Unit.2 Internet Addresses

Outline

2.1 The InetAddress Class

2.2 Inet4Address and Inet6Address

2.3 The NetworkInterface Class

2.4 Some Useful Programs

Internet Addresses

- IP (Internet Protocol) Addresses
  - IPv4 (4 Bytes): dotted quad format
    - www.csie.nuk.edu.tw 140.127.208.17
  - IPv6 (16 Bytes): 8 blocks of 4 hexadecimal digits separated by colons
    - www.csie.nuk.edu.tw **::ffff**:8c7f:d011
    - 2400:cb00:2048**:000**1**:0000:0000:**6ca2:c665 → 2400:cb00:2048**:**1**::**6ca2:c665
  - Mixed: last 4 bytes of the IPv6 written as an IPv4 dotted quad address
    - www.nuk.edu.tw ::ffff:140.127.208.17
    - FEDC:BA98:7654:3210:FEDC:BA98:7654:3210
      FEDC:BA98:7654:3210:FEDC:BA98:118.84.50.16
- Domain Names – Resolved by DNS Servers
  - FQDN: Fully Qualified Domain Name
    - www.csie.nuk.edu.tw.
  - One name can map to multiple IP addresses
  - One IP addresses can also have multiple names

4.1 The InetAddress Class

http://docs.oracle.com/javase/8/docs/api/java/net/InetAddress.html

- Creating new InetAddress objects
  - No public constructors; use static factory methods directly
    - Automatically connect to a DNS server to resolve a hostname
    - Throws an UnknownHostException, a subclass of IOException, if not found
- getByName(): lookup the name and the numeric address

```
InetAddress address = InetAddress.getByName("www.oreilly.com");
InetAddress address = InetAddress.getByName("208.201.239.100");
```

Example 4-1. A program that prints the address of www.oreilly.com

```
import java.net.*;

public class OReillyByName {

  public static void main (String[] args) {
    try {
      InetAddress address = InetAddress.getByName("www.oreilly.com");
      System.out.println(address);
    } catch (UnknownHostException ex) {
      System.out.println("Could not find www.oreilly.com");
    }
  }
}
```

```
% java OReillyByName
www.oreilly.com/208.201.239.36
```

- getAllByName(): lookup all the addresses of a host

```
try {
  InetAddress[] addresses = InetAddress.getAllByName("www.oreilly.com");
  for (InetAddress address : addresses) {
    System.out.println(address);
  }
} catch (UnknownHostException ex) {
  System.out.println("Could not find www.oreilly.com");
}
```

```
www.google.com/173.194.72.147
www.google.com/173.194.72.105
www.google.com/173.194.72.104
www.google.com/173.194.72.99
www.google.com/173.194.72.103
www.google.com/173.194.72.106
```

getLocalHost() and getByAddress()

- getLocalHost(): return an InetAddress object for the local host
  - Return 'localhost/127.0.0.1' if lookup failed

Example 4-2. Find the address of the local machine

```
import java.net.*;

public class MyAddress {

  public static void main (String[] args) {
    try {
      InetAddress address = InetAddress.getLocalHost();
      System.out.println(address);
    } catch (UnknownHostException ex) {
      System.out.println("Could not find this computer's address.");
    }
  }
}
```

```
% java MyAddress
titan.oit.unc.edu/152.2.22.14
```

- getByAddress(): create an InetAddress object from given address
  - Without talking to DNS

```
public static InetAddress getByAddress(byte[] addr) throws UnknownHostException
public static InetAddress getByAddress(String hostname, byte[] addr)
    throws UnknownHostException
```

  - Example

```
byte[] address = {107, 23, (byte) 216, (byte) 196};
InetAddress lessWrong = InetAddress.getByAddress(address);
InetAddress lessWrongWithname = InetAddress.getByAddress(
   "lesswrong.com", address);
```

