Components of a Scalable Web Hosting Platform using a Cloud and Peer-to-Peer Hybrid Architecture

by

Alexander Pokluda

CS854 Cloud Computing and Management
Research and Implementation Project

University of Waterloo
Waterloo, Ontario, Canada
April 2013

Abstract

This report summarizes the work done for the author's CS854 Cloud Computing and Management Research and Implementation Project. This work is also being contributed to the pWeb project, which is an effort to create a hybrid peer-to-peer and cloud-based platform for developing next-generation scalable web applications. The first chapter provides an in-depth overview of the pWeb project and describes the architecture and operation of the system with a focus on the device naming infrastructure that assigns permanent names to fixed and mobile devices. The second chapter gives a brief overview of the operation of the domain name system and an in-depth discussion of the architecture, design and implementation of the pWeb DNS Gateway–a new component of the pWeb system that has been implemented for this project. The DNS Gateway enables the pWeb device naming infrastructure to be integrated with the domain name system infrastructure. This allows pWeb and traditional web applications to resolve a device's permanent name to its current network address using the domain name system as an interface to the pWeb naming system. This chapter concludes with a summary of the performance of the DNS Gateway software as measured using ResPerf, a DNS performance measurement tool, which shows that the DNS Gateway software can easily handle at least 7,000 queries per second. The third chapter describes the Device Indexer system that is also being designed and implemented for the pWeb project. This system creates a searchable database of information about devices in the pWeb system.

Table of Contents

List of Figures

List of Listings

Chapter 1

Introduction

The pWeb project is a well established project aimed at creating a hybrid peer-to-peer and cloud-based platform for developing scalable web applications. Contemporary client-server web hosting applications deployed exclusively in the cloud face a number of challenges such as scalability, single point of failure, administrative overhead, and cost associated with maintaining a dedicated hosting infrastructure [2]. pWeb seeks to address these challenges by leveraging the strengths of both cloud and peer-to-peer infrastructures.

The current pWeb system architecture consists of five main software components. The core pWeb system is a three tier hierarchical architecture with instances of a unique software component deployed in each layer. These three tiers perform the core functions of device naming and content indexing and provide and consume content and services. Two additional components, the DNS Gateway and Device Indexer, operate outside the core pWeb infrastructure but nevertheless enhance the system with valuable functionality.

The work done for this project focused on contributing to a working implementation of the pWeb system that can be demonstrated at the sponsor's facility in coordination with other project members. We have successfully implemented a demonstrable system based on the three tier architecture using a top-down approach. The system currently implements the fundamental features and additional features will be added incrementally in subsequent iterations.

The pWeb system architecture including the DNS Gateway and Device Indexer are discussed next in Section 1.1, followed by a discussion of related work in Section 1.2. The

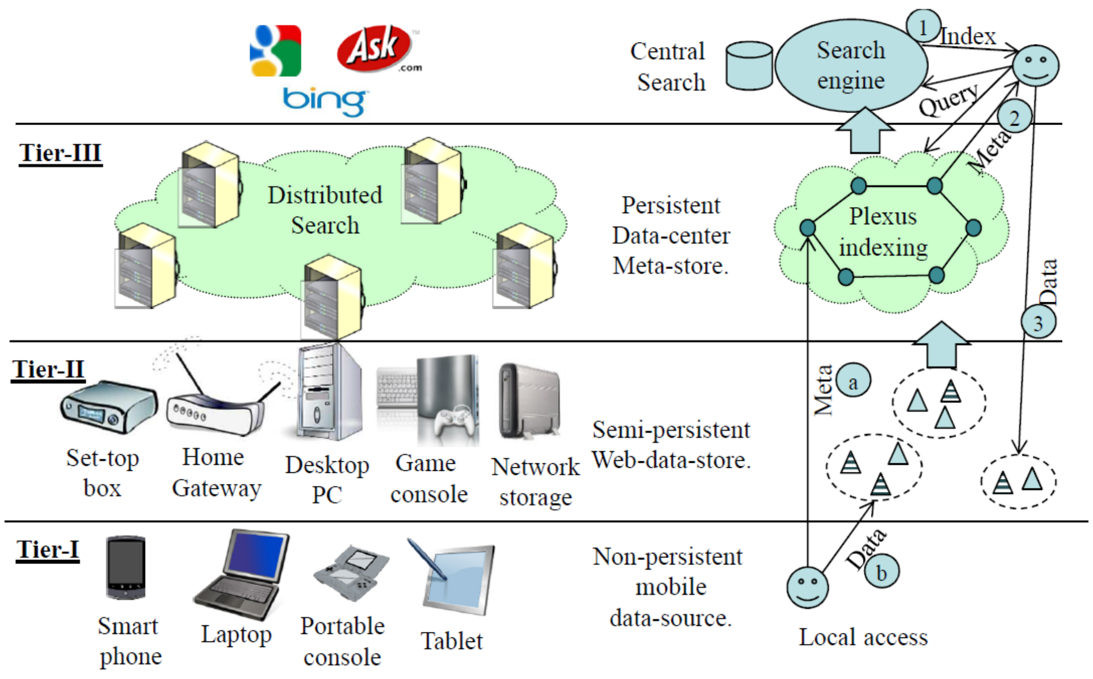


Figure 1.1: pWeb system architecture

status of the milestones that were presented in the proposal for this project are discussed in Section 1.3.

1.1 pWeb System Architecture

1.1.1 Core Architecture

The core of the pWeb system is a three tier architecture as shown in Figure 1.1. Nodes in Tier III perform distributed name resolution and indexing of devices in the pWeb network while devices in Tier I and Tier II both provide and consume application level content and services.

Nodes in Tier III are also known as pWeb Home Agents because they perform a role similar to that of the home agent in Mobile IP, an Internet Engineering Task Force standard communications protocol that is designed to allow mobile device users to move from one network to another while maintaining a permanent IP address [16, 17]; however, rather than provide devices with a permanent network address, pWeb Home Agents provide users with permanent device names that can be used to access services from the device regardless of the device's physical location and network address, similar to dynamic updates in the domain name system [21].

The infrastructure for Tier III consists of stable nodes at well known network locations. It is expected that organizations, institutions and individuals will contribute heterogeneous cloud resources to this tier by hosting instances of the pWeb Home Agent software in order to enable members of the organization or public at large to participate in the pWeb network. This is the same principle that is used by other similar and successful services such as the Jabber Instant Messenger service [10] and other services based on XMPP [8].

In addition to providing device name registration and network address resolution to devices in the pWeb network, the Home Agents make certain information available via an HTTP interface to enable the development of external applications. The Device Indexer, discussed shortly, makes use this feature.

Content and services in pWeb are provided by nodes in Tier I and Tier II that also participate in a peer-to-peer network. In general, these nodes are assumed to have shorter but regular uptimes and less bandwidth. Nodes in this tier may be servers, desktop computers, laptops or even mobile devices. A user publishes content using the client software running on one of these nodes. The client software will push the content meta-data, such as content name and location, to the cloud hosted indexing infrastructure in Tier III. The content itself will remain on the user's machine and be made publicly accessible over HTTP.

To ensure that content remains accessible despite peer churn and the limited resources of individual peers, the nodes in Tier II will be organized into replication groups. Each member of a replication group will mirror the content of the original publisher and make it publicly available as well. Nodes in Tier I are primarily publishers and consumers of content and use the public services offered by Tier I and Tier II to publish and access it.

1.1.2 Device Naming Services Overview

The Home Agent software is based on Plexus [3]. Plexus is a peer-to-peer search protocol and distributed hash table (DHT) that provides an efficient mechanism for advertising a bit sequence and discovering it using any subset of its 1-bits. When a new pWeb Home

```
Name    uw
Address 129.97.170.102
Address 129.97.172.102
```

Listing 1.1: Example Home Agent record inserted into Plexus DHT

Agent is brought online and joins the Tier III overlay network, it inserts its own name and network addresses into the distributed hash table. For example, a Home Agent hosted by the University of Waterloo would insert a record that is conceptually equivalent to the record shown in Listing 1.1. The `Name` field is a label, similar to a hostname, that uniquely identifies the Home Agent in the pWeb network. A label consists of a single word that may contain only the ASCII letters 'a' through 'z' in a case insensitive manner, the digits '0' through '9' and the hyphen '-'.

pWeb device names consist of three labels separated by periods, such as `mobile.alice.uw`. The leftmost label is assigned to a device and chosen by the device's owner. The middle label is the is chosen by and identifies the user, and the rightmost label identifies the home agent as mentioned above.

Registered device names and IP addresses are stored locally at the Home Agent where the device name was registered. This reduces the amount of data that must be stored in the DHT and makes device name registration and update operations much faster.

Home Agents also have the responsibility of resolving device names to network addresses, such as an Internet Protocol (IP) address, for their users. When a Home Agent receives a request to resolve a device name to network address, such as `desktop.bob.ut`, that is not registered locally, it extracts the Home Agent label from the device name, `ut` in this case, and looks up the network address for the remote Home Agent in the DHT. The local Home Agent then contacts the remote Home Agent and asks for the device's current network address, which it returns to the requester. Both the remote Home Agent's network address and the remote device's address will be cached at the local Home Agent for an appropriate period of time in order to serve future requests more quickly and reduce the load on the network.

This architecture for device name resolution and was chosen to support the explosive growth in smartphone services subscribers that is expected over the next few years. Recent forecasts by Credit Suisse, one of the world's largest and most profitable banks, predict that the number of global smartphone subscribers will reach 1.7 billion by the end of 2013 and nearly four billion by the end of 2017 [20]. A full analysis of the scalability of the device naming and Home Agent architecture is out of scope for this report, however.

