consider Dynamic DNS. The Internet Software Consortium (ISC) has also written a paper on this subject [11]. This paper is mostly geared toward the implementation of an anycast DNS system. They do not consider Dynamic DNS as part of their design.

The design studied in Sarat not only neglects DDNS, but it cannot support DDNS without significant modification. Although the design provides a high-availability design for the name resolution interface, it does not provide an automated mechanism for synchronizing the potentially rapidly changing resource records necessary to support DDNS. DDNS is important for any host that is configured via DHCP, but is especially important for mobile and nomadic systems where the IP address can change frequently.

Although no previous work has suggesting using anycast with DDNS, other research has explored the benefits and caveats of using anycast for static DNS [12, 13]. They studied and measured existing large-scale deployments of anycast DNS. They also recommend guidelines for minimizing the impact of those caveats. There is also prior work that uses anycast DNS to measure stability of the underlying BGP routing system [14]. However the focus of that research is to characterize anycast BGP, not to analyze DNS performance.

The design presented herein not only builds upon the solution provided by anycast DNS, but it also provides a solution for rapidly changing resource records necessary to support DDNS. Furthermore, the solution presented herein provides a way to integrate these solutions into a comprehensive solution for high availability DDNS.

2.6. Summary

The key concepts of the Internet needed to understand the research presented are described in this chapter. DHCP automates the configuration of Internet hosts by assigning unused IP addresses to those hosts. DNS translates domain names to IP addresses, but those domain names and IP addresses are manually (statically) assigned. DDNS extends DNS by providing support for dynamic (automatically) assigned updates. These updates can come from the DHCP server, for example. As explained in this chapter, anycast is an Internet addressing and routing scheme that allows clients to use any one of a group of identical servers that all share the same IP

address. Finally, prior research and practice related to this problem and the limitations of that research were analyzed.

Chapter 3. Problem

Internet hosts are located using IP addresses. Domain names map to these IP addresses. Users and applications often prefer to use the domain names when referring to internet hosts. Most internet applications assume the function of DNS for their operation. For these applications, a DNS outage is equivalent to an outage of the application itself. For critical applications, this is not acceptable. By simply replicating DNS records to a slave server, some level of reliability can be achieved using current practices.

To make matters worse, there have been a number of DNS outages caused by Distributed Denial of Service (DDoS) attacks. Since DNS is such an important part of the internet, it is also an attractive target for hackers. Targeted attacks can be launched against specific domains for various reasons. General attacks can also be launched against the root name servers.

As a solution to this problem, this thesis proposes a design. The viability of this design is tested by developing an implementation. The performance of this implementation is analyzed to actually measure and evaluate the properties of the system.

3.1. DNS Updates

Mobile and nomadic hosts join and leave networks over time. When moving to a new network, these hosts are given new addresses via DHCP servers and the DHCP servers update DDNS when this happens. Updates in the current implementation of BIND are sent directly to the master DNS server. If this server is not available, the update is lost. Hosts that attempt to reach the mobile or nomadic host through its domain name will not be successful. The mobile host will not be reachable until such time that the master DNS server is returned to service and the host gets a new IP address from DHCP. Figure 3.1 shows this scenario.

A first approach suggests that the DHCP server should be modified to update the slave server. Also, the DNS implementation would need to be updated to support this. This modified scenario is shown in Figure 3.2.

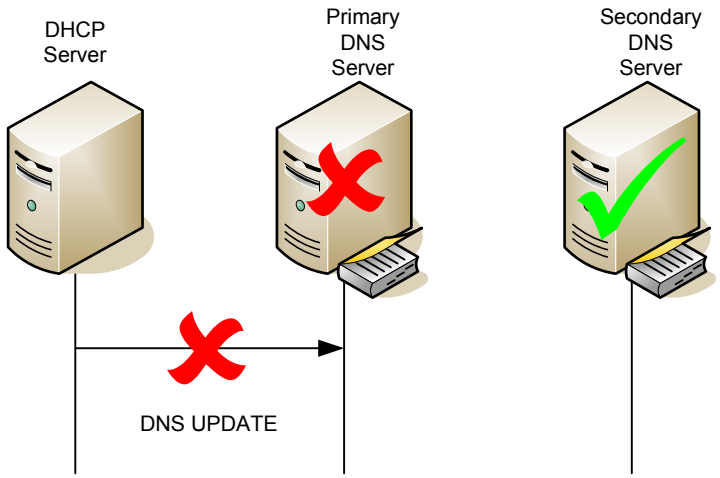


Figure 3.1. DDNS update failure.

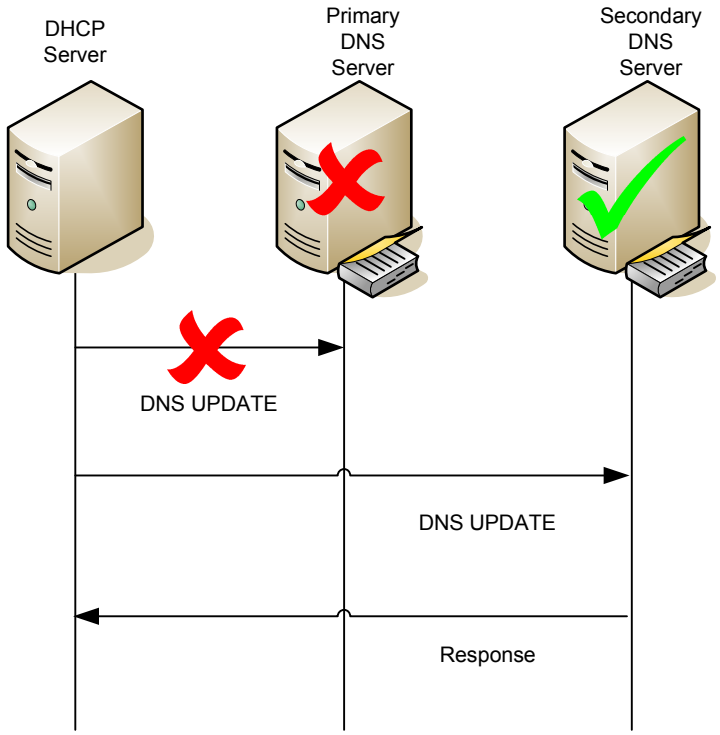


Figure 3.2. DDNS update to slave.

Unfortunately, there are some problems with this approach. First, current DHCP servers do not allow this. This is a simple problem to solve. The DHCP servers could simply be configured with a list of DNS servers that accept updates rather than a single host. A more difficult problem is that BIND and other DNS implementations do not allow slave servers to update masters.