Caching and Security Issues

- The InetAddress class caches the results of lookups
  - Java only caches unsuccessful DNS queries for 10 seconds by default
  - Times can be controlled and specified in Java security properties
    - networkaddress.cache.ttl: specifies the number of seconds a successful DNS lookup will remain in Java's cache
    - networkaddress.cache.negative.ttl: specifies the number of seconds an unsuccessful lookup will be cached

- Security Issues: a DNS lookup generates network traffic
  - Untrusted code
    - Prohibition against making network connections to hosts other than the codebase
    - An untrusted applet under the control of the default security manager will only be allowed to get the IP address of the host it came from (its codebase) and possibly the local host
  - Relaxed for trusted code
  - checkConnect(): test whether a host can be resolved

InetAddress: Create Objects and Getter Methods

static InetAddress[]	**getAllByName**(String host) Given the name of a host, returns an array of its IP addresses, based on the configured name service on the system
static InetAddress	**getByAddress**(byte[] addr) Returns an InetAddress object given the raw IP address
static InetAddress	**getByAddress**(String host, byte[] addr) Creates an InetAddress based on the provided host name and IP address
static InetAddress	**getByName**(String host) Determines the IP address of a host, given the host's name
static InetAddress	**getLocalHost**() Returns the address of the local host
static InetAddress	**getLoopbackAddress**() Returns the loopback address
byte[]	**getAddress**() Returns the raw IP address of this InetAddress object
String	**getCanonicalHostName**() Gets the fully qualified domain name for this IP address
String	**getHostAddress**() Returns the IP address string in textual presentation
String	**getHostName**() Gets the host name for this IP address

Examples – Find Hostname, IP and Version

Example 4-3. Given the address, find the hostname

```
import java.net.*;

public class ReverseTest {

  public static void main (String[] args) throws UnknownHostException {
    InetAddress ia = InetAddress.getByName("208.201.239.100");
    System.out.println(ia.getCanonicalHostName());
  }
}
```

```
% java ReverseTest
oreilly.com
```

Example 4-4. Find the IP address of the local machine

```
import java.net.*;

public class MyAddress {

  public static void main(String[] args) {
    try {
      InetAddress me = InetAddress.getLocalHost();
      String dottedQuad = me.getHostAddress();
      System.out.println("My address is " + dottedQuad);
    } catch (UnknownHostException ex) {
      System.out.println("I'm sorry. I don't know my own address.");
    }
  }
}
```

```
% java MyAddress
My address is 152.2.22.14.
```

Example 4-5. Determining whether an IP address is v4 or v6

```
import java.net.*;

public class AddressTests {

  public static int getVersion(InetAddress ia) {
    byte[] address = ia.getAddress();
    if (address.length == 4) return 4;
    else if (address.length == 16) return 6;
    else return -1;
  }
}
```

Address Types

boolean	isAnyLocalAddress()	Utility routine to check if the InetAddress in a wildcard address (0.0.0.0 / ::)
boolean	isLinkLocalAddress()	Utility routine to check if the InetAddress is an IPv6 link local address (Begin with FE80:0000:0000:0000 (8 Bytes) + Local address (often MAC))
boolean	isLoopbackAddress()	Utility routine to check if the InetAddress is a loopback address (127.0.0.1 / ::1)
boolean	isMCGlobal()	Utility routine to check if the multicast address has global scope (IPv4-all Multicast/IPv6-begin with FF0E or FF1E)
boolean	isMCLinkLocal()	Utility routine to check if the multicast address has subnet/link scope (IPv4-all Multicast/IPv6-begin with FF02 or FF12)
boolean	isMCNodeLocal()	Utility routine to check if the multicast address has node scope (for test) (IPv4-all Multicast/IPv6-begin with FF01 or FF11)
boolean	isMCOrgLocal()	Utility routine to check if the multicast address has organization scope (IPv6-begin with FF08 or FF18)
boolean	isMCSiteLocal()	Utility routine to check if the multicast address has site scope (IPv6-begin with FF05 or FF15)
boolean	isMulticastAddress()	Utility routine to check if the InetAddress is an IP multicast address (224.0.0.0~239.255.255.255 / FF00::)
boolean	isReachable(int timeout)	Test whether that address is reachable (Use traceroute/ICMP echo requests)
boolean	isReachable(NetworkInterface netif, int ttl, int timeout)	Test whether that address is reachable
boolean	isSiteLocalAddress()	Utility routine to check if the InetAddress is a IPv6 site local address Like LinkLocalAddress, but May be forwarded by routers (Begin with EEC0:0000:0000:0000 (8 Bytes) + Local address (often MAC))