1.1.3 DNS Gateway

Group formation and content replication in Tier II are not included in the fundamental feature set implemented in the current implementation of the pWeb system. Instead, devices in Tier I and Tier II are assumed to be always online and have sufficient bandwidth to serve the published content. Tier III will allow users to resister and resolve device names in order to share and access content. Users will be able to register names for Tier I and Tier II devices and update their network addresses using a name updater application that will function similar to a Dynamic DNS client [21] and will serve content using a standalone HTTP server. Tier III devices will use a standard, unmodified web browser to access content on the pWeb network. Names will be resolved by the web browser using a DNS compatible interface provided by a DNS Gateway that acts as an intermediary between the traditional domain name system (DNS) infrastructure and the Home Agents in Tier III of the pWeb architecture. This architecture is depicted in Figure 1.2.

Here content served by the device named `nexus.alice.uw` can be accessed by an unmodified web browser using the domain name `nexus.alice.uw.dyngw.org`, or the domain name `nexus.alice.uw.dht` if the local network has been configured to support the `.dht` pseudo top level domain that will be explained in Chapter 2. In the future, applications developed specifically for use with pWeb will have the ability to resolve device names to network addresses by contacting the device's Home Agent directly, bypassing the domain name system infrastructure; however, it is expected that the DNS Gateway will remain a critical piece of the pWeb infrastructure because the DNS Gateway enables seamless integration between pWeb and traditionally hosted web content. The DNS Gateway also enables pWeb content to be linked to by traditional hosted websites and web search engines to index content on pWeb, providing a unified browsing experience for the end user.

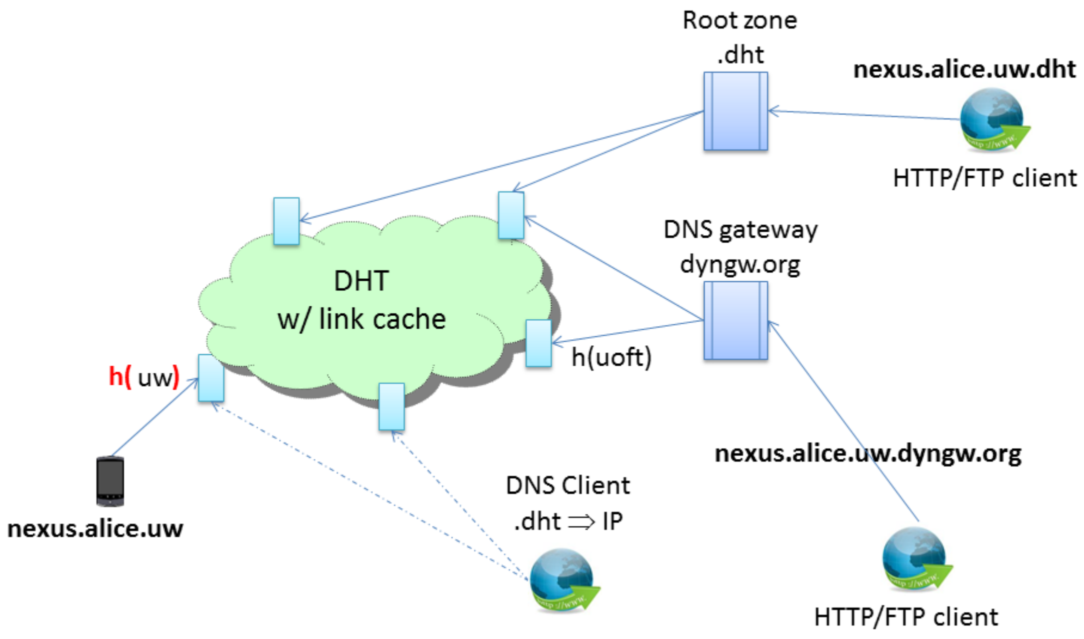


Figure 1.2: Integration of pWeb device naming with domain name system infrastructure

1.1.4 Device Indexer

The as previously mentioned, Home Agents in Tier III will make certain information available publicly over HTTP using a representational state transfer (REST) architecture [23]. Currently all registered device names and services offered by each device[1]and the Home Agent's neighbours in the Plexus peer-to-peer network are published using this interface. This information is made available to enable the development of internal and external applications that contribute to the functionality of the pWeb system.

The first application to take advantage of this interface is a Device Indexer that is being developed as part of the pWeb project. The Device Indexer itself consists of three main components: crawler and manager processes, a searchable database, and a web service frond-end. Multiple crawler processes, coordinated by a manager periodically poll each Home Agent's REST interface to discover all device names registered in the pWeb network.

[1]We plan to add extensive security and privacy controls to the Home Agent software that will enable users to strictly control the information that is made publicly available; however, the current version of the of the software makes all device names and last update times available publicly.

The device names are inserted into a searchable database that is made accessible to users through a web interface and other applications through its own REST interface. The Device Indexer and searchable database are represented by the Central Search in the top right corner of Figure 1.1.

1.2 Related Work

Other systems exist or have been proposed that share some similarities with pWeb; however, most of these systems are hybrid systems where a peer-to-peer network of caches is used to reduce the load on a traditional, centralized web serverthe most notable is Coral-CDN [9]. An exception is Freenet [5],which provides a web hosting platform based on a distributed peer-to-peer data store. Frennet and pWeb have similar goals; however, they are based on fundamentally different architectures. Freenet used a flat peer-to-peer architecture that prioritizes anonymity and censorship resistance, and as a result, content can be slow to access and dynamic pages are not supported. pWeb uses a unique three tier structured architecture that combines cloud and peer-to-peer resources. pWeb aims to democratize web hosting by providing individual publishers with the virtually limitless resources of a large peer-to-peer network with the stability and reliability of the cloud. pWeb will focus on providing support for dynamic content and an improved user experience with configurable security and privacy settings.

The implementation of the pWeb software is based on several previous publications that discuss various aspects of the system, including a high-level overview of the pWeb system [2] and a paper describing in detail the Plexus peer-to-peer protocol and distributed hash table [3] that constitutes a core component of the Home Agent software in Tier III. Extensive work has been done to define the pWeb system and much of this work is available in progress reports submitted to Orange Labs, one of the project sponsors. The most relevant report is [1] that describes the architecture, design and implementation of the pWeb system.

1.3 Proposed Milestones

Three categories of milestones were given in the proposal for the work to be done for the author's CS854 Cloud Computing and Management Research and Implementation project. The basic milestones included a minimal set of tasks that the author expected to accomplish in order to contribute to to making the working pWeb system described above. The additional milestones were tasks that the author hoped to accomplish provided that no significant difficulties were encountered while meeting the basic milestones. The extra milestones were tasks that the author would work on if the tasks for the basic and additional milestones were easier to complete than expected. The status of each of these three categories of milestones are discussed below.

1.3.1 Basic Milestones

The basic milestones were to design and implement the domain name system interface to the device naming services provided by Tier III in order to enable unmodified Tier I devices to access content on the pWeb network. The primary task associated with this was programming a name server that receives DNS requests, queries nodes in Tier III for the device's current network address and returns that address using a DNS reply. Additionally, an analysis of the scalability and performance of the DNS interface was to be performed. All of these milestones have been successfully met. The DNS Gateway, which implements the domain name system interface, is discussed in Chapter 2. The DNS Gateway software is distributed with this report.

1.3.2 Additional Milestones

The additional milestones were to design and implement a "name crawler." The purpose of the "name crawler" was to discover all registered device names and the services offered by those devices, and create a web accessible directory of them. Creating and maintaining a robust, functional and scalable database proved to be a challenge; nevertheless, the device indexing system has been designed and the implementation is nearing completion. The design and implementation are discussed very briefly in Chapter 3, Device Indexer.

1.3.3 Extra Milestones

The extra milestones were to assist with the development of a pWeb software application for Tier I and Tier II devices. This application integrates the name updater, a web server for publishing content, and a peer-to-peer communication component for establishing peer-to-peer connections with other devices in Tier I and Tier II. Throughout the duration of this project, the author assisted with a number of different tasks, including designing the software; researching and evaluating different embeddable web servers for desktop and mobile platforms; researching network address translation (NAT) traversal techniques and evaluating software libraries and servers that implement the techniques.

Chapter 2

DNS Gateway

As mentioned in the introduction, the pWeb DNS Gateway in an important piece of the pWeb infrastructure. It provides a seamless integration of pWeb and traditionally hosted web content and is the primary method of resolving a device's permanent name to its current network address in the current implementation of pWeb.

The pWeb device naming system has been designed to support billions of devices. Naturally, when using the domain name system as an interface to the pWeb network, we must consider the scalability of this interface. There are two aspects that must be taken into consideration: the load that is placed on the existing domain name system infrastructure before queries reach the DNS Gateway, and the scalability of the DNS Gateway itself. The domain name system infrastructure was not designed for host mobility, but previous studies have shown that the domain name system can be used as a location repository for mobile devices without placing undue stress on the global root nameservers [24, 25]. Thus, using the domain name system as an interface to the pWeb naming system should not place appreciable load on the root servers. The analysis of the scalability of the DNS Gateway itself is presented in Section 2.3, but first an overview of the domain name system is presented in Section 2.1 and a detailed description of pWeb device naming with the existing DNS infrastructure is given in Section 2.2

2.1 Domain Name System (DNS) Overview

The domain name system is a hierarchical distributed database that provides a naming system for computers and resources connected to the Internet. The domain name system associates human readable names with various record types. The predominant use of the domain name system–at least for human users– is to translate mnemonic names to resource Internet Protocol (IP) addresses. A domain name that is associated with at least one IP address is commonly referred to as a hostname.

