Client Server Application Programming

Week 2: Internet Addressing, The Domain Name System, Internet Addressing with Java, etc,

By Elubu Joseph - MSc.IS

Lecturer

Department of Information Technology

Kumi University

Email: josebulinda@gmail.com

jose@kumiuniversity.ac.ug

Agenda

1. Internet Addressing -Local Area Network Addresses, - Internet Protocol Addresses,
2. Internet Addressing with Java
3. The Domain Name System

Internet Addressing

Internet is a network of networks.

Unlike local area networks (LANs), the Internet is a vast collection of machines and devices spread out across the nation and the world.

When there are hundreds of millions (and eventually many billions) of computers and devices attached to a network, **the need to identify and locate a specific one is obviously important**.

Therefore **internet addressing is an** act of identifying computers and other related devices connected to the internet.

Indeed, one of the most fundamental concepts in network programming is that of the network address. Without it, there would be no way of identifying the sender of a data packet or where the packet must be sent, David R. and Michael R(2002).

Local Area Network Addresses

LAN is a connection of computers and other devices within a radius of 100 Meters.

Devices connected to a LAN have their own unique physical or hardware address. This assists other machines on the network in delivering data packets to the correct location.

The **address is useful only in the context of a LAN**, however—a machine can't be located on the Internet by using its physical address, which does not indicate the location of the machine. Indeed, machines often move from location to location, in the case of laptop or palmtop computers.

David R. and Michael R(2002).

Local Area Network Addresses+

Java network programmers **do not need to be concerned with the details** of how data is routed within a LAN. Indeed, Java does not provide access to the lower-level data link protocols used by LANs. Supporting the wide range of protocols available would be a mammoth task.

Since each type of protocol uses a different type of address and has different characteristics, different code would need to be written for each and every type of network. Instead, Java provides support for TCP/IP, which can be thought of as the glue that binds networks together.

No matter what type of LAN is used—if one is used at all—software can be written for it in Java providing it supports TCP/IP. Individual machines have unique addresses for the transmission and receipt of IP datagrams, in addition to their normal network or physical addresses. David R. and Michael R(2002).

Internet Protocol Addresses

Devices having a direct Internet connection are allocated a unique identifier known as an IP address. IP addresses may be two types:- Static IP Addresses or Dynamic IP Addresses.

Static IP addresses - are bound permanently to a certain machine or device, or

Dynamic IP **addresses -** are leased to a particular machine or device for a certain period, for example in the case of an Internet service provider [ISP] that offers a pool of modems for dial-up connections.

Internet Protocol Addresses +

Dynamically assigned IP addresses are typically used when many devices require Internet access for limited periods of time. Thus addresses can be allocated from a pool of remaining addresses on a case-by-case basis. However, an IP address can be bound to a single machine only; it cannot be shared concurrently.

This address is used by the Internet Protocol to route IP datagrams to the correct location. Without an address, a machine cannot be contacted; hence, all machines must have a unique IP address

Internet Protocol Addresses + +

NOTE.

The only exception to this is the case of an intranet. Within the environment of an intranet, a shared range of IP addresses can be used, specially set aside for intranet use. Anyone may bind a machine to one of these addresses; as they are not exposed to the public Internet there is no conflict with an external machine.

however, One must be careful, not to use the same IP address for multiple machines or devices within an intranet.

Structure of the IP Address

Under the Internet Protocol version 4, referred to as IPv4, the IP address is a 32-bit number made up of four octets (a series of 8 bits).

While computers read an IP address as a sequence of bits, people see them in dotted decimal format (for example, 127.0.0.1). There are five classes of IP addresses (A through E), and each class is allocated an address range, as shown

Range of IP Addresses by Class

Type	Address Range
Class B	128.0.0.0 – 191.255.255.255
Class C	192.0.0.0 – 223.255.255.255
Class D	224.0.0.0 – 239.255.255.255
Class E	240.0.0.0 – 247.255.255.255

Obtaining an IP Address

The central body responsible for allocating blocks of IP addresses is the Internet Corporation for Assigned Names and Numbers (ICANN), building on the work of an earlier organization, the Internet Assigned Numbers Authority (IANA). A person setting up a private network would be allocated either a Class A, B, or C address, and could then assign host IP addresses to the machines on that particular network.