Example 4-6. Testing characteristics of an IP

```
import java.net.*;

public class IPCharacteristics {

  public static void main(String[] args) {

    try {
      InetAddress address = InetAddress.getByName(args[0]);

      if (address.isAnyLocalAddress()) {
        System.out.println(address + " is a wildcard address.");
      }
      if (address.isLoopbackAddress()) {
        System.out.println(address + " is loopback address.");
      }
      if (address.isLinkLocalAddress()) {
        System.out.println(address + " is a link-local address.");
      } else if (address.isSiteLocalAddress()) {
        System.out.println(address + " is a site-local address.");
      } else {
        System.out.println(address + " is a global address.");
      }

      if (address.isMulticastAddress()) {
        if (address.isMCGlobal()) {
          System.out.println(address + " is a global multicast address.");
        } else if (address.isMCOrgLocal()) {
          System.out.println(address
           + " is an organization wide multicast address.");
        } else if (address.isMCSiteLocal()) {
          System.out.println(address + " is a site wide multicast
                             address.");
        } else if (address.isMCLinkLocal()) {
          System.out.println(address + " is a subnet wide multicast
                             address.");
        } else if (address.isMCNodeLocal()) {
          System.out.println(address
           + " is an interface-local multicast address.");
        } else {
          System.out.println(address + " is an unknown multicast
                             address type.");
        }
      } else {
        System.out.println(address + " is a unicast address.");
      }
    } catch (UnknownHostException ex) {
      System.err.println("Could not resolve " + args[0]);
    }
  }
}
```

```
$ java  IPCharacteristics 127.0.0.1
/127.0.0.1 is loopback address.
/127.0.0.1 is a global address.
/127.0.0.1 is a unicast address.
$ java  IPCharacteristics 192.168.254.32
/192.168.254.32 is a site-local address.
/192.168.254.32 is a unicast address.
$ java  IPCharacteristics www.oreilly.com
www.oreilly.com/208.201.239.37 is a global address.
www.oreilly.com/208.201.239.37 is a unicast address.
$ java  IPCharacteristics 224.0.2.1
/224.0.2.1 is a global address.
/224.0.2.1 is a global multicast address.
```

```
$ java  IPCharacteristics FF01:0:0:0:0:0:0:1
/ff01:0:0:0:0:0:0:1 is a global address.
/ff01:0:0:0:0:0:0:1 is an interface-local multicast address.
$ java  IPCharacteristics FF05:0:0:0:0:0:0:101
/ff05:0:0:0:0:0:0:101 is a global address.
/ff05:0:0:0:0:0:0:101 is a site wide multicast address.
$ java  IPCharacteristics 0::1
/0:0:0:0:0:0:0:1 is loopback address.
/0:0:0:0:0:0:0:1 is a global address.
/0:0:0:0:0:0:0:1 is a unicast address.
```

Object Methods

```
public boolean equals(Object o)
public int hashCode()
public String toString()
```

- equals(): both of InetAddress with the same IP address (not same hostname)
- hashCode(): solely from the IP address; consistent with the equals()
- toString(): has the form of `hostname/dotted quad address`