The domain name system distributes the responsibility of assigning domain names and mapping domain names to network addresses to authoritative nameservers in each domain. The authoritative servers may in turn delegate responsibility for subdomains to other authoritative nameservers. The domain name system specifies the technical functionality of the database as well as the data structures and protocols for communication between domain nameservers and clients, and the algorithms that a nameserver will use to respond to queries against its database.

The domain namespace takes the form of a tree structure. Each node in the tree has zero or more resource records associated with it that associate specific information with the domain name. The tree structure subdivide domains into administrative zones, starting at the root zone. Each zone contains at least one domain and zero or more subdomains.

Domain names consist of a series of labels containing ASCII letters, digits and hyphen characters each terminated with periods, also known as dots. The last dot is frequently omitted. The exact rules for forming domain names are defined in [15, 4, 6]. An example of a domain name is `www.pwebproject.net`. The rightmost label, `net`, is known as the Top Level Domain (TLD). The domain `pwebproject` is a subdomain of `net` and `www` is a subdomain of `pwebproject`. A subset of the domain namespace that shows the administrative zones for the domain name `www.pwebproject.net` is shown in Figure 2.1.

2.1.1 DNS Address Resolution Mechanism

The operating system on and end host such as a laptop or desktop personal computer or Internet server typically contains a stub resolver that resolves hostnames to network addresses on behalf of applications on the machine. The stub resolver offloads most of the work of resolving a name to a recursive DNS resolver (also know as a nameserver) that is

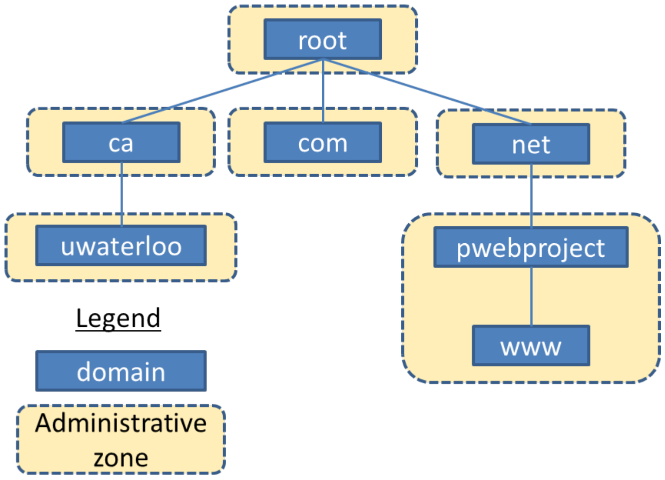


Figure 2.1: A subset of the domain namespace showing administrative zones

hosted in the local network or in the Internet Service Provider's network. The recursive DNS resolver uses the following algorithm to resolve a domain name, which is depicted in Figure 2.2. Note that the recursive DNS resolver performs a recursive operation from the client's perspective but actually queries other nameservers in an iterative manner.

1. When the recursive DNS resolver receives a query asking for the IP address corresponding to a domain name such as `www.pwebproject.net` it will look in its cache for the address of one of the root nameservers and forward the the query for `www.pwebproject.net` to the root nameserver. The cache is initially populated with a seed file containing the addresses of all known root nameservers and is periodically updated by system administrators from a reliable source.

2. The root nameserver will reply that it does not know the answer to the query but the authoritative nameserver for the `net` domain might. Thus, the root nameserver delegates responsibility to the subdomain `net` that is in a different administrative zone.

3. The recursive DNS resolver forwards the query for `www.pwebproject.net` to one of the authoritative nameservers for the `net` domain.

4. The authoritative nameserver for the `net` domain will reply that it does not know the

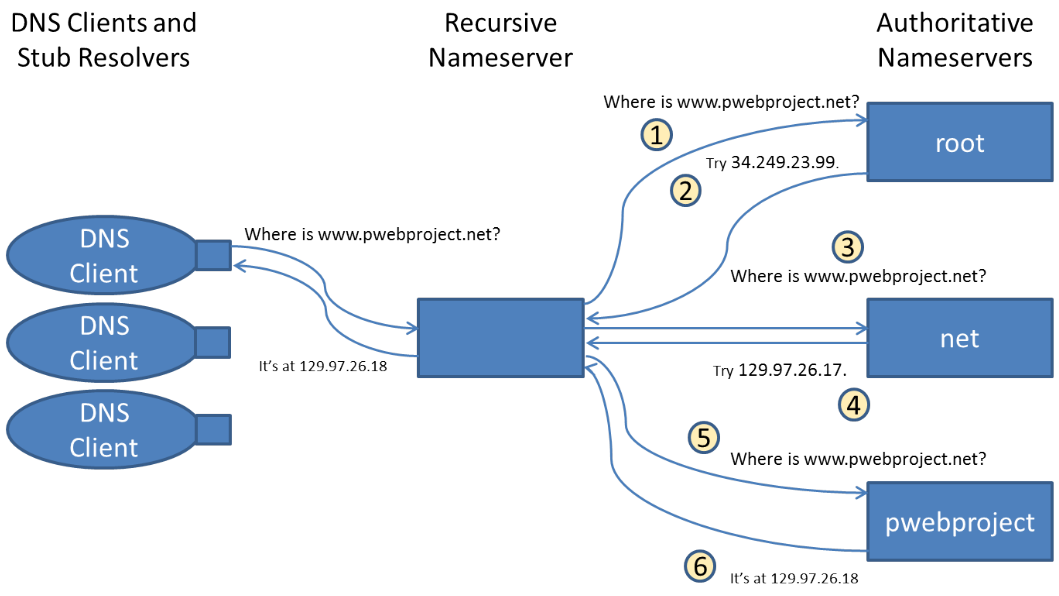

DNS Clients and Stub Resolvers
Recursive Nameserver
Authoritative Nameservers
Where is www.pwebproject.net?
1
root
Try 34.249.23.99.
2
3
Where is www.pwebproject.net?
DNS Client
Where is www.pwebproject.net?
DNS Client
net
It's at 129.97.26.18
Try 129.97.26.17.
4
DNS Client
5
Where is www.pwebproject.net?
pwebproject
6
It's at 129.97.26.18

Figure 2.2: How a recursive DNS resolver would resolve the domain name www.pwebproject.net

answer to the query but the authoritative nameserver for the pwebproject domain might.

5. The recursive DNS resolver forwards the query for www.pwebproject.net to one of the authoritative nameservers for the pwebproject.net domain.

6. The authoritative nameserver for the pwebproject.net domain will respond with the answer to the query. Note that the responsibility for the www subdomain is not delegated in this case. The www.pwebproject.net domain is in the same administrative zone as the pwebproject.net domain. Whether or not responsibility for a subdomain is delegated to another administrative zone is decided by the administrator of the parent domain.

In reality the algorithm as presented above would overload the nameservers near the root of the dns namespace. In order to reduce the load on these servers and improve the look-up time for subsequent requests, caching is employed at each stage in the name resolution process.

The delegation of responsibility from one administrative zone to another is achieved by an authoritative nameserver in the parent domain returning a nameserver record (NS) containing the addresses of authoritative nameservers in the subdomain.

2.2 pWeb Device Naming Integration with Existing DNS Infrastructure

The pWeb DNS Gateway supports deployment in two main configurations. It can be deployed using a gateway domain or using a pseudo-TLD. Both deployment configurations are described below. In each case, a DNS domain name is appended to the pWeb device name before the domain name system is used to resolve the device name to a network address. For example, when a gateway domain is used, the gateway domain is appended to the device name. Suppose and application needs to resolve the pWeb device name `mobile.alice.uw` to its current IP address and is using gateway domain `dht.pwebproject.net`. The gateway domain is appended to the device name, forming the pseudo-domain name `mobile.alice.uw.dht.pwebproject.net`[1]. The application then performs a standard DNS query for the pseudo-domain name in order to retrieve the device's current IP address.

Alternatively, the application could be configured to use a pseudo-TLD such as `dht` in order to resolve device names their current IP addresses. In this case, an application proceeds in the same manner by appending the TLD to the device name and performs a DNS query for the result. The difference is how the query is handled by the recursive nameserver handling the application's request. In either scenario, the query is eventually forwarded to a pWeb DNS Gateway server.

The method an application uses to determine whether a gateway domain or pseudo-TLD should be used for resolving pWeb device names is out of the scope of this report. However, one approach that could be adopted by applications designed specifically for use with pWeb is to perform a query using a pseudo-TLD, and if that fails, retry the query using a well-known gateway domain.

[1]Here the domain name `mobile.alice.uw.dht.pwebproject.net` is referred to as a *pseudo-domain name* because it is not a domain name as defined by the domain name system.

```
1 dht.pwebproject.net.            IN      NS      dnsgw1.pwebproject.net.
2                                 IN      NS      dnsgw2.pwebproject.net.
3 dnsgw1.pwebproject.net.         IN      A       129.97.26.34
4 dnsgw2.pwebproject.net.         IN      A       129.97.26.35
```

Listing 2.1: Nameserver records for `pwebproject.net` nameservers

Integration Using a Gateway Domain

In order to set up a pWeb gateway domain, the parent of the gateway domain must be configured to delegate queries for the domain to one or more DNS Gateway servers. This is achieved by configuring NS records in the parent domain.

The domain `dht.pwebproject.net` is currently set up as a DNS gateway to the pWeb naming system and is available for public use. It will will be used as an example in this section to explain how a gateway domain works. The domain `pwebproject` has been registered as a subdomain of the `net` top level domain[2]. As part of the registration process, NS records for the `pwebproject` subdomain were created in the `net` domain, delegating responsibility for queries to authoritative DNS servers in the `pwebproject.net` authoritative zone.