Another way to solve this problem is to modify BIND to allow more than one master for a zone. This scenario is shown in Figure 3.3

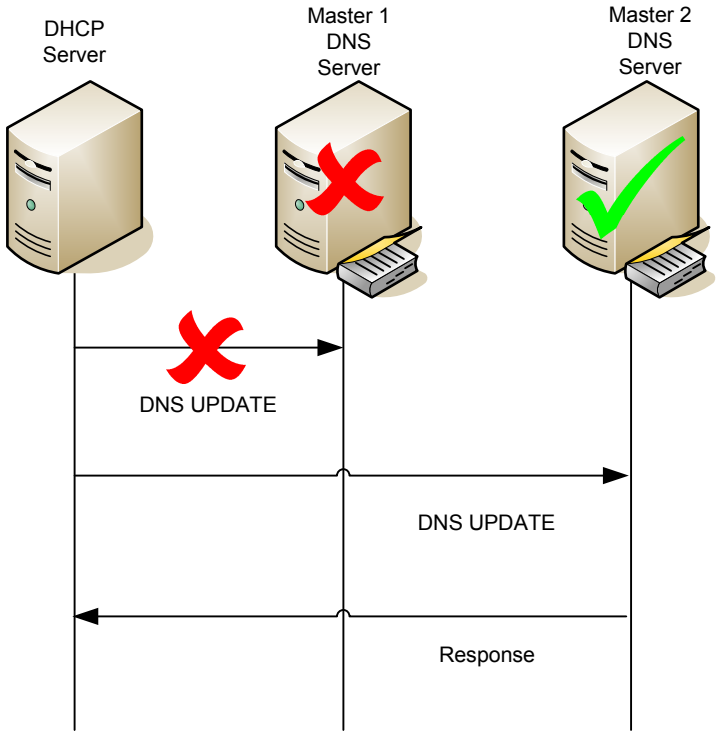


Figure 3.3. DDNS update to master 2.

These two problems are similar. The issue is that there is no mechanism in BIND to reconcile updates that happen on more than one server. The other issue is that clients who query the original master (Master 1) will not get answers when that server is down.

3.2. Research Approach

The approach taken in this research is to present a new architecture for DDNS that resolves the issues identified in this chapter. The anycast BGP and LDAP multi-master replication techniques described in Chapter 2 are key components of the design. The architecture that solves the identified problems is described in detail in Chapter 4. A concrete implementation of this design was built in a laboratory. This implementation is described in Chapter 5. The performance and functionality of the laboratory system was evaluated and measured to assure that it solved the problems that were identified with current DDNS systems. The results of this evaluation are presented in Chapter 6.

3.3. Summary

This chapter presents the key problem with the existing DNS implementation with regards to dynamic updates. Since slave servers cannot deliver DNS updates to master DNS server, the failure of a master DNS server is a single point of failure in the DDNS. Because of the importance of DNS on the Internet, this problem needs to be solved. Since DNS is so fundamental to the operation of the Internet, it is also a target for DDoS attacks. Any solution must be able to cope with these attacks to some degree. A solution for these problems is presented in the following chapter.

Chapter 4. Design

Problems of the current DDNS implementations were explained in Chapter 3. In this chapter, a design that addresses those limitations is presented. In particular, this design eliminates single points of failure from the system. Therefore, any single component of the solution can fail and all DNS clients will still be able to resolve domain names. The components of a generic DDNS system are outlined. The aspects of this particular DDNS solution are also presented. The selection of anycast and BGP provides a network path to a working DDNS server, even under failure of a DDNS server. This design uses a directory server as a storage facility for DNS resource records. The benefits of multi-master replication in the directory server design are discussed. Finally, the name server, BIND, is discussed. The BIND-SDB interface is described. This allows BIND to use LDAP directories to store DNS resource records. Finally, this chapter explains how the BIND-SDB interface is utilized in this design.

4.1. System Design

There are three major aspects of any DDNS system design (shown in Figure 4.1): DNS update processing, resource record storage, and DNS name resolution. The problems of update processing and name resolution are separate from each other. The DNS update is fundamentally a write operation, where the resource record is written to the data store. The DNS query is a read operation, where the resource record is retrieved from the data store. The constraint is that both operations must use the same data store. The design challenge is to provide a system that is always available for both operations even when single components of the system fail. Chapter 3 describes in detail why the current dynamic DNS implementations do not support write operation in the case where there is a failure of a single component, in particular the master server for that zone.

The DNS client interface solution uses the anycast BGP solution already in use in the Internet for static DNS name resolution. Anycast allows a single IP address to map to multiple identical servers. The routing system calculates the lowest cost path to a working anycast server. The DNS update interface solution is implemented as an agent that runs on the DHCP server. Using a

particular feature of the chosen directory server ensures that a single point of failure cannot disable the overall system. Anycast BGP is used to provide clients of a service with a single IP address that can be used to reach a group of servers that provide an identical service.

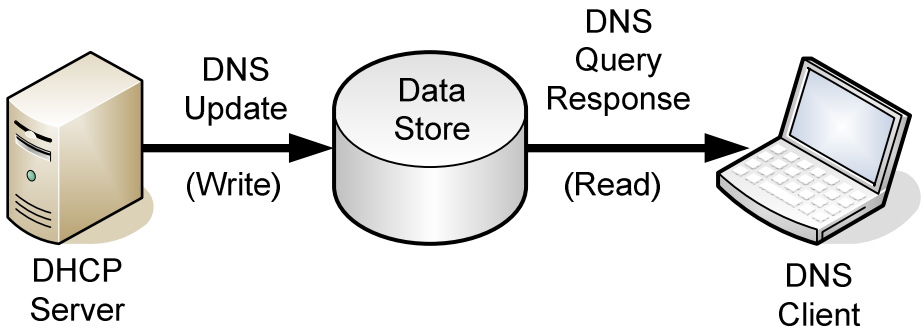


Figure 4.1. Aspects of DDNS system.

4.2. Anycast BGP

As explained in Chapter 2, anycast is best suited for protocols that have short-lived exchanges. Since DNS queries and responses are transmitted as single UDP packets, there is little opportunity for interruption of that exchange due to changes in the routing topology. Anycast is useful in this solution when the primary DNS server fails. For the best application performance in an outage scenario, the DNS client should not have to iterate through a list of potentially available DNS servers (incurring time-out delays at each server) until finally finding a working DNS server. Using anycast, if any of the pool of DNS servers is available, the DNS client's request will be routed to that DNS server on the first attempt. This approach is widely used on the Internet today for static DNS zones and has shown promising results [10]. This also simplifies the configuration of DNS clients, because redundancy can be accomplished with a single DNS server IP address. Using anycast increases the complexity of the solution to some degree. The BGP advertisement to the anycast address must be monitored such that the advertisement must only be made when the service is actually functioning. Also, the server that provides an anycast service must run a routing process that can produce these advertisements.