Example 4-7. Are www.ibiblio.org and helios.ibiblio.org the same?

```
public class IBiblioAliases {

  public static void main (String args[]) {
    try {
      InetAddress ibiblio = InetAddress.getByName("www.ibiblio.org");
      InetAddress helios = InetAddress.getByName("helios.ibiblio.org");
      if (ibiblio.equals(helios)) {
        System.out.println
            ("www.ibiblio.org is the same as helios.ibiblio.org");
      } else {
        System.out.println
            ("www.ibiblio.org is not the same as helios.ibiblio.org");
      }
    } catch (UnknownHostException ex) {
      System.out.println("Host lookup failed.");
    }
  }
}
```

```
% java IBiblioAliases
www.ibiblio.org is the same as helios.ibiblio.org
```

4.2 Inet4Address and Inet6Address

```
public final class Inet4Address extends InetAddress
public final class Inet6Address extends InetAddress
```

- Both overrides several of the methods in InetAddress but does not change their behavior in
  - Most of the time, simply not needed to know this
- Inet6Address.isIPv4CompatibleAddress(): one new method
  - Only the last four bytes are nonzero – IPv4 address stuffed into an IPv6
  - 0:0:0:0:0:0:d.d.d.d

4.3 The NetworkInterface Class

http://docs.oracle.com/javase/8/docs/api/java/net/NetworkInterface.html

- java.net.NetworkInterface objects represent physical hardware and virtual addresses

static **NetworkInterface**	**getByIndex**(int index) Get a network interface given its index
static **NetworkInterface**	**getByInetAddress**(**InetAddress** addr) Convenience method to search for a network interface that has the specified Internet Protocol (IP) address bound to it
static **NetworkInterface**	**getByName**(**String** name) Searches for the network interface with the specified name
Enumeration <**InetAddress**>	**getInetAddresses**() Convenience method to return an Enumeration with all or a subset of the InetAddresses bound to this network interface
List<**InterfaceAddress**>	**getInterfaceAddresses**() Get a List of all or a subset of the InterfaceAddresses of this network interface
static **Enumeration** <**NetworkInterface**>	**getNetworkInterfaces**() Returns all the interfaces on this machine
NetworkInterface	**getParent**() Returns the parent NetworkInterface of this interface if this is a subinterface, or null if it is a physical (non virtual) interface or has no parent
Enumeration <**NetworkInterface**>	**getSubInterfaces**() Get an Enumeration with all the subinterfaces (also known as virtual interfaces) attached to this network interface

NetworkInterface Examples

getByName()

```
try {
  NetworkInterface ni = NetworkInterface.getByName("eth0");
  if (ni == null) {
    System.err.println("No such interface:  eth0");
  }
} catch (SocketException ex) {
  System.err.println("Could not list sockets.");
}
```

getByInetAddress()

```
try {
  InetAddress local = InetAddress.getByName("127.0.0.1");
  NetworkInterface ni = NetworkInterface.getByInetAddress(local);
  if (ni == null) {
    System.err.println("That's weird. No local loopback address.");
  }
} catch (SocketException ex) {
  System.err.println("Could not list network interfaces." );
} catch (UnknownHostException ex) {
  System.err.println("That's weird. Could not lookup 127.0.0.1.");
}
```

Example 4-8. A program that lists all the network interfaces

```
import java.net.*;
import java.util.*;

public class InterfaceLister {

  public static void main(String[] args) throws SocketException {
    Enumeration<NetworkInterface> interfaces = NetworkInterface.
    getNetworkInterfaces();
    while (interfaces.hasMoreElements()) {
      NetworkInterface ni = interfaces.nextElement();
      System.out.println(ni);
    }
  }
}
```

```
% java InterfaceLister
name:eth1 (eth1) index: 3 addresses:
/192.168.210.122;

name:eth0 (eth0) index: 2 addresses:
/152.2.210.122;

name:lo (lo) index: 1 addresses:
/127.0.0.1;
```