A discussion of the process of setting up a private network is beyond the scope of this course, of course, but you can find further information on ICANN and IANA organization Websites useful, at http://www.icann.org/ and http://www.iana.org

Obtaining an IP Address+

The most common way to obtain an IP address is to have one assigned to you by a network administrator, ISP, or other network service.

When you establish a dial-up connection, you will usually be assigned an IP address. This address is normally dynamically assigned from a pool of available addresses, and when you reconnect you'll usually get a different address.

In an intranet, a network administrator may assign a static address to your machine, or you may have a dynamically assigned address allocated by a Dynamic Host Control Protocol (DHCP) server.

DHCP provides addresses on demand; if machines are going online and offline frequently, a smaller pool of addresses can be used.

Special IP Addresses

Are IP addresses reserved for specific common usage.

Programmers should be aware of some special IP addresses as well. The first, and most important from a network programming perspective, is known as the **loopback** or **localhost address**.

When writing and debugging network software, programmers often want to connect to the local machine for testing purposes.

Regardless of whether a connection to the Internet exists via a dial-up service or work is being done offline, the local machine may be accessed using the loopback address; this address is **127.0.0.1**

Special IP Addresses+

NOTE.

Your machine may be known by many IP addresses in addition to the loopback address. For example, you may be assigned an IP address when you dial your ISP, or your network administrator may have assigned an address for your intranet.

Programmers experimenting with writing networking software may find that it makes sense from a financial perspective to disconnect from their ISP and use the loopback address whenever possible.

Special IP Addresses++

Another set of useful IP addresses are those reserved for private networking. In an intranet environment, it may be desirable to configure all machines with a unique IP address, without having them exposed to the public Internet.

The Internet Assigned Number Authority (IANA) has reserved three sets of addresses for use within a local intranet environment, as described in **RFC 1918**. e.g. 10.0.0.0 to 10.255.255.255, 172.16.0.0 to 172.31.255.255 and 192.168.0.0.

If you plan on setting up a LAN of your own, you can pluck addresses from these ranges without worrying about a collision conflict with a host on the Internet.

On the Internet, routers will not forward data for these addresses, so they can be safely used locally. It must be noted, however, that this is the only time when IP addresses can be safely picked, unless you are allocated a block of addresses

Beyond IP Addresses
The Domain Name System

The Domain Name System

While IP addresses comprise an efficient system for network administrators, most people find memorizing them to be an impossible task.

People generally find words much easier to recall than the dotted decimal format of an IP address. It is easier to remember a name such as **Monowaa** or **Google** or **KUMU** than a set of numbers designating such a dotted decimal IP address.

The Domain Name System

The domain name system (DNS) is a naming system that makes the Internet user-friendly, by associating a textual name with an IP address.

Any entity, be it commercial, government, or private, can apply for a domain name, which can be used by people to locate that entity on the Internet.

Simple text names, as opposed to arbitrary numbers, are used for identification. In addition, organizations can allocate their own hostnames, such as www.kumiuniversity.ac.ug and www.monowaa.com, without having to register every host with an outside body. Once you have a domain name, you can control how it is used.

What is a domain name?

Domain name is a textual name associated with an IP address offered by a DNS.

E.g.

monowaa.com may be associated 10.212.2.123

Or

kumiuniversity.ac.ug may be associated 66.340.1.269

Therefore, instead of the user to remember decimal digits, he/she will use the easy name.

Many registries perform the service of registering domain names, at variable prices. The original domain name registrar, Network Solutions, is located at http://www.networksolutions.com/.

How Domain Name System Work

Given the vast number of machines connected to the Internet, the number of domain-name-to-IP address mappings is too great for any one system to handle. Even if there were a large enough system to store all of these mappings, it would be quickly overloaded by requests. Furthermore, in the event of a system breakdown, isolating the problem to a fraction of the Internet would be preferable to having the entire system grind to a shuddering halt.

