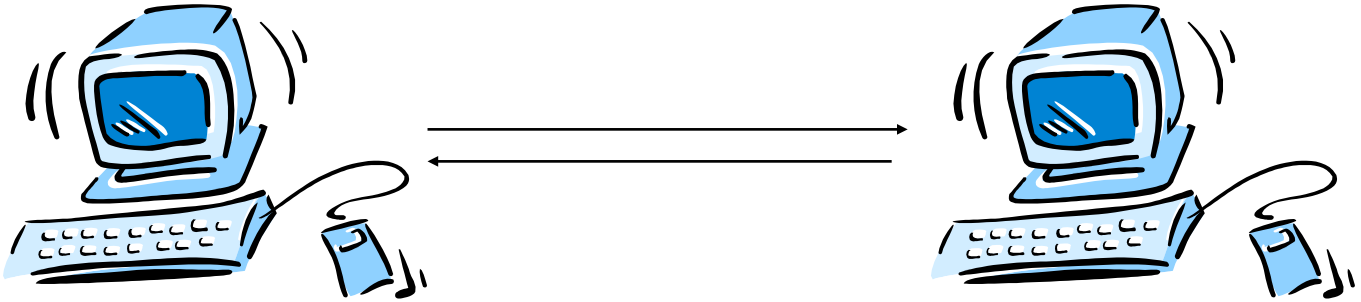


95-702 Distributed Systems

Naming

Sources: “Computer Networks” by Peterson and Davie and Sun Microsystems Java documentation.