4.3. LDAP Multi-master Replication

Although anycast solves the problem of redundancy from the client's perspective, the redundancy of the server data itself cannot be addressed by anycast. To achieve redundancy of the zone data, every (not any) DNS server must have a complete image of the zones data at all times. There are many different techniques that can distribute a data store across multiple servers. In particular, clustered databases, synchronized files, and distributed file systems were considered. One reason for selecting LDAP is that the hierarchical DNS name space maps naturally to the directory server's hierarchical data base structure. Also, the LDAP replication mechanism is loosely coupled and suitable for distribution in the presence of significant network latency. This makes it a good solution for redundancy configurations across large geographic areas.

Figure 4.2 shows how multi-master replication works in this application. DHCP server 1 sends a DNS update via an LDAP add to its preferred directory server, Master 1. This update is propagated over the multi-master replication scheme to Master 2. Likewise, when DHCP server 2 sends a DNS update to Master 2, that update is also propagated to Master 1. Each directory server uses a unique unicast management IP address that is different than the anycast IP address used by the DNS process. This is because anycast is not suitable for the longer-lived replication transactions used by the LDAP protocol.

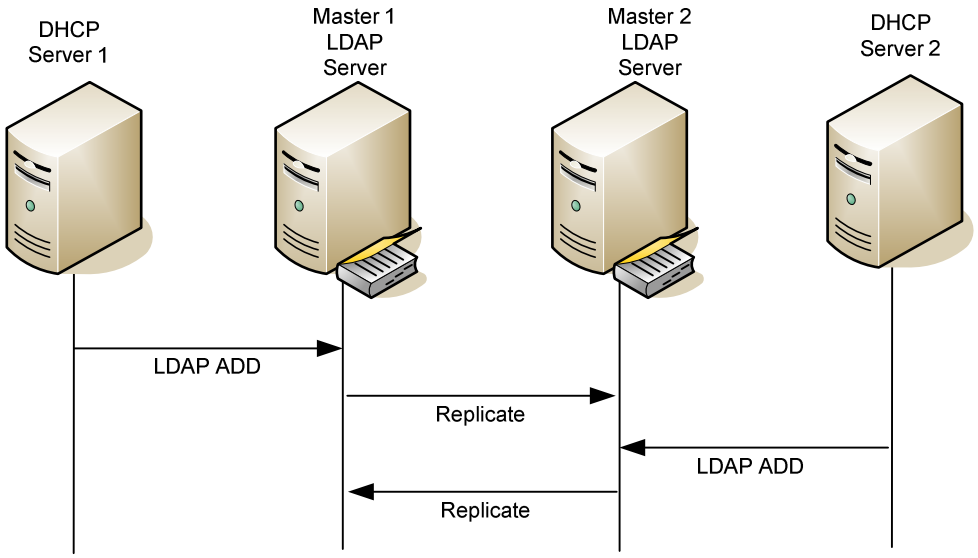


Figure 4.2. LDAP multi-master replication.

When a directory server goes down, the DHCP server update agent can iterate through a list of known LDAP servers until it finds one that is working. It can then add the record to that functioning directory server. This solves the problem of the single point of failure in the existing DDNS system. Furthermore, after a failed directory server returns to service, the multi-master replication mechanism automatically synchronizes the resource record updates that were missed by that directory server during the outage. This provides a simple mechanism of automatic recovery of that directory server.

4.4. BIND Simplified Database Backend (SDB) LDAP

In addition to the advanced replication features of the directory server discussed in Section 4.3. , a directory is also well-adapted to this problem because BIND supports SDB LDAP for storage of DNS zone information. Traditionally, BIND uses flat text files to store DNS zones and resource records. This works fine for static DNS zones, but has some important limitations with dynamic zones. When dynamic zones are used with zone files, a special file, called a journal file, tracks changes to any resource records. This is done because changes to the main zone file are not read until the DNS server application is reloaded. For this reason, simply copying zone files to multiple servers is not sufficient to provide a complete backup of that zone. Also, simply trying to merge journal file changes across two running DNS servers using an "out-of-the box" file system replication technique or file synchronization protocol would not produce the desired effect. The simple reason for this is that BIND does not support modification of the journal file outside of BIND itself.

BIND also supports the SDB for relational databases like MySQL. This approach was not chosen because MySQL, in particular, and other data base cluster implementations warn against distributing the database as a cluster over wide area networks [15, 16]. This precludes using a solution based on these databases as a geographically-dispersed redundancy solution. Fortunately, BIND SDB also supports an LDAP backend. The BIND SDB interface for LDAP provides a mechanism for the BIND process to look up the resource records relevant to the DNS query in a directory.

4.5. Summary

The design presented in this chapter addresses limitations in the current implementation of DDNS. This design allows for the failure of any single component without loss of access to the DDNS service. The discussion breaks down a generic DDNS solution into its basic components. The design is constructed using standard technologies that are already in wide use on the Internet. Anycast BGP provides clients with a high-availability interface to the DNS server for name resolution. The design also uses the BIND-SDB interface to connect BIND to an LDAP directory to store and retrieve resource records.

Chapter 5. Implementation

The design of a high-availability DDNS solution presented in Chapter 4. leaves many degrees of freedom. For hosts on different LANs to communicate, a router is required. DHCP server software provides clients with IP addresses and updates DNS servers with those addresses and an associated host name. Software that is capable of responding to DNS queries is also necessary. A directory server is used as a storage mechanism for DNS resource records. The specific software selected and their properties relevant to this research are highlighted.

5.1. Reference Implementation

This laboratory configuration is intended to represent an Internet Service Provider's (ISP) network. Refer to Appendix B. Laboratory Network Diagram for a diagram of the reference implementation laboratory. Three autonomous regions are defined and implemented. These regions are named region one, region two, and region three. Each region is a replica of the other two regions. These regions all contain a single core router labeled rtr-1, rtr-2, and rtr-3. These routers run the routing suite Quagga. Each region also contains a single server running BIND, the Fedora Directory Server, and Quagga. These servers are referred to as the name servers ns-1, ns-2, and ns-3. Finally, each region contains a DHCP server named dhcp-1, dhcp-2, and dhcp-3, respectively. The DHCP servers run the ISC implementation of the DHCP server. The IP addresses, hostnames, networks, interconnections, and layer 2 switches are also shown in the diagram, for reference. All servers, except the name servers, run FreeBSD 6.2. The name servers run Fedora Core 6 (Linux). As of this writing, a binary version of Fedora Directory server is not available for FreeBSD.