NetworkInterface Getter Methods

boolean	**equals**(**Object** obj) Compares this object against the specified object
String	**getDisplayName**() Get the display name of this network interface
byte[]	**getHardwareAddress**() the hardware address (usually MAC) of the interface if it has one and if it can be accessed given the current privileges
int	**getIndex**() Returns the index of this network interface
Enumeration <**InetAddress**>	**getInetAddresses**() Convenience method to return an Enumeration with all or a subset of the InetAddresses bound to this network interface
List <**InterfaceAddress**>	**getInterfaceAddresses**() Get a List of all or a subset of the InterfaceAddresses of this network interface
int	**getMTU**() Returns the Maximum Transmission Unit (MTU) of this interface
String	**getName**() Get the name of this network interface
NetworkInterface	**getParent**() Returns the parent NetworkInterface of this interface if this is a subinterface, or null if it is a physical (non virtual) interface or has no parent
Enumeration <**NetworkInterface**>	**getSubInterfaces**() an Enumeration with all the subinterfaces (also known as virtual interfaces) attached to this network interface
int	**hashCode**() Returns a hash code value for the object.
boolean	**isLoopback**() Returns whether a network interface is a loopback interface.
boolean	**isPointToPoint**() Returns whether a network interface is a point to point interface.
boolean	**isUp**() Returns whether a network interface is up and running.
boolean	**isVirtual**() Returns whether this interface is a virtual interface (also called subinterface).
boolean	**supportsMulticast**() Returns whether a network interface supports multicasting or not.
String	**toString**() Returns a string representation of the object.

4.4 Some Useful Programs

- SpamCheck: asks sbl.spamhaus.org if an IPv4 is a spammer
  - i.e. A DNS query for 17.34.87.207.sbl.spamhaus.org succeeds (/returns 127.0.0.2) if 17.34.87.207 is a spammer

- Processing Web Server Logfiles: reads a web server logfile and prints each line with IP addresses converted to hostnames
  - Usually a Web server simply logs the IP addresses and converts them to hostnames at a later time
  - Common logfile format:

```
205.160.186.76 unknown - [17/Jun/2013:22:53:58 -0500] "GET /bgs/greenbg.gif HTTP 1.0" 200 50
```

Example 4-9. SpamCheck

```
import java.net.*;

public class SpamCheck {

  public static final String BLACKHOLE = "sbl.spamhaus.org";
  public static void main(String[] args) throws UnknownHostException {
    for (String arg: args) {
      if (isSpammer(arg)) {
        System.out.println(arg + " is a known spammer.");
      } else {
        System.out.println(arg + " appears legitimate.");
      }
    }
  }

  private static boolean isSpammer(String arg) {
    try {
      InetAddress address = InetAddress.getByName(arg);
      byte[] quad = address.getAddress();
      String query = BLACKHOLE;
      for (byte octet : quad) {
        int unsignedByte = octet < 0 ? octet + 256 : octet;
        query = unsignedByte + "." + query;
      }
      InetAddress.getByName(query);
      return true;
    } catch (UnknownHostException e) {
      return false;
    }
  }
}
```

```
$ java SpamCheck 207.34.56.23 125.12.32.4 130.130.130.130
207.34.56.23 appears legitimate.
125.12.32.4 appears legitimate.
130.130.130.130 appears legitimate.
```

- Read IPv4 address list from the command line
- Send DNS query d.c.b.a.sbl.spamhaus.org for each IPv4 address of a.b.c.d
  – The query succeeds if it is a spammer

Example 4-10. Process Logfiles (Single Thread)

```
import java.io.*;
import java.net.*;

public class Weblog {

  public static void main(String[] args) {
    try (FileInputStream fin =  new FileInputStream(args[0]);
      Reader in = new InputStreamReader(fin);
      BufferedReader bin = new BufferedReader(in);) {

      for (String entry = bin.readLine();
        entry != null;
        entry = bin.readLine()) {
        // separate out the IP address
        int index = entry.indexOf(' ');
        String ip = entry.substring(0, index);
        String theRest = entry.substring(index);

        // Ask DNS for the hostname and print it out
        try {
          InetAddress address = InetAddress.getByName(ip);
          System.out.println(address.getHostName() + theRest);
        } catch (UnknownHostException ex) {
          System.err.println(entry);
        }
      }
    } catch (IOException ex) {
      System.out.println("Exception: " + ex);
    }
  }
}
```