Each of the authoritative nameservers for the `pwebproject.net` domain are in turn configured with NS records that delegate responsibility for the `dht.pwebproject.net` subdomain to DNS Gateway servers[3]. DNS resource records, such as NS records, have a standard textual representation that is defined in [15]. The textual representation of the NS records for the `dht.pwebproject.net` domain are shown in Figure 2.1. Lines 1-4 Figure 2.1 show that there are two authoritative nameservers for the `dht.pwebproject.net` domain at the IP version 4 addresses 129.97.26.34 and 129.97.26.35, denoted by A records. When a DNS query arrives at a DNS Gateway server, the gateway and one or more Home Agents will find the device's current IP address and the gateway will return the result as a DNS reply. The procedure that the DNS Gateway and Home Agents use to process a query is outlined below and depicted in Figure 2.3.

1. When the DNS Gateway receives a query, it checks that the domain being queried

[2]The registration of domain names under top level domains is out of the scope of this report, but many resources explaining the registration process can be found on the web.

matches its configuration. If it does not match, the gateway returns an error. Next the device name is extracted and a query for the device's current IP address is sent to a Home Agent in the pWeb network. The DNS Gateway to Home Agent protocol is discussed in Section 2.4.1. The DNS Gateway must be configured with the hostname of one or more Home Agents. The DNS Gateway configuration is discussed in Section 2.4.2.

2. When the Home Agent receives the query, it extracts the device's Home Agent identifier from the device name. It then retrieves the Home Agent's IP address from its cache or the DHT.

3. The Home Agent that received the query then contacts the device's Home Agent and asks for the device's current IP address. The device's current IP address is returned to the DNS Gateway and it returns the current IP address to the requester using a DNS reply. The current version of the DNS Gateway software does not cache the results from Home Agents, but it does specify a configurable Time To Live (TTL) value in the DNS reply sent back to the recursive DNS server. This TTL value specifies the amount of time that the recursive DNS server should cache the result obtained from the DNS Gateway.

Integration Using `.dht` Pseudo Top-Level Domain

Recall that in a typical scenario, stub resolvers on an end system such as a desktop or laptop machine, offload most of the work of resolving domain names to a recursive DNS server in the local network or in the Internet Service Provider's network. The recursive DNS server will then query the a root server and any other nameservers that it is referred to in order to resolve the domain name to an IP address. In order to support a pseudo-top level domain in the network serviced by the recursive resolver, and entry for the `dht` pseudo-top level domain is permanently added to the recursive DNS server's cache. This entry contains NS records that delegate responsibility for the `dht` domain to pWeb DNS Gateway servers that are also hosted in the local network.

[3] A subdomain of the `pwebproject.net` domain is used as the DNS Gateway domain rather that `pwebproject.net` domain itself so that other subdomains, such as `www.pwebproject.net` can be crated as well.

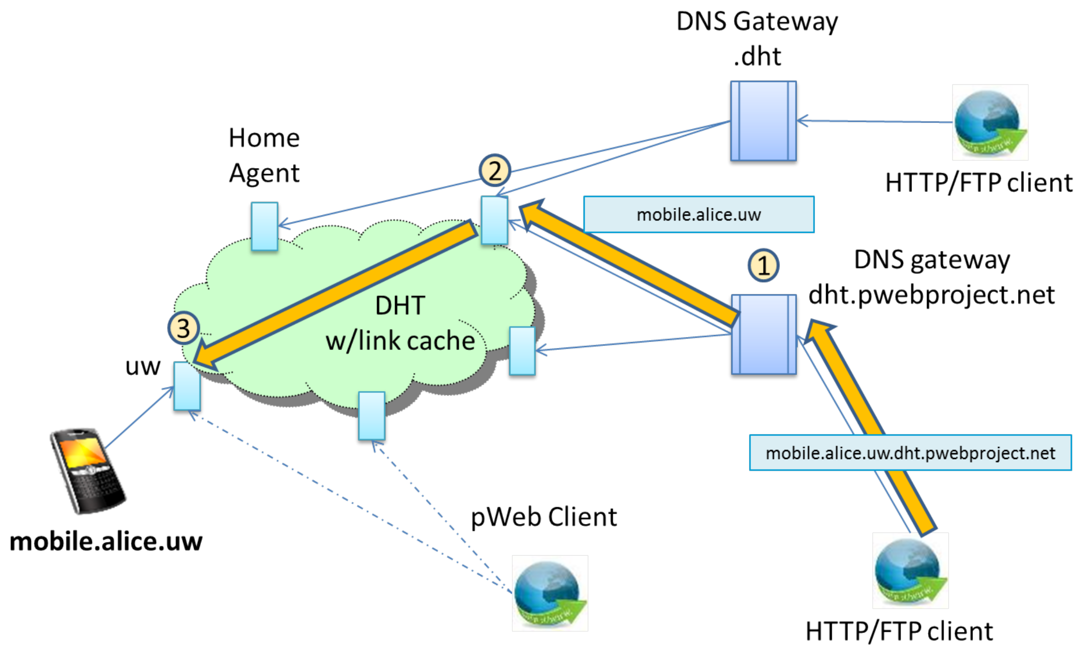


Figure 2.3: Procedure used by the DNS Gateway and Home Agents to resolve a device name to IP address

When using a gateway domain, no non-standard configuration is required at any point within the domain name system, and by default a DNS Gateway server set up using a gateway domain will be globally accessible. By contrast, when using the pseudo-top level domain, changes must be made to the nameservers in the network to be served and the pseudo-top level domain will only be usable within that network. However, this approach is less centralized than the gateway domain approach, provides more redundancy and is more scalable.

Interception of a Gateway Domain in Local Network

For a well-known gateway domain, such as `dht.pwebproject.net`, the recursive DNS nameservers for a network could be configured in order to "intercept" queries against this domain and forward them to local pWeb DNS Gateways. This approach has the advantage that applications designed for use with the pWeb system could be configured to use a globally accessible DNS Gateway, but when the application is used within a network supporting this type of interception, the query is forwarded to a local DNS Gateway server, improving the response time and reducing the load on the globally accessible DNS Gateway servers.

2.3 Scalability

The pWeb DNS Gateway has been designed to be extremely scalable. In its basic form, the DNS Gateway translates messages between the DNS protocol and the Home Agent protocol. No communication takes place between DNS Gateway servers and any number can be deployed in parallel. Thus, the pWeb DNS Gateway is not an architectural bottleneck in the pWeb device name resolution system. Additionally, the implementation of the DNS Gateway software has been designed so that it can take full advantage of modern server hardware. More details about the efficiency and scalability of the implementation can be found in following section and Section 2.8.

2.4 Design and Implementation

The pWeb DNS Gateway has been programmed from scratch with scalability and performance in mind. The DNS Gateway exclusively uses asynchronous I/O for network communication through the Boost Asio library [13] that is a cross platform C++ library for network and low-level I/O programming and has been proposed for inclusion of the next version of the official C++ standard [12]. The Asio library uses the most efficient asynchronous I/O API offered by the operating system on each platform that it supports [22, 14]. For example, on Linux 2.6, the Asio library uses epoll as the socket event notification mechanism.

Network servers have traditionally be programmed using a process-per-connection or a thread-per-connection model. The overhead of these strategies limit the scalability of the software on modern hardware [11, 22]. In this approach one process or one thread (or even two threads in some cases) is created to service a single TCP connection. The creation of processes or threads is a relatively expensive procedure that consumes both processor time and memory. After the process or thread is created, the application issues a read or write system call whenever it is ready to read or write data from the connection that blocks until the operation completes, preventing the process or thread from doing any other useful work. A large number of processes and threads causes the operating system to spend a significant amount of time switching between them and scheduling times for them to run. Most network servers that use a process-per-connection or thread-per-connection strategy cannot handle more than ten thousand concurrent connections even on modern hardware. This scalability limit is known as The C10K Problem [11]. Blocking operations on UDP sockets cause similar scalability problems, although the effectf is much less severe.

The asynchronous I/O approach, however, has been designed and implemented to ensure that the software is not a bottleneck to scalability and can take full advantage of modern hardware with multicore processors and fast network connections. In the asynchronous I/O approach, all network operations such as establishing a connection or performing a read or write operation, are non-blocking or asynchronous. This decouples the threading model from network operations, enabling the application designer to use as many or as few threads as they see fit. The pWeb DNS Gateway uses one application thread by default, but provides a configuration parameter to increase the number of threads, with an option to match the number of application threads to hardware cores. Matching the number of application threads to the number of hardware cores ensures that DNS Gateway can take advantage of all available processing power if necessary on the host machine.

The DNS Gateway uses both TCP and UDP connections. The DNS protocol defaults to UDP for simple queries in order to avoid paying the cost of establishing a TCP connection. TCP connections are used when a query or reply is too large for a UDP datagram and for more complex operations. The DNS Gateway to Home Agent protocol, discussed in the next section, uses TCP exclusively.

The DNS Gateway has flexible configuration and logging subsystems that are discussed in Section 2.4.2 and Section 2.4.3 respectfully. The DNS Gateway also has an instrumentation subsystem that is discussed in Section 2.4.4. A discussion of known implementation issues is discussed in Section 2.4.5.

2.4.1 DNS Gateway to Home Agent Protocol

The DNS Gateway to Home Agent protocol is a custom message-based application protocol over TCP. The DNS Gateway to Home Agent protocol uses two types of messages: PeerInitiateGET messages are sent from the DNS Gateway to Home Agents in order to request a device's current IP address, and MessageGET_REPLY messages are sent from the Home Agent to DNS Gateway to answer device IP look-up requests. All messages are sent over TCP connections. For every message sent, the sender opens a TCP connection, writes the message to the socket and then closes the connection. The full specification of the protocol is included in Appendix A. The format for the protocol was chosen in order to simplify the connection handling code in the Home Agent software. Unfortunately, this format is not very appropriate for the DNS Gateway to Home Agent protocol and suffers from the following limitations.