5.2. Quagga

Quagga is used as the routing software for this implementation. Quagga is a fork of the Zebra routing suite, which is no longer actively developed. The BGP routing process is used to implement anycast. This implementation uses a standard Quagga implementation. Three core BGP routers are configured in the laboratory with unique AS numbers. Three additional BGP

routing instances, also with unique AS numbers, are configured on each of the name servers. Each name server announces the same anycast IP address to their respective core router. The core routers propagate these announcements to each other. Each core router evaluates the weight of the route to the address. Each core router prefers the directly attached, or local, name server in the normally functioning case. In the fault case, the route converges on the new best route, to a name server.

There is a BGP routing process running on the name server that advertises the anycast address to the next hop router. A monitoring agent was implemented to periodically check the ability of the name server to resolve a known host name. If this check fails, the name server monitoring agent withdraws the BGP advertisement to the next-hop router. After the advertisement is withdrawn, the agent periodically checks to see if the DNS server returned to service. When this happens, the agent reinserts the route advertisement for the anycast address.

5.3. ISC DHCP

ISC's DHCP server is used in this implementation. The DHCP server uses a standard configuration. Instead of relying on the DHCP server to update the DNS server, a lease file monitoring agent periodically checks the DHCP leases file for changes. The agent directly connects to the LDAP backend that supports the DNS server zone data and performs an LDAP add to record the appropriate DNS record.

5.4. BIND

In this implementation, BIND is used as the DNS server. In the default configuration, BIND uses plain text files to store zone data. The BIND-SDB LDAP interface provides a mechanism to use an LDAP directory to store resource records. To use the BIND-SDB LDAP interface, BIND must be compiled with this option. This is not a standard build parameter, but several Linux and BSD distributions include a binary SDB version of BIND as well as the standard non-SDB version. In the named.conf file, the SDB-enabled zones use the "database" declaration rather than the typical "file" declaration. Although the syntax is a bit perplexing, the idea is straightforward. Rather than providing the file system location of a flat text file that stores resource records, an

LDAP Uniform Resource Locator (URL) [17] is provided that points the BIND-SDB interface to the appropriate directory location.

5.5. Fedora Directory Server

The Fedora Directory Server is used in this implementation to serve as the backend data store for DNS zone data. This data is replicated over three separate directory servers. This provides redundancy and improved read-only performance. Replication is provided by a multi-master replication scheme. As long as one directory is functional, the solution can continue to process DNS updates. This must be considered an advanced LDAP configuration, but it is well supported by the software and documentation. Unfortunately, the Fedora Directory Server schema does not support the BIND-SDB interface as distributed. However, the OpenLDAP server schema does. Using the OpenLDAP server schema as a reference, a Fedora Directory Server schema was constructed that supports the BIND-SDB interface.

5.6. Laboratory Design

All of the routers and servers were built using commodity desktop Personal Computer (PC) components. Figure 5.1 shows the rack layout diagram of the laboratory. The servers are color-coded by region. Figure 5.2 shows pictures of the finished laboratory.

5.7. Summary

The implementation presented in this chapter describes how the design for a high-availability DDNS solution was realized. This implementation is a representation of an ISP with three cooperating regions. Each region is a replica of the other two. Each region has a core router running BGP. Each region also has a DDNS server that connects to an LDAP directory to store and retrieve resource records. All directory servers replicate all resource records to each other using a multi-master replication mechanism. Details about the software selected and configuration specifics were also covered.

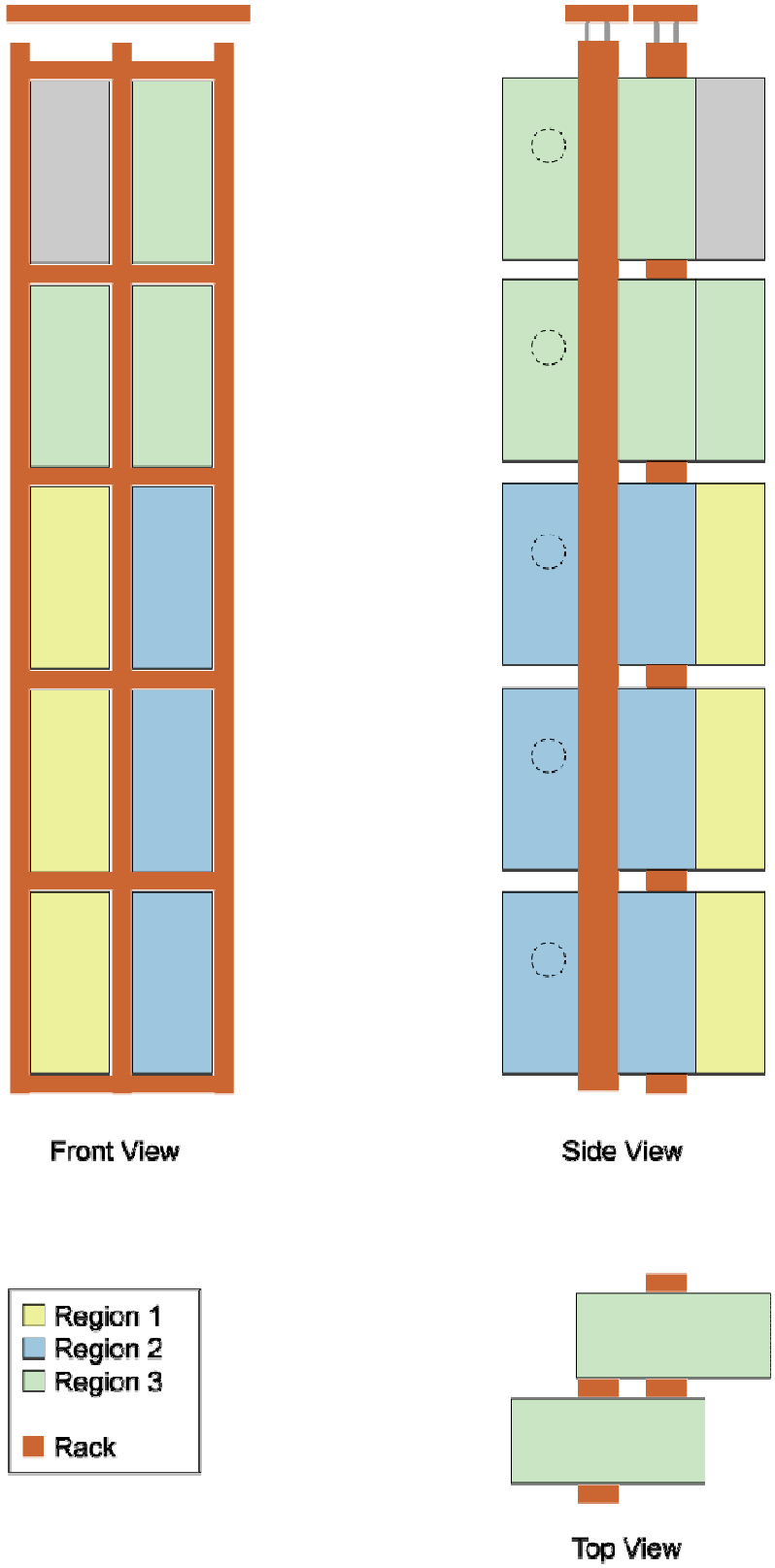


Figure 5.1. Laboratory rack diagram.