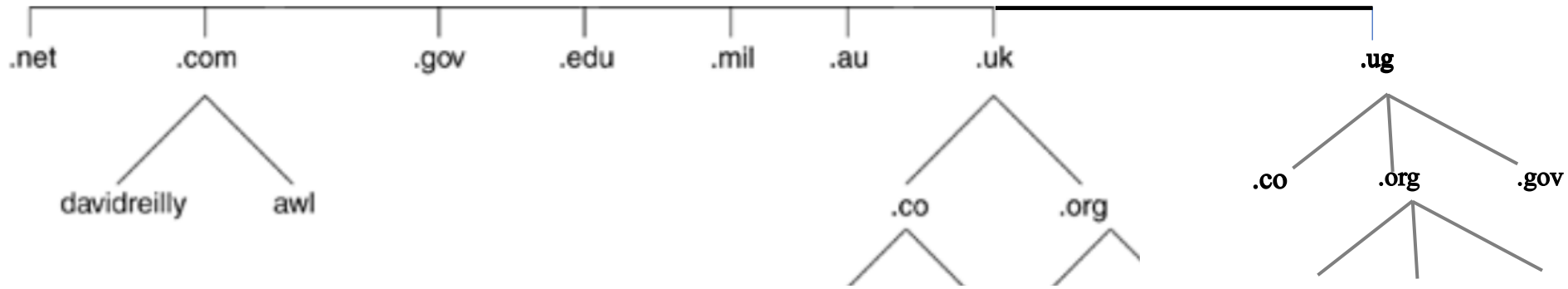
Ironically, however, when the Internet was originally conceived, host-to-IP-address mappings were stored in a single file, called hosts.txt, which was downloaded and mirrored across the Internet. This file still exists in many operating systems and can be used to override DNS mappings or cover up the ones that are missing or unresolvable by the local DNS servers. In some cases, they can be used to cover up not having a local DNS server on an intranet.

How Domain Name System Work++

For that reason, the DNS (outlined in RFC 1034/1035) is a more sophisticated and robust system. It can be thought of as a distributed database, in which responsibility for accepting new registrations, and returning the addresses of existing registrations, is spread out across many different hosts.

Different categories, such as commercial and educational, are handled by different registry servers. Further more, international registries handle their own mappings (called cou try-code top-level domains), and can be subdivided into further categories. This forms a hierarchical structure, a small subset of which is shown below. Since the range of address categories changes rapidly, only a small number are shown.

Subset of the DNS hierarchy



How Domain Name System Work+++

This hierarchical structure is broken up by the type of address (either .net, .com, .gov, .edu, .mil, or one of the newer addresses such as .info or .biz) or by the country (.au, .uk, .ug, .kor among many others).

For example, to access the site www.google.com, a request to resolve this name would be sent to the .com DNS server.

Some countries have their own way of organizing DNS records for example, as seen above,

the Uganda (.ug) uses .co rather than .com

Domain Name Resolution

is **the process by which internet users receive the address of the domain they were looking for. Or one can also** Domain Name Resolution is **the task of converting domain names to their corresponding IP address**