- A new TCP connection must be established for each message sent and received. This introduces connection establishment overhead and latency to the communication protocol.
- Most of the fields in the messages exchanged are unused. This means that most of the data exchanged over the network between the DNS Gateway and Home Agents serves no useful purpose.

- Both the Home Agent and DNS Gateway must be able to accept incoming connections. This introduces administration overhead when the DNS Gateway is behind a firewall because additional firewall ports must be opened.
- The DNS Gateway does not attempt to establish a connection to the Home Agent until it has a query to send. This means that the DNS Gateway does not know in advance if the Home Agent is inaccessible because it has crashed or is permanently or temporarily offline.

A different protocol that does not suffer from these limitations was originally proposed for use between the DNS Gateway and Home Agents. This protocol has been implemented in the DNS Gateway but has not yet been adopted by the Home Agent software. The specification for this protocol is included in Appendix B.

2.4.2 Configuration

The DNS Gateway software uses the Boost Program Options library [18] for runtime configuration management. Most configuration options have long and short forms and can be specified on the command line or in a configuration file. Short options specified on the command line must be proceeded by a single dash, `-`, and long options must be proceeded by a double-dash, `--`. Some options require an argument. An argument is specified immediately after and separated from the option name by either a space or an equals character, '=', on the command line. In a configuration file, an equals sign must be used. More information on the configuration file format is available in the Program Options library documentation at the following link: `http://www.boost.org/doc/libs/1_53_0/doc/html/program_options/overview.html#idp120049152`. When run with the option `--help`, the DNS Gateway will print a summary of the available configuration options and exit. This summary is shown in Listing 2.2. The `--version`, `--help` and `--config` options may be specified only on the command line while all remaining options may be specified on the command line or in a configuration file.

```
pWeb DNS Gateway v1.0.0
Allowed options:

Generic options:
  -v [ --version ] [=arg(=1)] Print program version and exit
  -h [ --help ] [=arg(=1)]    Print summary of configuration options and exit
  -c [ --config ] arg         Path to configuration file

Configuration:
  --ttl arg (=3600)                    Number of seconds DNS clients will be
                                       instructed cache name to IP mappings
  -t [ --timeout ] arg (=12)           Maximum number of seconds to wait for
                                       response from a Home Agent before
                                       returning an error to the DNS client
  -l [ --log_file ] arg (=dnsgw.log) Log file path
  -L [ --log_level ] arg (=WARN)     Log level
                                           Only log messages with a level less
                                       than or equal to the specified severity
                                       will be logged. The log levels are NOTSET <
                                       DEBUG < INFO < NOTICE < WARN < ERROR <
                                       CRIT  < ALERT < FATAL = EMERG
  -i [ --iface ] arg                   IP v4 or v6 address of interface to listen
                                       on for DNS queries
  -p [ --port ] arg (=53)              TCP and UDP port to listen on for DNS
                                       queries
  -H [ --home_agent ] arg              List of Home Agent addresses to connect to
                                           Any number of Home Agent addresses may
                                       be specified, separated by commas. Each
                                       address should have the form '<hostname or
                                       IP address>:<port>'. The DNS Gateway will
                                       use all the Home Agent addresses specified
                                       in a round-robin manner.
  -N [ --nshostname ] arg              The hostname of the DNS gateway
                                           This hostname is included in DNS
                                       replies and in messages sent to to Home
                                       Agents. Home Agents will send replies to
                                       this hostname using separate TCP
                                       connections.
    -P [ --nsport ] arg                TCP port used to receive replies from Home
                                       Agents
    -s [ --suffix ] arg (=.dht.)       Suffix to be removed from domain names in
                                       order to obtain pWeb device names
    --threads arg (=1)                 Number of application threads
                                           Set to 0 to use one thread per
                                       hardware core
    --instsrv arg                      Address of instrumentation server
                                           The address should have the form
                                       '<hostname or IP address>:<port>'
```

Listing 2.2: DNS Gateway configuration options

There is one additional undocumented option that can be specified on the command line or in a configuration file: the option `--debug` with short form `-d` enables debugging mode. In debug mode all log messages are printed to the console as well as the log file.

2.4.3 Logging

The DNS Gateway uses log4cpp [7] as its logging subsystem. Log4cpp is a very efficient and flexible logging library modelled after the popular log4j library. It can be easily configured to log to multiple destinations using multiple formats. The DNS Gateway currently supports logging to a file only, but support for other destinations, such a syslog daemon, may be added in the near future.

2.4.4 Instrumentation

The DNS Gateway contains a custom instrumentation service. The DNS Gateway collects information about every query it receives including the device name being queried, the time the query was received, the amount of time taken to process the query with sub-millisecond accuracy on most platforms, and a status code that identifies any errors encountered while processing the request. In the current implementation, the instrumentation data is serialized using the Boost Serialization library [19]. Instrumentation data is collected in a buffer that uses a Nagle timer before being sent over UDP. Serialized instrumentation objects are on the order of tens of bytes. The Nagle timer helps reduce the overhead of the UDP protocol by packing as many instrumentation datagrams as possible into a single UDP datagram during one Nagle period.

The purpose of the instrumentation library is to facilitate the real-time monitoring of one or more DNS Gateways. Instrumentation data could be fed into a network operations centre's monitoring system and used to produce real-time charts of the Gateway's performance and status indicators on a dashboard or heads-up display.

2.4.5 Known Issues

The current version of the DNS Gateway has a few known limitations. Each of these limitations are relatively easy to correct and may be fixed in future versions of the DNS Gateway.

- The device name to IP mappings retrieved from the Home Agents are not cached in the DNS Gateway itself although the results will be cached by the DNS servers querying the DNS Gateway.
- DNS queries contain a 16-bit serial number that is used to match queries and responses. The DNS Gateway copies the DNS query serial number to the PeetInitiateGET message sent to a Home Agent and treats all serial and sequence numbers globally. This has reliability and security implications. A more robust solution would be to use a larger namespace for sequence numbers in PeetInitiateGET messages that is formed by hashing the DNS client's IP address and port with the DNS serial number.
- The DNS Gateway does not implement support for some standard DNS resource record types such as NS and SOA records.

Standards Compliance

The DNS Gateway has been designed to be compliant with RFC 1035 [15]. This RFC defines the DNS resource record formats and algorithms used by DNS servers in the domain name system. This is still the canonical definition of the domain name system; however, the system was designed to be extensible and many new features, such as extensions for compression and encryption, are now in widespread use but not currently implemented in the DNS Gateway.

2.5 Installing the DNS Gateway from Source

The DNS Gateway is written in portable C++ and makes use of several portable, high quality C++ libraries. The following sections describe the DNS Gateway dependencies and build procedures.

2.5.1 Dependencies

The DNS Gateway depends on the Boost and log4cpp libraries. The DNS Gateway distribution also includes two additional tools for testing and debugging the gateway and it's configuration. One of these tools requires the cURL library. The libraries may be downloaded and installed from their project websites[4]or from your platforms package management system; however, the DNS Gateway requires Boost version 1.53 or newer, which is not available in the official Ubuntu repositories for the current versions of Ubuntu. Boost 1.53 for Ubuntu Precise Pangolin (12.04 LTS) and Quantal Quetzal (12.10) can be installed from this author's PPA on Launchpad at `https://launchpad.net/~apokluda/+archive/boost1.53`.

The DNS Gateway also includes a few tools for testing and debugging the gateway.

2.5.2 Building

The DNS Gateway uses the CMake build system. Listing 2.3 shows the commands to build and install the DNS Gateway from source on Unix-like platforms. The recommended way to build and install the DNS Gateway on Windows platforms is to use the GMake GUI. Instructions on how to use the CMake GUI are available on the CMake website at `http://www.cmake.org`.

[4]The Boost libraries are available from `http://www.boost.org/`, the log4cpp libraries are available from `http://log4cpp.sourceforge.net/`, and the cURL libraries are available from `http://curl.haxx.se/libcurl/`.

```
alex@alex-desktop:~$ tar -xzf pWeb-1.0.0-Source.tar.gz
alex@alex-desktop:~$ cd pWeb-1.0.0-Source/
alex@alex-desktop:~/pWeb-1.0.0-Source$ cmake .
alex@alex-desktop:~/pWeb-1.0.0-Source$ make
alex@alex-desktop:~/pWeb-1.0.0-Source$ make install
```

Listing 2.3: Commands to build and install the DNS Gateway on Unix-like platforms

2.6 Installation as a System Service

The recommended way to run the DNS Gateway is as a system service. On Unix-like platforms, the recommend way to configure the DNS Gateway to run as a system service is to use the daemon utility. The daemon utility can be installed on Ubuntu platforms by installing the system `daemon` package. Once the package is installed, you must configure your system to start and stop the DNS Gateway daemon when the system boots and shuts down. Ubuntu systems use Upstart for this purpose. Listing 2.4 shows an Upstart script that can be used to run the DNS Gateway as a system service using the daemon utility on Ubuntu. The script should be saved to `/etc/init/dnsgw.conf`.

```
description "pWeb DNS Gateway Server"
author "Alexander Pokluda <apokluda@uwaterloo.ca>"

start on runlevel [2345]
stop on runlevel [!2345]

exec daemon --user=daemon:daemon --name=dnsgw --respawn -- \
    /usr/local/bin/dnsgw -c /etc/dnsgw.conf
```

Listing 2.4: Upstart script to run the DNS Gateway as a system service on Ubuntu

2.7 Test Tools

The DNS Gateway distribution contains two test tools that can be used to aid in development and testing of the DNS Gateway. The program `dnsgw_test_client` is designed to check the status of a set of home agents and/or register device names at a set of home agents. The tool has been designed to be extremely scalable and uses the same design principles as the DNS Gateway itself in order to support potentially tens of thousands of Home Agents very efficiently.