Figure 5.2. Laboratory rack images.

Chapter 6. Performance

In this chapter, the performance of the reference implementation presented in Chapter 5. is examined. The functionality of the solution is demonstrated. The objectives of the design are presented and defined. Metrics related to these objectives are also defined and evaluated. The first metric, downtime during failure, quantifies how long the outage due to a single-host failure affects clients of the DDNS solution. The replication overhead metric indicates the network traffic cost of the replication mechanism. The latency of the replication mechanism is also characterized. This chapter also analyzes the results of these experiments. The results show that this architecture should meet the requirements of a high-availability solution for DDNS.

6.1. Objectives

Along with the research objectives and functional objectives of the design, there are three major performance objectives of the implementation of the design. These objectives, availability, scalability, and consistency, are described below. Metrics that correspond to the objectives or, at least, give insight into the objectives are also defined. Finally, experiments that directly evaluate those metrics were conducted.

In the context of this research, **availability** refers to the percentage of time that the proposed solution can provide the DDNS service. This includes the ability to accept updates as well as to provide name resolution. A long-term and large-scale study of this solution, both beyond the scope of this research, would be required to accurately measure this property.

Scalability refers to the ability of the application of this solution to continue to provide an effective DDNS service as the network grows to a very large size. The intent of this design is that that it can be applied to a large ISP or the global Internet.

Another objective of this design is **consistency**. In terms of the data replication solution, consistency is the property that when the data is retrieved from one name server it has the same value as if it were retrieved from another name server. The assumption is that the data should also be correct and up to date.

6.2. Metrics

A basis for characterizing the availability of the system can be found by measuring the **downtime during failover**. This is a specific measure of the amount of time the name service is unavailable, starting at the time of failure of a single name server and ending at the resumption of service by another available name server, presumably the next "closest" name server. Using certain normalizing assumptions, this can be extrapolated to an uptime metric. For instance, if we assume one failure per year and take the amount of time to recover from that failure, then the availability for that year, expressed as a percentage, can be obtained. The actual number of failures per year is a random variable that can be estimated using Markov chains and mean time between failure (MTBF) values for the specific hardware. However, since this is implementation specific, it is beyond the scope of this research.

By measuring the **replication overhead** required to provide redundancy, one can examine the tradeoff between "cost" (network capacity consumption) and availability and consistency. The replication overhead is determined by measuring the network traffic cost without replication and comparing it to the network traffic cost with replication. The difference in traffic is the overhead cost of providing the redundancy. This gives an indication of scalability. If the network overhead were to increase exponentially, that might indicate a significant concern with the scaling of the solution. However, if the network overhead increases linearly, this would be an indication that the solution would scale up to large network sizes.

Another important metric is **latency**. In general, latency is the time between an event and some phenomenon associated with that event. The most interesting latency in this solution is the time between the event that a resource record is added to one name server and the time that all name servers respond with the new value. The metric of latency of replication is presented as a measure of consistency of the data storage system. The design of the multi-master replication scheme is intended to be loosely consistent. If experiments were to show that the there was a large amount of latency during replication, that would indicate poor consistency. However, if experiments show that latency is low, that would indicate good consistency. What constitutes "good" or "bad" consistency depends on the requirements of the specific application that uses the data. The property to examine with such applications is their tolerance for inconsistency. The

scope of this research is to describe how to obtain this metric. It is outside of the scope to determine appropriateness or target values for the latency or other metrics for specific applications.

6.3. Experimental Results

Three experiments were designed to evaluate the metrics defined in Section 6.2. for the laboratory implementation. These experiments were designed to isolate the stated parameters and demonstrate the property under investigation. Since the design and the test bed implementation are inherently complex, special attention was given to making the experiment as simple as possible.

6.3.1. Downtime During Failover

The design of this solution is such that during the outage of a single name server, other name servers are functional. That is, the outage of a single name server does not affect the other functioning name servers. From the perspective of the DNS client, the failover mechanism is provided by anycast routing, as shown in Figure 2.13. The client uses the IP address for the anycast group of name servers regardless of the functionality of any particular name server. Therefore, the downtime of the service, from the perspective of a particular client, is the time between the event when the misbehaving DNS server's route advertisement is withdrawn and the time when the new route to a functioning server converges. During this outage window, a client's requests are not delivered to a functioning name server. After this window ends, the client's requests are served by a functioning name server. From the perspective of the client, the outage has ended. Even though the faulty name server is still offline, there is no functional impact to the client.

This experiment was conducted as follows. The first name server host, ns-1 as shown in the diagram in Appendix B. Laboratory Network Diagram, runs an agent that monitors the health of the name service. This is done by querying the name service for a known resource record once per second. At this point, any clients attached to router rtr-1 can successfully query valid resource records via name server ns-1. Then, the name service daemon was manually shut down to simulate a failure. At this point, clients attached to rtr-1 can no longer query any name servers.

As soon as the name server stops responding to the agent's query, the agent withdraws the route advertisement of the anycast address from the BGP instance running on the name server. Meanwhile on the core routers, rtr-1 and rtr-2, the routing table is monitored on a one-second interval. At the time when the appropriate change to the route table occurs, route convergence is complete. Thereafter, clients attached to router rtr-1 can successfully query and receive responses from name server ns-2. The convergence time that elapsed is considered the downtime during failover. Experimental results are shown in Table 6.1.

It should be noted that there are apparent negative values in convergence time. This is due to the error in time synchronization between hosts. This error is a limitation of using Network Time Protocol (NTP) as a time synchronization method. The time synchronization error among all hosts is less than one second. The results show that the convergence delay is not actually measurable for this particular test bed and experiment. However, it does show that the convergence delay is bounded by an upper limit of one second. All hosts in this test bed are co-located and are within three hops of other nodes, so propagation and queuing delays are minimal. In a geographically distributed deployment, the propagation and router queuing delays would increase, which would also increase the total failover time. On real-world Internet core routers, the BGP routing tables will be much larger. This will have an adverse effect on BGP convergence time. A detailed analysis of the various contributions to BGP convergence time is beyond the scope of this work. However, others have treated this subject in detail [10, 18, 19].

A similar process occurs when the server process on ns-1 is restarted. The convergence time to begin using the restored server is longer in this case. The average convergence time for this case was 12 seconds. The maximum observed convergence time for this case was 30 seconds. However, this is not a concern because there is no outage window associated with this delay. This is because name server ns-2 is responding to name queries until the convergence completes, at which time, name server ns-1 is able to respond to name queries. This is in contrast to the failover case, because the network delivers packets to the non-functioning ns-1 name server, or drops them, until route convergence completes.