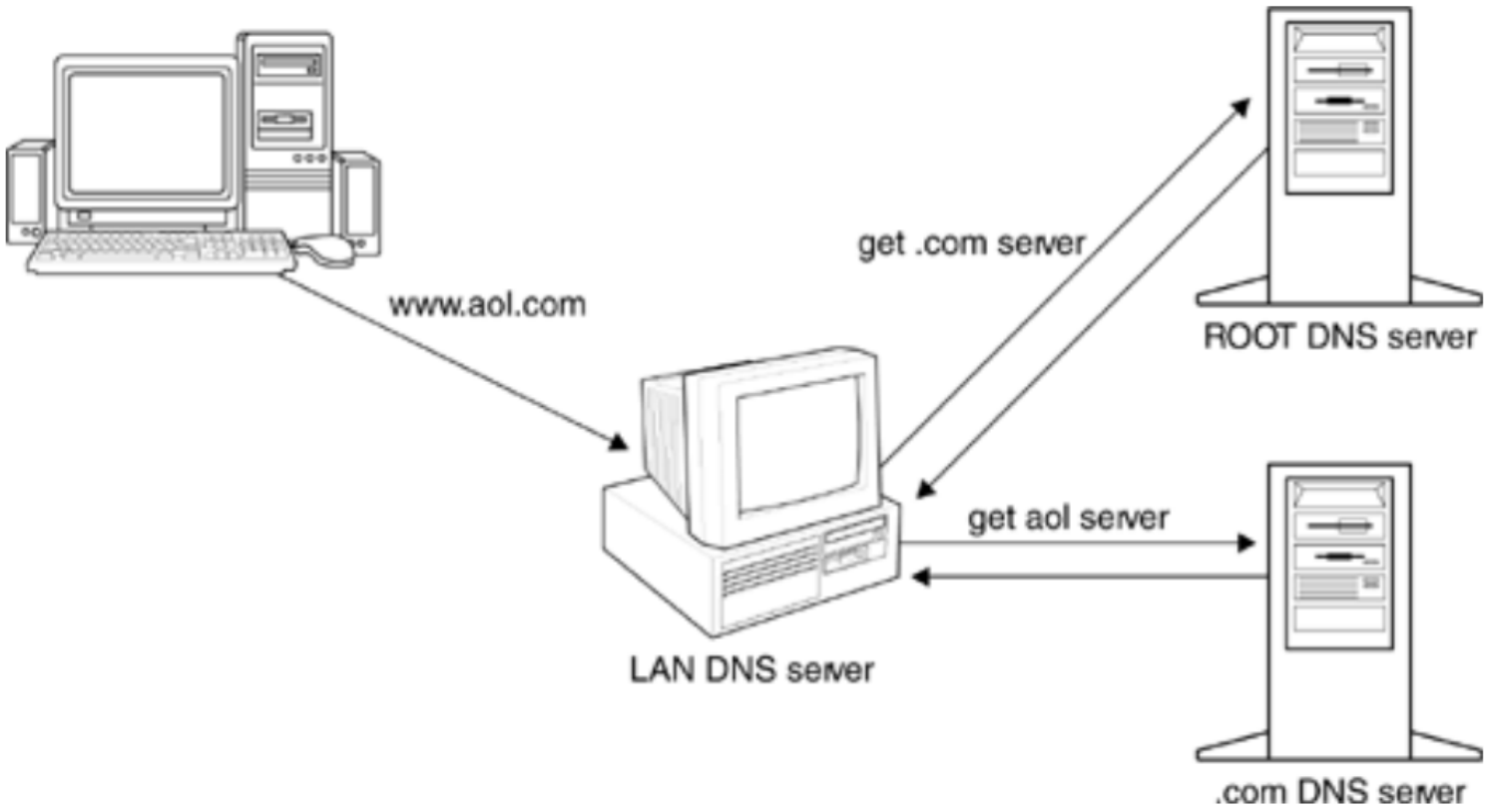
When software applications need to look up a hostname (such as ftp.davidreilly.com), they don't contact the .com registry directly. Your network administrator, or ISP, configures your system to access the application's DNS server, which handles the lookup process. Often, however, the DNS server doesn't even need to send out a query, as the same sites will be requested (either by multiple users or a single one).

This works much the same as a Web browser caching pages, and prevents excessive overload of the network.

When a request for a hostname such as www.aol.com or www.monowaa.com is made, the operating system of the client computer contacts the local DNS server on the LAN because it irresponsible for locating the domain server aol.com or monowaa.com, and interrogating it to find the IP address of the hostname (www.aol.com or www.monowaa.com).

To do this, it must query the "root" level domain server, which refers it to the .com server. Finally, the IP address is returned to the user .