The program `dnsgw_test_ha` is a mock home agent. This program includes a subset of the Home Agent code. In particular, it uses the same connection handling and message parsing and composing code as the Home Agents. The difference is that rather than query the DHT and/or a remote Home Agent for a device's current IP address it will return either the IP address 6.6.6.6 or a Name Not Found error for each query. The default is to respond to each query immediately, but the tool can also be configured to wait a specified amount of time before sending a response in order to simulate the latency associated with performing a look-up operation in the DHT followed by contacting a remote Home Agent.

2.8 Performance Evaluation

The `dnsgw_test_ha` test tool from the previous section has been used to estimate a *lower bound* on the performance of the DNS Gateway using the ResPerf DNS performance measurement tool[5]that is designed to estimate the latency and throughput of DNS servers. The test was run with all three components–the DNS Gateway, measurement tool, and mock home agent–on a quad-core desktop PC with 16 GiB of RAM. Queries for pWeb device names were generated by the measurement tool and sent to the DNS Gateway. The DNS Gateway in turn issued a query for the device's current IP address to the mock Home Agent. The mock Home Agent was configured to return the IP address 6.6.6.6 immediately. The report produced by the ResPerf measurement tool is included in Appendix C.

.

As can be seen in the report, the performance of the DNS Gateway and mock Home Agent starts to suffer at about 4 seconds into the test. This is because the mock Home Agent is unable to keep up with the increasing rate of queries generated by the measurement tool. When the DNS Gateway does not receive a response from the mock Home Agent after a short period of time, it responds to the measurement tool's query with an error

[5]The ResPerf DNS performance measurement tool is available from Nominum at http://www.nominum.com/support/measurement-tools/.

message. The statistics section of the report shows that the DNS Gateway responded to all queries issued by the measurement tool. The pWeb Home Agent uses a traditional thread-per-connection design as discussed in Section 2.4 that limits its scalability. This test shows the benefit of the asynchronous I/O approach used by the DNS Gateway and that the DNS Gateway can achieve a throughput of *at least* 7,084 queries per second using one application thread.

2.9 Summary

This chapter discussed the system and software architecture, design and implementation of the pWeb DNS Gateway. The DNS Gateway is provides a bridge between the domain name system infrastructure and pWeb device naming system and helps provide a seamless integration between pWeb and traditionally hosted web content. It is and will continue to be an important part of the pWeb infrastructure.

Chapter 3

Device Indexer Overview

The pWeb Device Indexer is still under active development but the implementation is nearly complete. The crawler consists of crawler and manager applications that are being programmed from scratch. The design and implementation of the crawler and manager parallels that of the DNS Gateway and test tools. They use efficient and scalable asynchronous I/O for network communication, the Boost Program Options library for runtime configuration, the log4cpp library for logging and the cURL library for accessing HTTP interfaces.

The architecture of the Device Indexer system is shown in Figure 3.1. The crawler processes are responsible for pulling device metadata from the pWeb Home Agents using their HTTP interface. Whenever a crawler discovers a new Home Agent, it informs the manager which then assigns responsibility for monitoring this Home Agent to one of the crawler processes. After discovering device metadata, the crawler processes feed the data in to a cluster of Solr Enterprise Search servers[1].

One or more web servers running the Django web framework provide a web front-end that can be used by human users as well as other applications in order to search for devices in the pWeb network based on the published metadata.

[1]Solr is an open source top level Apache project that is related to Hadoop, Lucene and Nutch. The Solr homepage is located at http://lucene.apache.org/solr/.

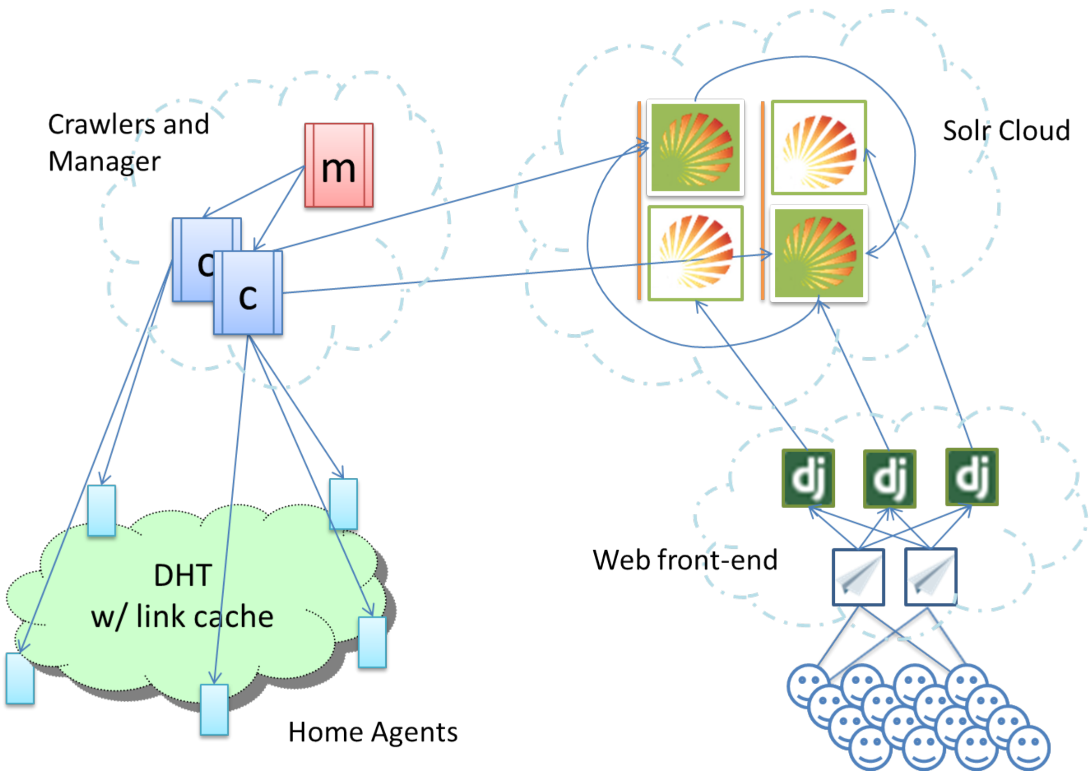


Figure 3.1: The pWeb Device Indexer architecture

Chapter 4

Conclusion

This report summarized the work done for the author's CS854 Cloud Computing and Management Research and Implementation Project. This work is also being contributed to the pWeb project, which is an effort to create a hybrid peer-to-peer and cloud-based platform for building next-generation scalable web applications. An overview of the pWeb system and its three tier architecture was provided. The architecture, design and implementation of the DNS Gateway that allows the domain name system to be used as an interface to the pWeb device naming system was discussed. The DNS Gateway allows traditional and pWeb applications to use the domain name system as an interface to the pWeb device naming system, providing a seamless integration between pWeb and traditional web applications. An analysis of the performance of the DNS Gateway software showed that it can easily handle at least 7,000 queries per second using a single application thread. The Device Indexer, another system that is being designed and implemented, was discussed briefly. The Device Indexer creates a searchable database of information about devices in the pWeb system.

Appendices

Appendix A

DNS Gateway to Home Agent Protocol

The following pages contain the specification for the version of the DNS Gateway to Home Agent protocol currently in use. The protocol was designed to simplify connection handling in the implementation of the Home Agent software but suffers from a number of limitations that are discussed in Section 2.4.1. The latest version of this document is available at `https://docs.google.com/document/d/1x58zjm0SeMgDkUsnl0X4H_3bnzpv33nsJ1Vtzu11Egs/pub`.

DNS Gateway to Home Agent - Alternative Protocol
Documented by Alexander Pokluda
Last Updated March 14, 2013

All messages are sent over TCP connections. For every message sent, the sender opens a socket, writes the message to the socket, and then closes the connection.

The receiver listens for incoming connections. When a connection is established, the receiver passively reads a message and then waits for the connection to be closed by the sender.

There are two types of messages: PeerInitiateGET, and MessageGET_REPLY. Each of these two message types is preceded by an ABSMessage header.

Field Types

All integer values are transmitted in *little-endian* format. Integers may be signed (int) or unsigned (uint). The width of each integer type is denoted in bits (eg, 8, 32).

Double values are transmitted in IEEE 754 Double-Precision Floating Point Format. Each double is 8 octets long, transmitted in *little-endian* format.

Strings are transmitted as a 4 octet length field (int 32) followed by ‘length’ ASCII characters. Strings are not null terminated.

Unused Fields

Many fields are unused and must consist entirely of zero bits.

ABSMessage

The following fields are sent in order. The use and/or value of each field is explained for PeerInitiateGET (aka GET) and MessageGET_REPLY (aka REPLY) messages from the perspective of the DNS Gateway. GET messages are sent from the DNS Gateway to the Home Agent and REPLY messages are sent from the Home Agent to the DNS Gateway.

- Message Type (uint 8)
    - GET: The value of this field must be 5 for PeerInitiateGET
    - REPLY: The value of this field must be 43 for MessageGET_REPLY.
- Sequence Number (int 32)
    - GET: The DNS Gateway copies the DNS serial number received from DNS clients to GET messages sent to the home agent. (A more robust approach could be implemented if necessary by the DNS Gateway by hashing the DNS Client’s serial number with connection specific information such as source/destination IP addresses and port numbers.)
    - REPLY: Ignored.