Table 6.1. Downtime during failover

Route Withdrawn ns-1 HH:MM:SS	Route Changed rtr-1 HH:MM:SS	Route Changed rtr-2 HH:MM:SS	Convergence Time rtr-1 (s)	Convergence Time rtr-2 (s)
03:47:14	03:47:14	03:47:14	0	0
03:59:25	03:59:25	03:59:25	0	0
04:00:42	04:00:43	04:00:42	1	0
04:02:06	04:02:05	04:02:05	(-1)	(-1)
04:04:17	04:04:18	04:04:18	1	1
04:06:24	04:06:24	04:06:24	0	0
04:08:56	04:08:56	04:08:56	0	0
04:10:26	04:10:27	04:10:26	1	0
04:11:58	04:11:58	04:11:58	0	0
04:13:38	04:13:37	04:13:37	(-1)	(-1)
		Mean	0.2	0.0

6.3.2. Replication Overhead

To develop a system that can withstand failure of a component, redundancy must be part of the solution. However, redundancy comes with a cost. In the multi-master replication directory implementation used in this research, there are network traffic costs associated with that replication. To get a notion of how this cost might affect the scalability of this solution, the cost must be measured. This experiment considers two cases. The base case considers the network traffic required simply to add the resource records to a single directory instance. The replication case considers the network traffic needed for replication. The overhead is then defined as the network traffic required by the replication case beyond the traffic required by the base case. Since the DHCP server's agent communicates via LDAP to the name server, only the traffic used by the LDAP protocol itself is considered. For the purpose of this experiment, TCP overhead, needed to manage TCP connections and as acknowledgments, is neglected.

In this experiment, 10,000 records were added from DHCP server dhcp-1 to DNS server ns-1. The base case uses a total of 3.99 Megabytes (MB) to transfer this data, as shown in Figure 6.1. In the replication case, shown in Figure 6.2, a total of 14 MB is transferred. The overhead incurred by replication is, thus, 10 MB. A three-fold increase in redundancy, going from one name server to three name servers, is achieved with a 3.5-times increase in network traffic.

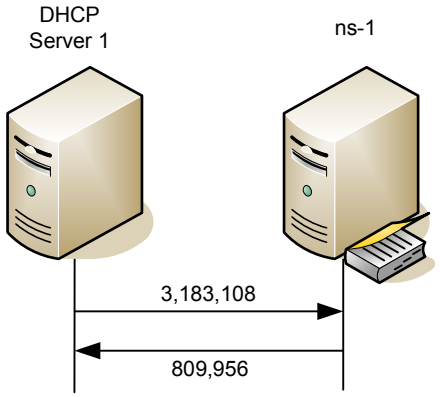


Figure 6.1. Replication base case (data amount in bytes).

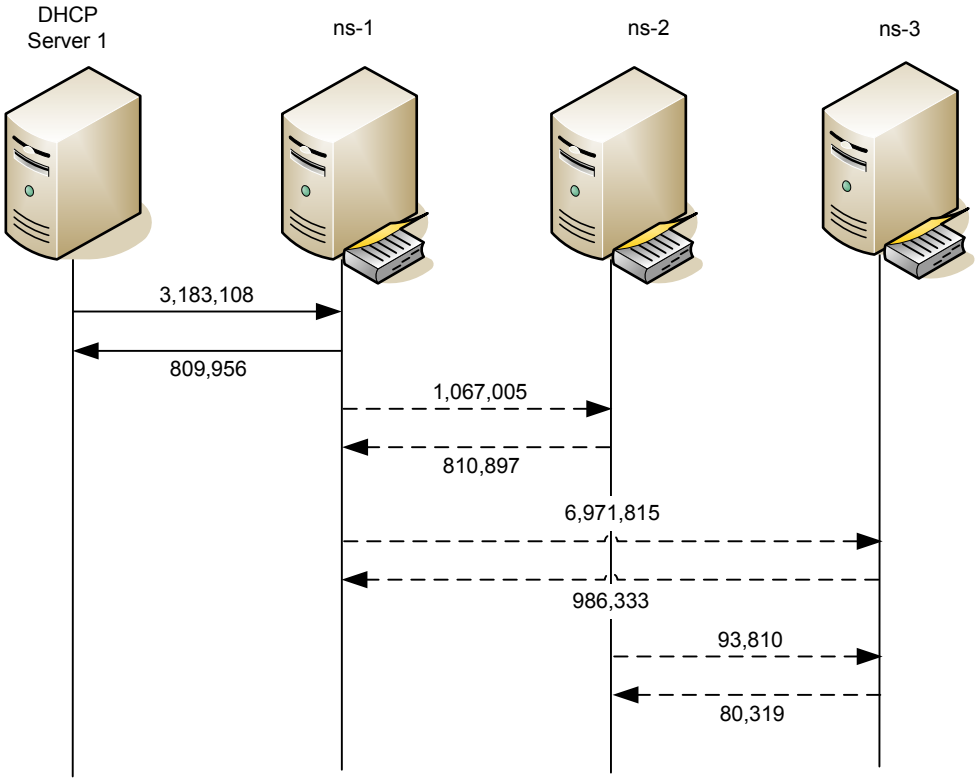


Figure 6.2. Replication overhead (data amount in bytes).

6.3.3. Latency

For the latency experiment, a resource record is added to name server ns-1 by the DHCP lease monitoring agent. The latency that was measured between the time that record is added to ns-1 and the time that the record is available on the name servers ns-2 and ns-3 was less than one second. This is below the NTP synchronization error threshold. Since the name servers are co-located, in a geographically large and multi-hop network, propagation and queuing time may become significant factors in the latency of the overall solution.

6.4. Analysis of Results

These results show that the implementation functions as designed. This verifies that the architecture presented is a viable high-availability solution for DDNS. While it is impossible to confirm without large-scale network testing, the performance observed indicates that this solution is likely suitable for use in a wide-area deployment serving a large number of clients.

The results of the downtime during failure experiment indicate that this solution should provide high availability even under various types of failure. Even if such a network were to have several different failures a year, with a downtime during failure on the order of one second, very promising availability numbers should be achievable. For reference, five minutes of downtime per year would be equivalent to 99.999%, or "five-9's" availability. Obviously, these experiments were conducted under laboratory conditions on a small network. The main limitation of this experimental design is that the entire network was located at the same site. However, this does demonstrate the performance of the system itself. Since delays incurred by long network distances and multiple hops are well-researched, it should be fairly straightforward to estimate the delay of deploying over large geographic distances for specific deployments.