205.160.186.76 unknown - [17/Jun/2013:22:53:58 -0500]
"GET /bgs/greenbg.gif HTTP 1.0" 200 50

- It spends a huge amount of time sitting and waiting for DNS requests to return
- A thread pool is absolutely necessary
  - One main thread reads the logfile and
  - Passes off individual entries to other threads for processing

Example 4-11. Process Logfiles (Thread Pool)

```
import java.net.*;
import java.util.concurrent.Callable;

public class LookupTask implements Callable<String> {

  private String line;

  public LookupTask(String line) {
    this.line = line;
  }

  @Override
  public String call() {
    try {
      // separate out the IP address
      int index = line.indexOf(' ');
      String address = line.substring(0, index);
      String theRest = line.substring(index);
      String hostname = InetAddress.getByName(address).getHostName();
      return hostname + " " + theRest;
    } catch (Exception ex) {
      return line;
    }
  }
}
```

```
public class PooledWeblog {

  private final static int NUM_THREADS = 4;

  public static void main(String[] args) throws IOException {
    ExecutorService executor = Executors.newFixedThreadPool(NUM_THREADS);
    Queue<LogEntry> results = new LinkedList<LogEntry>();

    try (BufferedReader in = new BufferedReader(
      new InputStreamReader(new FileInputStream(args[0]), "UTF-8"));) {
      for (String entry = in.readLine(); entry != null; entry = in.readLine()) {
        LookupTask task = new LookupTask(entry);
        Future<String> future = executor.submit(task);
        LogEntry result = new LogEntry(entry, future);
        results.add(result);
      }
    }

    // Start printing the results. This blocks each time a result isn't ready..
    for (LogEntry result : results) {
      try {
        System.out.println(result.future.get());
      } catch (InterruptedException | ExecutionException ex) {
        System.out.println(result.original);
      }
    }

    executor.shutdown();
  }
  private static class LogEntry {
    String original;
    Future<String> future;

    LogEntry(String original, Future<String> future) {
     this.original = original;
     this.future = future;
    }
  }
}
```

1. Callback per entry
2. Add callback to queue
3. Wait results in order

- Processed in parallel
  - 10x-50x faster

```
import java.io.*;
import java.util.*;
import java.util.concurrent.*;

// Requires Java 7 for try-with-resources and multi-catch
```

Summary

4.1 The InetAddress Class

http://docs.oracle.com/javase/8/docs/api/java/net/InetAddress.html

- Create Objects and Getter Methods
- Example 4-1. Print the address of a host name (OreillyByName)
- Example 4-2. Print the address of the machine it's run on (MyAddress)
- Example 4-3. Given the address, find the hostname (ReverseTest)
- Example 4-4. Find the IP address of the local machine (MyAddress)
- Example 4-5. Determining whether an IP address is v4 or v6 (AddressTests)
- Example 4-6. Testing characteristics of an IP address (IPCharacteristics)
- Example 4-7. Check whether two host names are the same (IBiblioAliases)

4.2 Inet4Address and Inet6Address

http://docs.oracle.com/javase/8/docs/api/java/net/Inet4Address.html
http://docs.oracle.com/javase/8/docs/api/java/net/Inet6Address.html

4.3 The NetworkInterface Class

http://docs.oracle.com/javase/8/docs/api/java/net/NetworkInterface.html

- Example 4-8. List all the network interfaces (InterfaceLister)

4.4 Some Useful Programs

- Example 4-9. SpamCheck (SpamCheck)
- Example 4-10. Process LogFiles – Single Thread (Weblog)
- Example 4-11. Process LogFiles – Thread Pool (LookupTask, PooledWeblog)