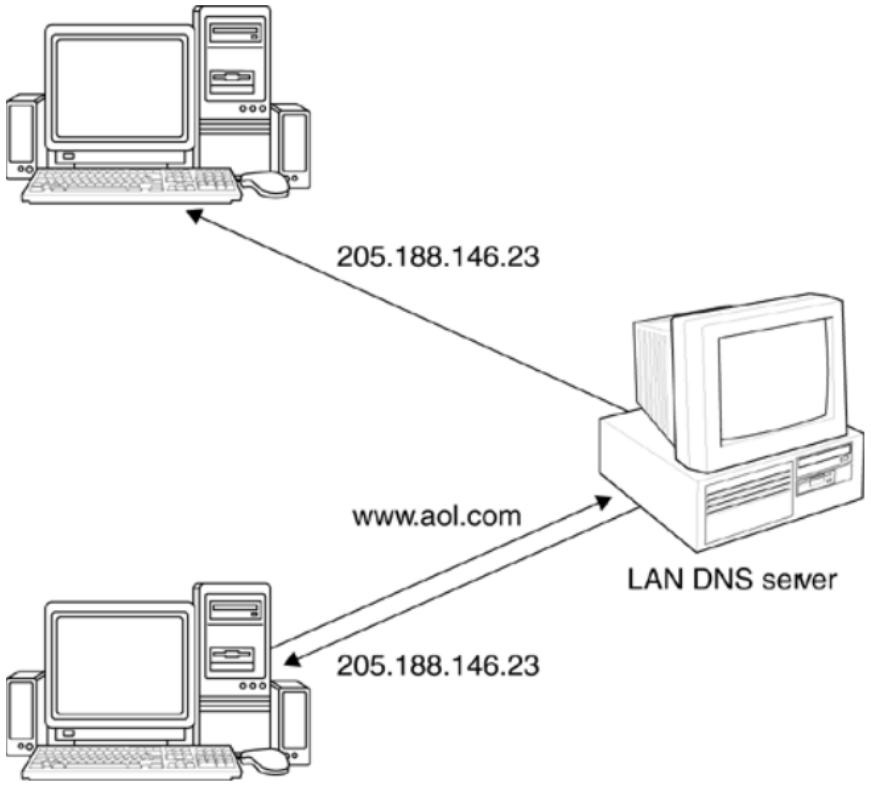
Domain Name Resolution steps



Request for www.aol.com reaches LAN DNS server and is resolved

Domain Name Resolution steps++

As can be seen from the example in the figure below, DNS requests can be cached. The LAN DNS server can maintain a cache of recently requested addresses, to prevent overloading of the rootlevel domains. Furthermore, this improves network performance, as the delay between requesting a domain name and receiving a response is diminished.



DNS server returns IP address, and caches query from another machine.

DNS Resolution steps Summarized

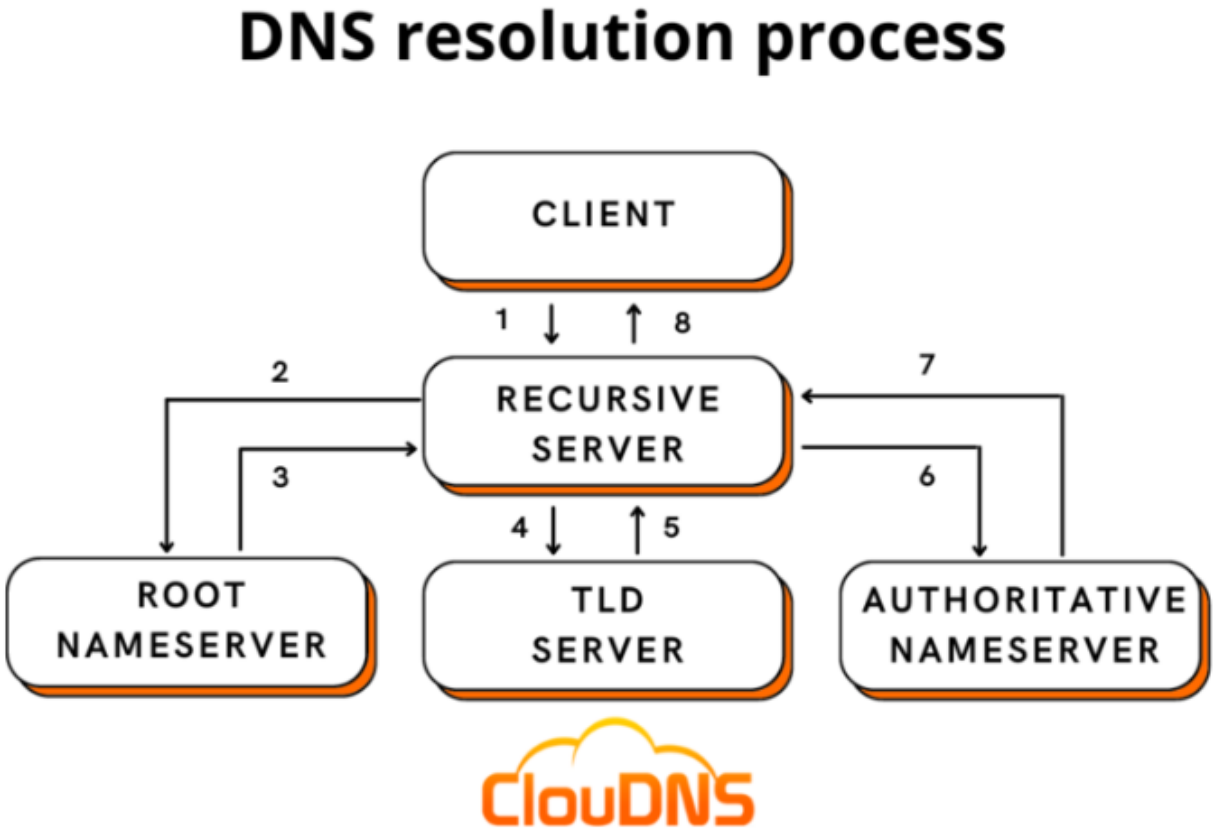
1. A user is typing a domain name like cloudns.net into their browser. The user needs an A or AAAA DNS record to resolve the domain name.
2. If your device's cache has the IP address of *cloudns.net*, the domain name resolution will finish here, and the user will be able to open the website. But, if it does not, there will be more steps. The devices keep DNS records for visited sites, depending on the TTL (Time to Live) values of those DNS records.
3. If your computer doesn't have the needed IP address, it will search for the answer by performing a DNS resolution query. The next destination on the way will be the recursive DNS servers of the internet services provider. They also keep a cache with DNS records of domain names that users have accessed. If the desired site's DNS records are still there, the user will get an answer to its query and access the site. If not, there will be a series of interactive DNS queries to find the answer.