The experimental results for replication overhead are also promising. The LDAP multi-master replication scheme supports up to four masters. The experimental design uses three masters. Given that a single non-redundant master creates a certain amount of network traffic, ideally adding a second server would only double the total network traffic. By that logic, adding a third server for increased redundancy could be expected to bring the total up to at least three times the original amount. Surprisingly, the total amount of network traffic for three servers was only 3.5 times the amount of a single server. This shows that the overhead, or cost, of increasing the availability of the system is reasonably low. The limitation of four masters should be noted, but it seems to be a reasonable one. Since DNS is hierarchical in nature, it should be straightforward to limit the number of zones on a set of name servers appropriately to stay within the capacity of those systems. If more capacity is needed, multiple sets of multi-masters can be used, applying the standard delegation principles of the DNS system and partitioning zones appropriately.

The latency experiments show that good consistency might be reasonably expected. Of course, the degree of tolerance of inconsistency of the application that is going to use the DDNS system

must be considered. Nevertheless, it should be noted that this is a loosely-consistent system. If it is determined that a given application requires latency of less than one second, an improved methodology for measuring latency must be developed. Another alternative would be to thoroughly test the application using this architecture and ensure that its performance is adequate.

6.5. Summary

The performance of the reference implementation was examined. It was shown that the implementation functions as designed. The design objectives, availability, scalability and consistency were defined. Metrics related to these objectives, downtime during failure, replication overhead, and latency, were evaluated. The downtime during failure was shown to be less than one second, the threshold of time synchronization for the laboratory. The replication overhead analysis shows that the network traffic cost of replication is reasonably low. The latency of the replication mechanism was also characterized. Applications that can tolerate a minimal amount of temporary inconsistency should be able to utilize the architecture presented. The performance results show that this architecture is a high-availability solution for DDNS, and that its use for applications over large networks is promising.

Chapter 7. Conclusions

This thesis presents an architecture that resolves an important limitation with the current DDNS implementation. This design uses an LDAP directory to store and retrieve DNS resource records. All records written to one directory server are replicated by that directory server to the remaining functional directory servers. This is the multi-master replication mechanism. The DHCP lease monitor agent writes to an assigned directory server. If that directory server is not available, the agent tries another directory server in its known list of directory servers.

This solution also makes use of anycast BGP routing for the DDNS servers. This is identical to the way that anycast BGP is used for DNS servers. All DDNS servers use the same IP address to listen for queries. The name server agent monitors the health of the name service and ensures that the advertisement to that anycast address will be withdrawn if the name service fails for any reason. When the name service anycast advertisement is withdrawn, the routing system converges on a new route to the next available name server.

The implementation presented shows the viability of this architecture. A reference implementation was built successfully and worked as designed. The design objective of high-availability was approximated by measuring the downtime during failover. This downtime was shown to be less than one second. For comparison 99.999% availability equates to 5 minutes of annual downtime. Scalability was another design objective. Scalability was characterized by the metric of replication overhead. Providing the redundancy of three name servers only increases the network traffic requirements by 3.5 times. The network traffic requirements were evaluated using a data set of 10,000 resource records. The last design objective, consistency, was evaluated in terms of the latency of replication. During an update of a resource record to a single name server, the resulting replication to the other name servers occurred in less than one second. This is the amount of time that the data might be temporarily inconsistent. These results show that this solution is not only successful, but is likely to function well in serving a large number of clients over a wide area network.

The main contribution of this research is to propose a novel architecture for high-availability DDNS. The presented design eliminates the single point of failure in the current DDNS

architecture. There has been no known prior work on reliability for DDNS. The observed problem with DDNS was not documented in any prior work. Before this research, anycast has only been used for static DNS. The architecture shows how anycast can be used for dynamic DNS in a robust way. The research demonstrated the viability of the solution with a reference implementation. The system was measured and showed a low downtime during failover, low network traffic replication overhead and a low latency of replication. The importance of this work is that it ensures the functionality of critical services that currently depend on DDNS. This research also expands the utility of DDNS to new application that require high-availability.

Although the system is shown to work well, there are several areas for future work. ISC's DHCP should be improved to have a plug-in architecture similar to BIND-SDB to allow it to store leases in the same directory that BIND uses to retrieve the resource records. ISC's DHCP could send DNS updates directly to BIND if BIND supported the SDB interface for updates. To do this, the DHCP server would need the capability of iterating through a list of DNS servers in case one DNS server was unavailable. Also, the schemas for OpenLDAP and Fedora Directory Server should be standardized, by a set of RFCs, for instance, that make different LDAP schemas interchangeable. There are already RFCs that define some schema standards. It would be easier to move applications across different directory servers if these schemas were truly interchangeable. Currently, there is a manual merge, trial and error process that is involved when moving an application from one directory server to another. To get this architecture into production quickly, the BIND monitor agent should be made more robust to handle more error conditions.

To improve performance for read access, the LDAP architecture can be easily expanded with hubs and consumers to distribute the read-only load across more machines. Also, there are opportunities to improve BGP convergence time by delegating the DNS zones appropriately and instituting local anycast strategies to handle heavy load regions.