- Destination Hostname (string)
    - GET: The hostname of the home agent that the DNS Gateway is sending the request to.
    - REPLY: Ignored.
- Destination Port (int 32)
    - GET: The port of the home agent that the DNS Gateway is sending this message to.
    - REPLY: Ignored.
- Source Hostname (string)
    - GET: The hostname of the DNS Gateway (replies from the home agent should be sent back to this hostname).
    - REPLY: Ignored.
- Source Port (int 32)
    - GET: The port number that DNS Gateway is listening on for replies sent over a separate TCP connection.
    - REPLY: Ignored

The following 7 fields are unused/ignored in both GET and REPLY and must be set to zero in messages sent from the DNS Gateway to the Home Agent:

- Overlay Hops (uint 8)
    - GET: Unused
    - REPLY: Ignored
- Overlay TTL (uint 8)
    - GET: Unused
    - REPLY: Ignored
- IP Hops (int 32)
    - GET: Unused
    - REPLY: Ignored
- Latency (double)
    - GET: Unused
    - REPLY: Ignored
- Destination Overlay ID (int 32)
    - GET: Unused
    - REPLY: Ignored
- Destination Prefix Length (int 32)
    - GET: Unused
    - REPLY: Ignored
- Destination Max Length (int 32)
    - GET: Unused
    - REPLY: Ignored
- Source Overlay ID (int 32)
    - GET: Unused
    - REPLY: Ignored

- Source Prefix Length (int 32)
    - GET: Unused
    - REPLY: Ignored
- Source Max Length (int 32)
    - GET: Unused
    - REPLY: Ignored

PeerInitiateGET
Messages of this type are sent from a DNS Gateway to a pWeb Home Agent in order to request that in resolve a device name to IP address. A PeerInitiateGET message consists of the ABSMessage Header and the following fields:

- Device Name (string)
    - The name of the device that the home agent is being asked to find the IP address for.

MessageGET_REPLY
Messages of this type are send from a Home Agent to a DNS Gateway in order to provide a response to a PeerInitiateGET message. A MessageGET_REPLY message consists of the ABSMessage Header and the following fields in order:

- Resolution Status (int 32)
    - Ignored
- Resolution Hops (int 32)
    - Ignored
- Resolution IP Hops (int 32)
    - Ignored
- Resolution Latency (double)
    - Ignored
- Origin Sequence Number (int 32)
    - Used as DNS sequence number
- Target Overlay ID (int 32)
    - Ignored
- Target Prefix Length (int 32)
    - Ignored
- Target Max Length (int 32)
    - Ignored
- Destination Hostname (string)
    - This field contains the IPv4 address for the device as a string in dotted-decimal notation if the lookup was successful.
- Destination Port (int 32)
    - **Ignored.** This field presumably contains the number of a port that can be used to access some sort of service on a device, but there is no standard mechanism in the DNS protocol to return this information to the DNS client.
- Device Name (string)

- Ignored.

Appendix B

Proposed DNS Gateway to Home Agent Protocol

The following pages contain the specification for a proposed DNS Gateway to Home Agent protocol. The latest version of this document is available at https://docs.google.com/document/d/1I_n1iqBrHMFi3B6YKjLIg8wX-8VRuliO8j1WNA2r4tw/pub.

DNS Gateway to Home Agent Protocol Version 1.0
Specification by Alexander Pokluda
February 28, 2013

Overview

The pWeb DNS Gateway connects to one or more home agents using TCP connections. In this document, the DNS Gateway may be referred to as the *client* and the Home Agent may be referred to as the *server.*

Unless specified otherwise, all integer values are transmitted as unsigned integers in *network byte order* *(big endian). IP address values are also transmitted in network byte order.*

This is a preliminary version of the protocol. It may evolve and change over time.

Connection Establishment

The DNS Gateway will be configured to connect to one or more home agents. When the DNS Gateway starts up, it will attempt to establish a TCP connection with each home agent at the hostname and port specified in its configuration. This TCP connection will be used for all name lookup requests sent from the DNS Gateway to the Home Agent. If the the connection is closed unexpectedly, the DNS Gateway will attempt to reestablish the connection.

Handshake

The handshake consists of the client sending a version number to the server as a two octet integer. In this version of the protocol, the version number is 1.

After receiving the version number, the server should verify that it supports the version of the protocol specified and then send the same version number back to the client. The connection is now fully established and the client can begin sending queries to the server.

Messages

Each message starts with a two octet integer that identifies the type of the message (message ID). Currently two message types are defined:

- QUERY = 1
  - A request to resolve a device name to an IP address sent from the client to the server. This message is called QUERY.
- REPLY = 2
  - A response sent from the server containing a status code and optionally an IP address. This message is called REPLY.

QUERY Message

A query message starts with a two octet serial number. The serial number is used to match REPLY messages with QUERY messages. Reply messages can be sent in any order--they do not need to be sent in order that the QUERY messages were received. The client must ensure that all unanswered queries have unique serial numbers.

After the serial number is a two octet integer specifying the length (in octets) of the device name being queried, followed by the device name encoded in ASCII format.

REPLY Message
A reply message starts with the same two octet serial number sent with the corresponding query. Next is one octet status code. Currently two status codes are defined:

- SUCCESS = 0
  - The device name was resolved to an IP address successfully and an IP address follows the status code (see below).
- GENERAL FAILURE = 1
  - The Home Agent was unable to resolve the device name to an IP address.

A status code of 0 must be followed by an IP address. An IP address is sent as a one octet version number (4 for IPv4 and 6 for IPv6) followed by either 4 octets for an IPv4 address or 16 octets for an IPv6 address.

Miscellaneous
When the connection between the DNS Gateway and Home Agent is terminated, all outstanding queries/requests must be cancelled.

The server must respond to all queries within 30 seconds. If it is unable to resolve a name within 30 seconds, then it must send a reply indicating failure.

Examples
The following are examples of messages that could be sent from the client to the server and server to client in a session. The response to the query with serial number 8547 may be sent before the response to the query with serial number 24513.

```
All integers represented in base 10.
One box (marked with "+---+") indicates one octet.

CLIENT -> SERVER

 Msg ID  Ser #   Len     Device Name
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|     1 | 24513 |    14 | n   e   x   u   s   .   a   l   i   c   e   .   u   w |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

 Msg ID  Ser #   Len     Device Name
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|     1 |  8547 |    16 | l   a   p   t   o   p   .   s   h   i   h   a   b   .   u   w |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
SERVER -> CLIENT

 Msg ID  Ser #  Stts IPv IP Address
+---+---+---+---+---+---+---+---+---+---+
|     2 |  8547 | 0 | 4 |    2170661478 |
+---+---+---+---+---+---+---+---+---+---+

 Msg ID  Ser #   Stts
+---+---+---+---+---+
|     1 | 24513 | 1 |
+---+---+---+---+---+
```

Appendix C

Report Produced by ResPerf

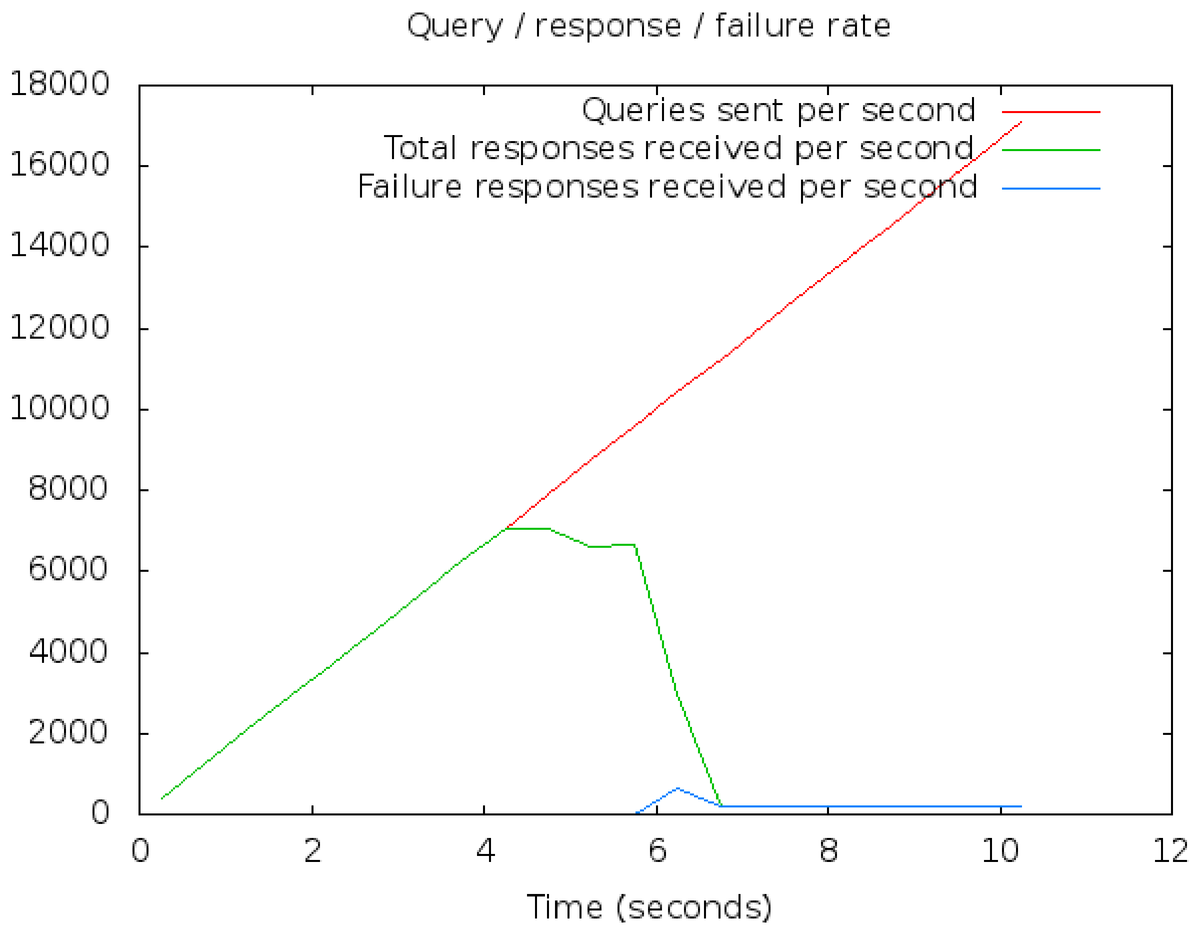
The following pages contain a report produced the ResPerf DNS performance testing tool. The test was run using the mock Home Agent, DNS Gateway and performance testing tool on the same machine.