DNS Resolution steps Summarized

4. If the domain name resolution didn't finish with the previous step, the recursive nameserver would search for the answer. The next step will be to ask the Root server, which is indicated with a "." sign after the TLD (top-level domain). The Root server does not have an answer about the exact domain name, but it will provide one for the part it is responsible for – it will indicate all the nameservers for the TLD that we ask The TLD DNS servers will have the answer of which exactly are the authoritative nameservers for the domain you are searching. The TLD servers of .net will have that information for all of the domain names that finish with .net. They will return that answer so the query can continue.
5. Now that we know where the authoritative nameserver of the domain name we want is, we can ask and get the A and AAAA records to understand the site's IP address.
6. The Authoritative nameservers of the domain name will provide the DNS records, the DNS resolution will be made. The recursive nameserver of our ISP and our device will both save the DNS records that we obtained based on their TTL values. That way, if we soon want to visit the site again, we will save time and access the site faster.

DNS Resolution steps Summarized

7. Visit the site. Now with the DNS record already obtained, the user can access the site.



Internet Addressing with Java

By now it should be clear that a host on the Internet can be represented either in dotted decimal format as an IP address, or as a hostname such as ftp.kumu.com. Under Java, such addresses are represented by the java.net.InetAddress class.

This class can fill a variety of tasks, from resoling an IP address to looking up the hostname. In the next section, this important class is examined in detail.

The java.net.InetAddress Class

The InetAddress class is used to represent IP addresses within a Java networking application. Unlike most other classes, there are no public constructors for that of InetAddress. Instead, there are two static methods that return InetAddress instances.

Those and the other major methods of this class are covered in the list below; all are public unless otherwise noted.

1. boolean equals(Object obj)— compares two IP addresses, and returns "true" if there is a match. Note, however, that some machines can be known by multiple IP addresses, so this is not an absolute test for equality; what is tested is only that the two addresses are equal, not that they are the same machine.
2. byte[] getAddress()— returns the IP address in byte format (as opposed to dotted decimal notation). The bytes are returned in network byte order, with the highest byte as bytearray[0].

The java.net.InetAddress Class+

3. static InetAddress[] getAllByName (String hostname) throws java.net.UnknownHostException,java.lang.SecurityException— returns, as a static method, an array of InetAddress instances representing the specified hostname. While most machines will have a single IP address, there are some situations in which one hostname can be mapped to many machines and/or a hostname can map to many addresses on one machine (virtual addresses).

If the host cannot be resolved, or if resolving the host conflicts with the security manager, an exception will be thrown.

The java.net.InetAddress Class+

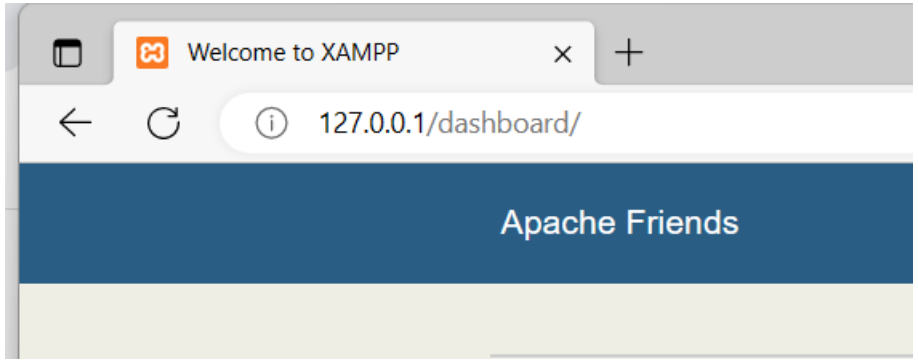
4. static InetAddress getByName (String hostname) throws java.net.UnknownHostException, java.lang.SecurityException— returns an InetAddress instance representing the specified hostname, which may be represented either as a text hostname (e.g., monowaa.com) or as an IP address in dotted decimal format. If the host cannot be resolved, or resolving the host conflicts with the security manager, an exception will be thrown.
5. String getHostAddress()— returns the IP address of the InetAddress in dotted decimal format.
6. static InetAddress getLocalHost() throws java.net.UnknownHostException, java.lang.SecurityException— returns, as a static method, the IP address of the localhost machine. If the IP address cannot be determined, or doing so conflicts with the security manager, then an exception will be thrown.