Appendix A. Glossary

A – IPv4 Address Resource Record in DNS

AAAA – IPv6 Address Resource Record in DNS

AS – Autonomous System

BGP – Border Gateway Protocol

BIND – Berkeley Internet Name Domain

BOOTP – Bootstrap Protocol

DDNS – Dynamic Domain Name System

DDoS – Distributed Denial of Service

DHCP – Dynamic Host Configuration Protocol

DNS – Domain Name System

IANA – Internet Assigned Numbers Authority

IETF – Internet Engineering Task Force

IP – Internet Protocol

ISC – Internet Software Consortium

ISP – Internet Service Provider

LAN – Local Area Network

LDAP – Lightweight Directory Access Protocol

MB – Megabyte

MTBF – Mean Time Between Failure

MX – Mail Exchanger Resource Record in DNS

NAPTR – Naming Authority Pointer

NTP – Network Time Protocol

OSPF – Open Shortest Path First

PC – Personal Computer

RFC – Request for Comment

SDB – Simplified Database Backend

SRV – Service Resource Record in DNS

TCP – Transmission Control Protocol

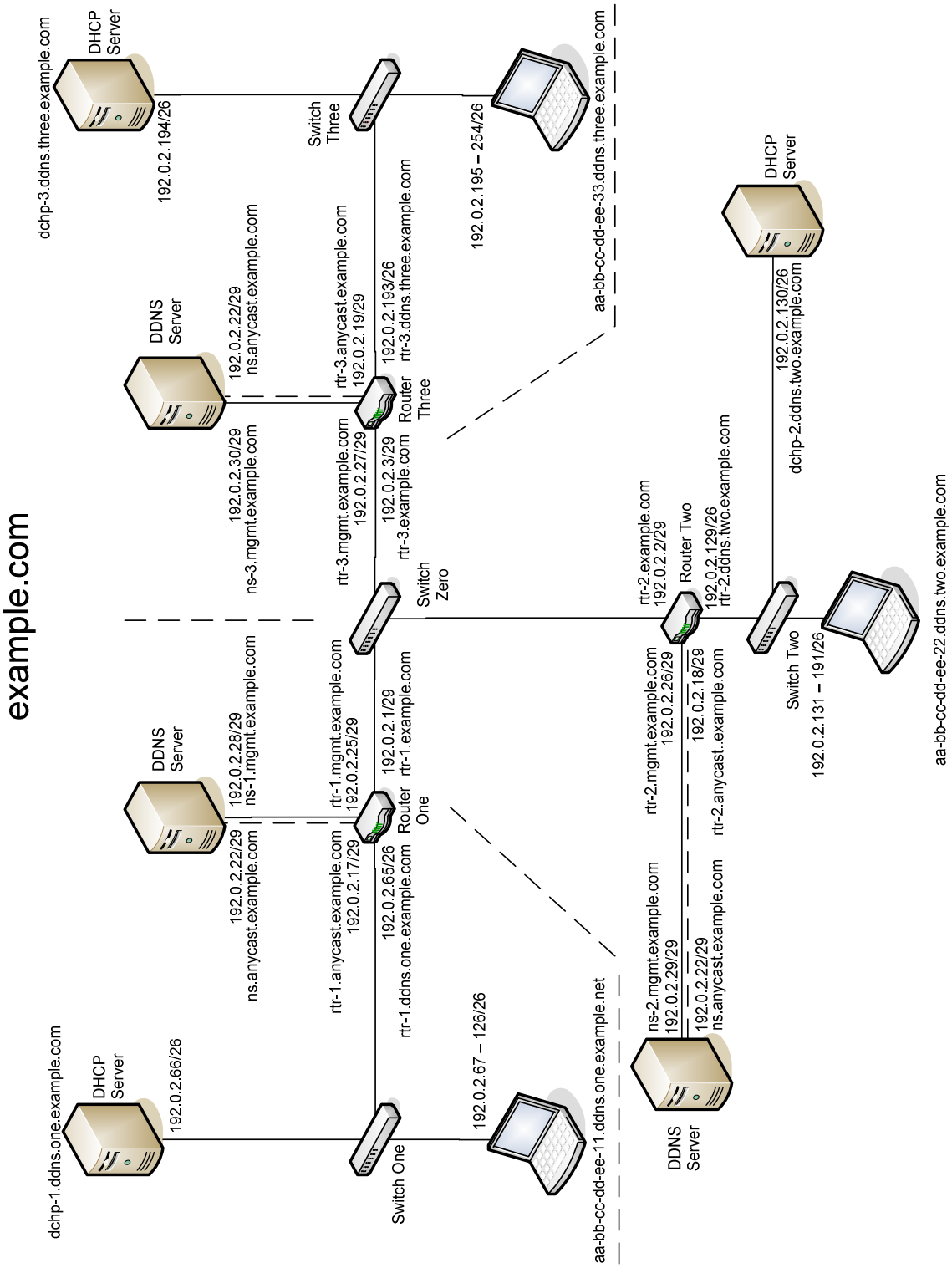
TLD – Top Level Domains

TXT – Text Resource Record in DNS

UDP – User Datagram Protocol

URL – Uniform Resource Locator

Appendix B. Laboratory Network Diagram



References

[1] P. Albitz and C. Liu, *DNS and BIND*, 5th ed.: O'Reilly, 2006.

[2] P. V. Mockapetris, "RFC 1034 - Domain names - concepts and facilities," Internet Engineering Task Force, Nov. 1987.

[3] R. Droms, "RFC 2131 - Dynamic Host Configuration Protocol," Internet Engineering Task Force, March 1997.

[4] P. Vixie, Ed., S. Thomson, Y. Rekhter, and J. Bound, "RFC 2136 - Dynamic Updates in the Domain Name System (DNS UPDATE)," Internet Engineering Task Force, April 1997.

[5] C. Partridge, T. Mendez, and W. Milliken, "RFC 1546 - Host Anycasting Service," Internet Engineering Task Force, Nov. 1993.

[6] "Anycast," Wikipedia. "http://en.wikipedia.org/wiki/Anycast", May 15, 2008.

[7] Y. Rekhter and T. Li, "RFC 1771 - A Border Gateway Protocol 4 (BGP-4)," Internet Engineering Task Force, March 1995.

[8] T. Hardie, "RFC 3258 - Distributing Authoritative Name Servers via Shared Unicast Addresses," Internet Engineering Task Force, April 2002.

[9] J. Abley and K. Lindqvist, "RFC 4786 - Operation of Anycast Services," Internet Engineering Task Force, Dec. 2006.

[10] S. Sarat, V. Pappas, and A. Terzis, "On the use of anycast in DNS," In *Proceedings of the International Conference on Computer Communications and Networks (ICCCN)*, 2006, pp. 71-78.

[11] J. Abley, "A Software Approach to Distributing Requests for DNS Service Using GNU Zebra, ISC BIND 9, and FreeBSD," In *Proceedings of USENIX Annual Technical Conference*, 2004.

[12] B. Hitesh, F. Paul, and R. Sylvia, "A measurement-based deployment proposal for IP anycast," In *Proceedings of the 6th ACM SIGCOMM conference on Internet Measurement* Rio de Janeriro, Brazil: ACM, 2006.

[13] H. Ballani, and Francis, P, "Understanding IP Anycast," *Cornell University Technical Report*, 2006.

[14] J. Hiebert, P. Boothe, R. Bush, and L. Lynch, "Determining the cause and frequency of routing instability with anycast," In *Proceedings of Technologies for Advanced Heterogeneous Networks II*, 2006, pp. 172-185.

[15] "MySQL 5.1 Reference Manual," MySQL. "http://dev.mysql.com/doc/refman/5.1/en/ha-overview.html", 2008.

[16] "Oracle Application Server Concepts," Oracle. "http://download.oracle.com/docs/cd/B15904_01/core.1012/b13994/avscalperf.htm", 2008.

[17] T. Howes and M. Smith, "RFC 1959 - An LDAP URL Format," Internet Engineering Task Force, June 1996.

[18] B. C. Zhang, D. Massey, and L. X. Zhang, "Destination reachability and BGP convergence," In *Proceedings of the IEEE Global Telecommunications Conference (Globecom)*, 2004, pp. 1383-1389.

[19] Z. Liu, B. Huffaker, M. Fomenkov, N. Brownlee, and K. Claffy, "Two days in the life of the DNS anycast root servers," In *Proceedings of Passive and Active Network Measurement*, 2007, pp. 125-134.