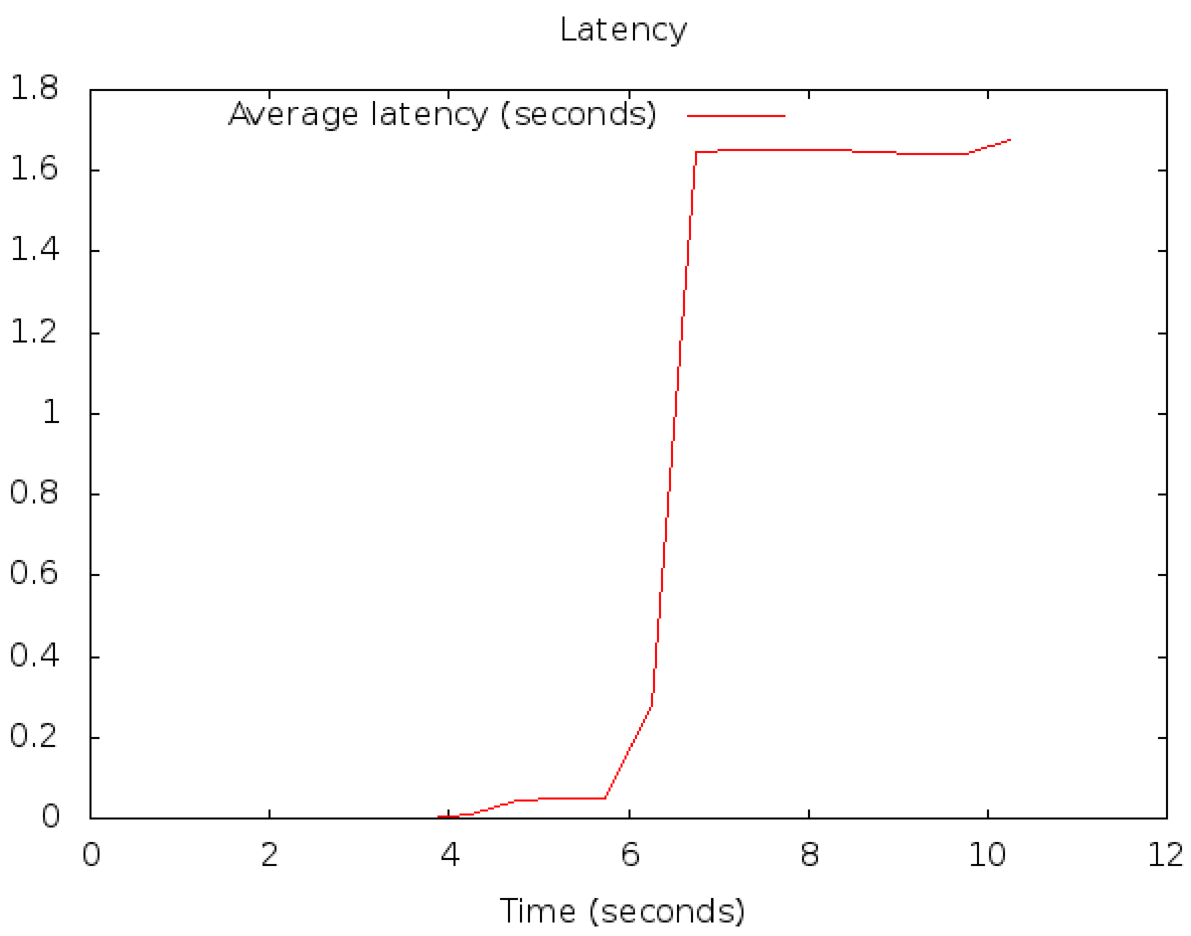
Resperf report 20130414-2054

Resperf output

```
DNS Resolution Performance Testing Tool
Nominum Version 2.0.0.0

[Status] Command line: resperf -P 20130414-2054.gnuplot -s 127.0.0.1 -p 5354 -d 1000000-dht
[Status] Sending
[Status] Reached 65536 outstanding queries
[Status] Waiting for more responses
[Status] Testing complete

Statistics:

  Queries sent:         94571
  Queries completed:    94571
  Queries lost:         0
  Run time (s):         100.000000
  Maximum throughput:   7084.000000 qps
  Lost at that point:   10.53%
```

Plots




Latency
Average latency (seconds)
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
0
2
4
6
8
10
12
Time (seconds)

References

[1] Reaz Ahmed, Faizul Bari, Raouf Boutaba, and Rakib Haque. Liverable-1.2: Architecture, design and implementation. Technical report, University of Waterloo, 2011. Project Report for Orange Labs.

[2] Reaz Ahmed and Raouf Boutaba. pWeb: P2P web hosting.

[3] Reaz Ahmed and Raouf Boutaba. Plexus: A scalable peer-to-peer protocol enabling efficient subset search. In *IEEE/ACM Transactions on Networking*, volume 17, pages 130–143. IEEE, 2009.

[4] R. Braden. Requirements for Internet Hosts - Application and Support. RFC 1123 (INTERNET STANDARD), October 1989. Updated by RFCs 1349, 2181, 5321, 5966.

[5] Ian Clarke, Oskar Sandberg, Brandon Wiley, and TheodoreW. Hong. Freenet: A distributed anonymous information storage and retrieval system. In Hannes Federrath, editor, *Designing Privacy Enhancing Technologies*, volume 2009 of *Lecture Notes in Computer Science*, pages 46–66. Springer Berlin Heidelberg, 2001.

[6] R. Elz and R. Bush. Clarifications to the DNS Specification. RFC 2181 (Proposed Standard), July 1997. Updated by RFCs 4035, 2535, 4343, 4033, 4034, 5452.

[7] Log for C++ Project. Log for c++ project. http://log4cpp.sourceforge.net/, November 2011.

[8] XMPP Standards Foundation. Xmpp.

[9] Michael J. Freedman, Eric Freudenthal, and David Mazières. Democratizing content publication with coral. In *Proceedings of the 1st conference on Symposium on*

Networked Systems Design and Implementation - Volume 1, NSDI'04, pages 18–18, Berkeley, CA, USA, 2004. USENIX Association.

[10] Jabber. http://www.jabber.org/.

[11] Dan Kegel. The C10K problem. http://www.kegel.com/c10k.html, July 2011.

[12] Christopher Kohlhoff. Networking library proposal for tr2 (revision 1). Mailing, JTC1/SC22/WG21 - The C++ Standards Comittee, March 2007. Doc No: N2175=07-0035.

[13] Christopher Kohlhoff. Boost.asio - 1.53. http://www.boost.org/doc/libs/1_53_0/doc/html/boost_asio.html, 2012.

[14] Christopher Kohlhoff. Boost.asio: Platform-specific implementation notes. http://www.boost.org/doc/libs/1_53_0/doc/html/boost_asio/overview/implementation.html, 2012.

[15] P.V. Mockapetris. Domain names - implementation and specification. RFC 1035 (INTERNET STANDARD), November 1987. Updated by RFCs 1101, 1183, 1348, 1876, 1982, 1995, 1996, 2065, 2136, 2181, 2137, 2308, 2535, 2845, 3425, 3658, 4033, 4034, 4035, 4343, 5936, 5966, 6604.

[16] C. Perkins. IP Mobility Support for IPv4, Revised. RFC 5944 (Proposed Standard), November 2010.

[17] C. Perkins, D. Johnson, and J. Arkko. Mobility Support in IPv6. RFC 6275 (Proposed Standard), July 2011.

[18] Vladimir Prus. Boost.program_options. http://www.boost.org/doc/libs/1_53_0/doc/html/program_options.html, 2004.

[19] Robert Ramey. Boost.serialization. http://www.boost.org/doc/libs/1_53_0/libs/serialization/doc/index.html, 2004.

[20] Credit Suisse. China smpartphone sector: Global handset forecast lifted by emerging market demand. https://plus.credit-suisse.com/r/TpQjQj, January 2013. Credit Suisse Plus Article.

[21] P. Vixie, S. Thomson, Y. Rekhter, and J. Bound. Dynamic Updates in the Domain Name System (DNS UPDATE). RFC 2136 (Proposed Standard), April 1997. Updated by RFCs 3007, 4035, 4033, 4034.

[22] Felix von Leitner. Scalable network programming or: The quest for a good web server (that survives slashdot). http://bulk.fefe.de/scalable-networking.pdf, October 2003.

[23] Erik Wilde and Cesare Pautasso. *REST: From Resarch to Practice*. Springer, 2011.

[24] Yi Wu, J. Tuononen, and M. Latvala. An analytical model for dns performance with TTL value 0 in mobile internet. In *TENCON 2006. 2006 IEEE Region 10 Conference*, pages 1–4, 2006.

[25] Yi Wu, J. Tuononen, and M. Latvala. Performance analysis of dns with TTL value 0 as location repository in mobile internet. In *Wireless Communications and Networking Conference, 2007.WCNC 2007. IEEE*, pages 3250–3255, 2007.