The java.net.InetAddress Class+

7. boolean isMulticastAddress()— returns "true" if the InetAddress is a multicast address, also known as a Class D address.
8. String toString()— returns a string representation of the InetAddress. Readers are advised to use the getHostName() and getHostAddress() methods, to control which type of data is being requested.

Using InetAddress to Determine Localhost Address

The first, and most simple, example of InetAddress is to find out the IP address of the current machine. If a direct connection to the Internet exists, a meaningful result will be obtained, but dial-up users and those without any Internet connection (such as in an intranet environment) may get the loopback address of 127.0.0.1.



E.g. When running XAMPP application. One can access the dashboard by either entering hostname in this case localhost or 127.0.0.1 ip address

The program given below shows how possible it is to determine the address.

Using InetAddress to Determine Localhost Address

```
import java.net.*;
public class CS {
    public static void main(String[] args) {
        System.out.println ( "Looking up local host“ );
        try{
        // Get the local host
        InetAddress localAddress =InetAddress.getLocalHost();
        System.out.println ("IP address : " + localAddress.getHostAddress() );
        }
        catch (UnknownHostException uhe)
        {
        System.out.println ("Error - unable to resolve localhost");
        }
    }
}
```

Using InetAddress to Determine Localhost Address++

```
1   package cs;
2   import java.net.*;
3   public class CS {
4       public static void main(String[] args) {
5           System.out.println ("Looking up local host");
6           try{
7           // Get the local host
8           InetAddress localAddress =InetAddress.getLocalHost();
9           System.out.println ("IP address : " +
10          localAddress.getHostAddress() );
11          }
12          catch (UnknownHostException uhe)
13          {
14          System.out.println ("Error - unable to resolvelocalhost");
15          }
16      }
17  }
```

```
run:
Looking up local host
IP address : 127.0.0.1
BUILD SUCCESSFUL (total time: 0 seconds)
```

Using InetAddress to Determine Localhost Address+++
How the above CS Application works

The CS application starts by prompting the user that an IP address lookup will be performed (this is important if there is any delay in determining the IP address).

The networking operation must be enclosed within a **try/catch block**, because it is possible that no IP address will be found and an exception will be thrown. Using the static method InetAddress.getLocalHost(), we obtain an object representing an IP address.

To display the address in dotted decimal notation, the InetAddress.getHostAddress() method is used

Using InetAddress to Find Out About Other Addresses using Java program: NetworkResolverPro

The previous example familiarized you with the InetAddress class. Below is a more complex example, which resolves hostnames to IP addresses and then attempts to perform a reverse lookup of the IP address.

This Java program named **NetworkResolverPro** is a network resolver that takes a hostname as an argument from the command line or prompts the user to enter a hostname if no argument is provided. The program then resolves the hostname to an IP address and a hostname.

The **main** method begins by declaring a **String** variable named **hostname**. If one argument is provided in the command line, the program assigns that argument to **hostname**. Otherwise, it prompts the user to enter a hostname using the **Scanner** class and assigns the input to **hostname**.

Using InetAddress to Find Out About Other Addresses using Java program: NetworkResolverPro+

The program then prints a message indicating which hostname it is resolving.

Next, the program attempts to resolve the hostname using the **getByName()** method of the **InetAddress** class. This method takes the hostname as an argument and returns an instance of the **InetAddress** class that represents the IP address of the hostname.

If the resolution is successful, the program prints the IP address and the hostname using the **getHostAddress()** and **getHostName()** methods of the **InetAddress** class, respectively.