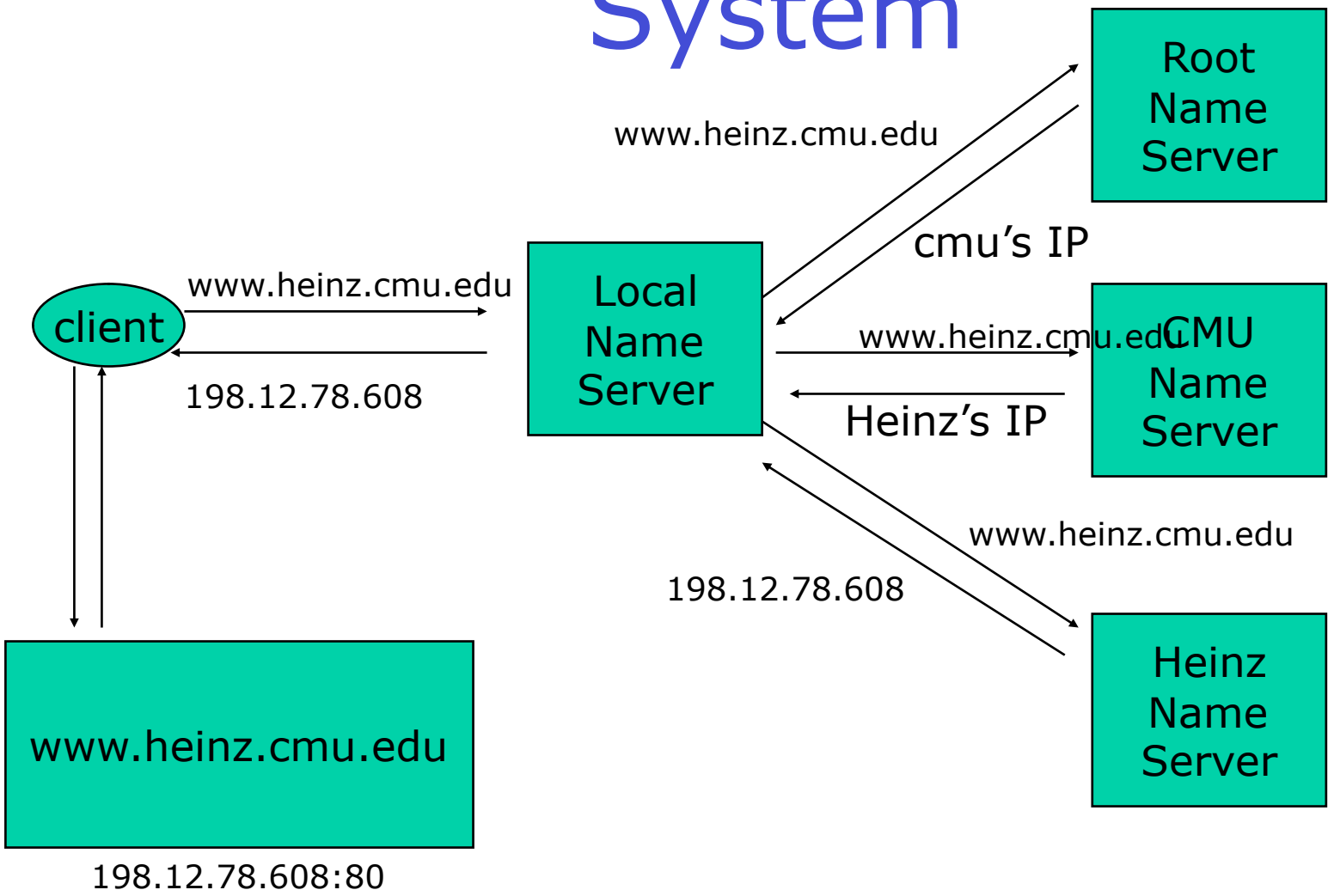
Name Services

- User friendly name usually vary in length and are mnemonic.
- Little in a user friendly name helps locate a host.
- Addresses, by contrast, may have imbedded routing information.
- Flat addresses (those not divisible into component parts) are the exception.
- IP addresses have a network part and a host part.
- IP addresses are used by routers.
- But IP addresses are not user friendly.
- A system is needed to map user friendly names to router friendly addresses.

Terminology

- A namespace defines a set of possible name.
- In a flat namespace, names are not divisible into components.
- In a hierarchical namespace, names are broken into components. E.g., Unix, DOS, URL's, etc.
- A naming system is a collection of bindings of names to values. These values are often addresses.
- A resolution mechanism is a procedure that, when invoked with a name, returns a corresponding value.
- A name server is a specific implementation of a resolution mechanism that is available on a network and can be queried by sending it a message.

Example: Domain Name System



Naming Concepts

- Very common problem: map people friendly names to objects.
- Examples:

Notice the hierarchies in each case.

mm6@andrew.cmu.edu -> Mike's mailbox

www.cnn.com -> cnn's web server

c:\somedir\f1.dat -> a file on my C drive

cn=Rosanna Lee, o=Sun, c=US -> info about Rosanna

Naming Conventions

- Different naming systems use different conventions (or syntax) for names
- Examples:

  DOS uses slashes and colons and periods c:\some\f.dat

  Unix uses slashes /usr/local/filename

  DNS uses dots www.cnn.com

  LDAP (The lightweight directory access protocol) uses name, value pairs cn=Rosanna Lee, o=Sun, c=US

Context

A *context* is a set of name-to-object bindings.

Every context has an associated naming convention.

A context may allow operations such as bind, unbind, lookup.

A context may associate a name with another context (subcontext, or subdirectory).

Directory Service

- A *Directory Service* is an extension of a naming service that allows one to lookup objects based on names or based on attributes.
- Attributes have *attribute identifiers* and a set of *attribute values*.
- Quiz: What is the directory service often associated with web services?

Answer: UDDI - Universal Description, Discovery and Integration

Also called Reverse Lookup or Content-Based searching

- Example queries to directory services:

  Find all machines whose IP address begins with 192.115.50.

  Find all companies that provide hardware support services.

Directory Enabled Applications

A *directory-enabled* application is an application that uses a naming or directory service.

Applications can share the common infrastructure provided by the directory.