If there is an error during hostname resolution (e.g., the hostname is invalid or the DNS server is not responding), the program catches the **UnknownHostException** exception and prints an error message.

Using InetAddress to Find Out About Other Addresses using Java program: NetworkResolverPro+

```
package cs; import java.net.*;
import java.util.Scanner;
public class NetworkResolverPro {
public static void main(String[] args) {
        String hostname;
        if (args.length == 1){
            hostname = args[0];
        } else {
            Scanner scanner = new Scanner(System.in);
            System.out.print("Enter hostname to resolve: ");
            hostname = scanner.nextLine();
        }
        System.out.println ("Resolving " + hostname);
        try{
            // Resolve host and get InetAddress
            InetAddress addr = InetAddress.getByName(hostname);
            System.out.println ("IP address : " +addr.getHostAddress() );
            System.out.println ("Hostname : " +addr.getHostName() );
        } catch (UnknownHostException uhe){
            System.out.println ("Error - unable to resolve hostname" );
        } }}
```

Using InetAddress to Find Out About Other Addresses +

How NetworkResolverPro Works

```
1  package cs; import java.net.*;
2  import java.util.Scanner;
3  public class NetworkResolverPro {
4      public static void main(String[] args) {
5          String hostname;
6          if (args.length == 1){
7              hostname = args[0];
8          } else {
9              Scanner scanner = new Scanner(System.in);
10             System.out.print("Enter hostname to resolve: ");
11             hostname = scanner.nextLine();
12         }
13         System.out.println ("Resolving " + hostname);
14         try{
15             // Resolve host and get InetAddress
16             InetAddress addr = InetAddress.getByName(hostname);
17             System.out.println ("IP address : " +addr.getHostAddress() );
18             System.out.println ("Hostname : " +addr.getHostName() );
19         } catch (UnknownHostException uhe){
20             System.out.println ("Error - unable to resolve hostname" );
21         } }}
```

Output with correct host name "**google.com**"

```
run:
Enter hostname to resolve: google.com
Resolving google.com
IP address : 216.58.223.110
Hostname : google.com
```

Output with incorrect hostname "**kumu**"

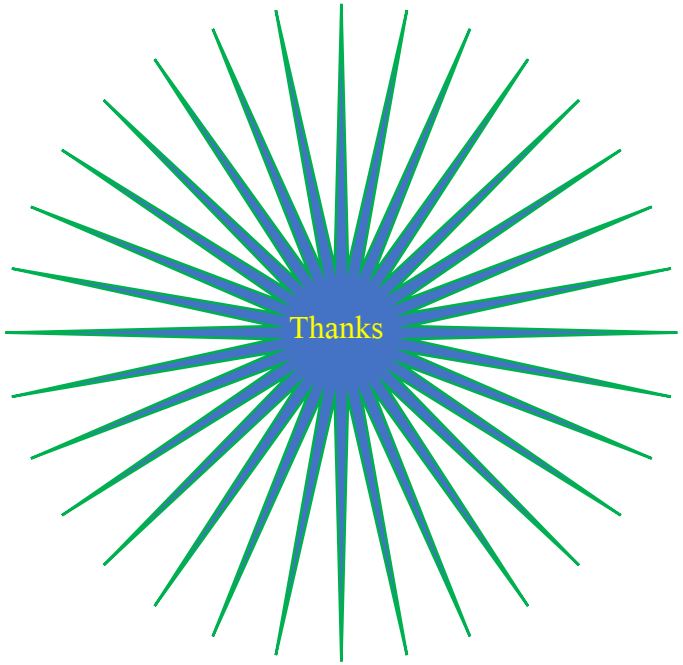
```
run:
Enter hostname to resolve: kumu
Resolving kumu
Error - unable to resolve hostname
```

Summary

Summary

1. Internet Addressing -Local Area Network Addresses, - Internet Protocol Addresses,
2. Internet Addressing with Java
3. The Domain Name System

Thank you for Listening



References

Java Network Programming and Distributed Computing, David R, Michael R (2002), Publisher: Addison Wesley, ISBN: 0-201-71037-4

PramatarovHi, M. (2022, November 18). *What is domain name resolution?* ClouDNS Blog. Retrieved March 30, 2023, from https://www.cloudns.net/blog/domain-name-resolution/

What is domain name resolution. BleepingComputer. (2004, April 9). Retrieved March 30, 2023, from https://www.bleepingcomputer.com/tutorials/what-is-domain-name-resolution/