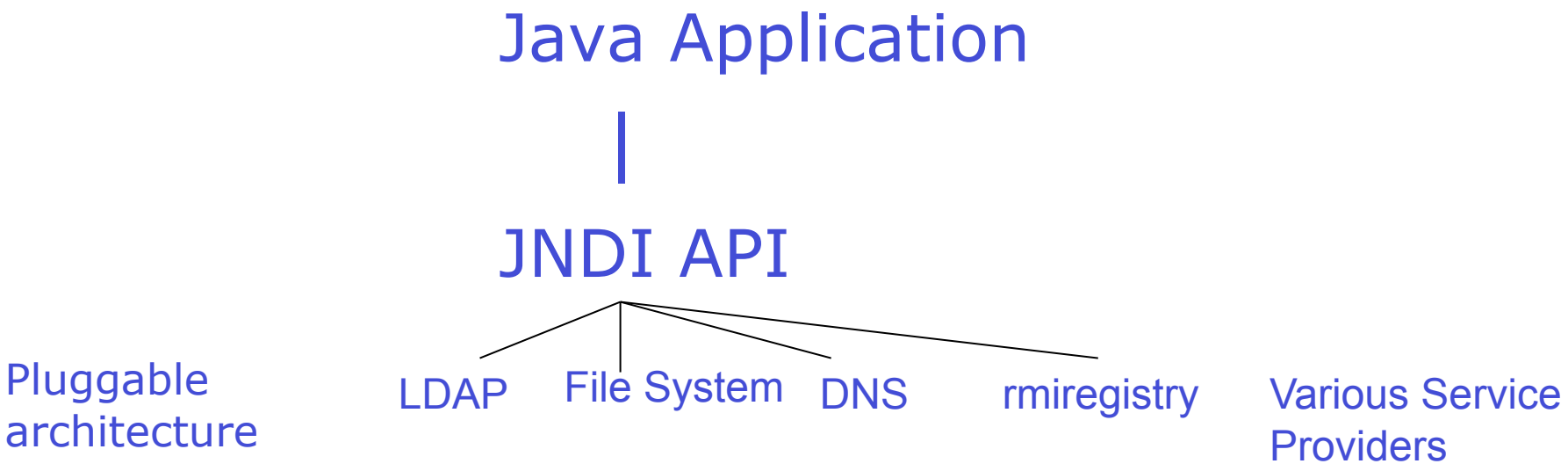
Example: A mail client, scheduling systems and mail forwarding program might all use the same address book stored in a common directory.

The directory may also be used as an object store for programs needing the same object.

Java Naming and Directory Interface JNDI

The Java Naming and Directory Interface (JNDI):

- Is an abstraction API (just like JDBC is an abstraction API on RDBMS databases).
- Handles or sits on top of different naming services.



JNDI

- The javax.naming packages contains mostly Java interfaces.
- Some vendor implements the interface to provide JNDI support for their service.
- To use the service, you need a JNDI service provider that implements the interface.
- JDK1.4 comes with RMI, DNS, COS, and LDAP service providers.
- Sun's web site has an additional JNDI service provider that works with the local file system

Namespaces are represented by the Context Interface

- Different classes implement this interface differently depending on which naming service they are accessing.
- Has methods to
  - bind and unbind objects to names
  - create and delete sub-contexts
  - lookup and list names
- Since a Context is a Java object it can be registered in another Context with its own name.

The Context Interface

- Start from some “root” context.
- Get the “root” from the InitialContext class
- Two simple examples:

  LookUp.java

  ListCurrentDirectory.java

LookUp.java

```
// pre: download JNDI provider from Sun
// add .jar files to classpath

import javax.naming.Context;
import javax.naming.InitialContext;
import javax.naming.NamingException;
import java.util.Hashtable;
import java.io.File;
```

```
public class LookUp {

  public static void main(String args[]) throws NamingException {

   try {

     System.out.println("Using a file system (FS) provider");

     // initialize the context with properties for provider
     // and current directory
     Hashtable env = new Hashtable();
     env.put(Context.INITIAL_CONTEXT_FACTORY,
                "com.sun.jndi.fscontext.RefFSContextFactory");
     env.put(Context.PROVIDER_URL,
                 "file:D:\\McCarthy\\www\\95-702\\examples\\JNDI");

     Context ctx = new InitialContext(env);

     Object obj = ctx.lookup(args[0]);
```

```
    if(obj instanceof File) {

        System.out.println("Found a file object");

        System.out.println(args[0] + " is bound to: " + obj);

        File f = (File) obj;

        System.out.println(f + " is " + f.length() + " bytes long");
     }
    // Close the context when we're done
    ctx.close();
   }
   catch(NamingException e) {
       System.out.println("Naming exception caught" + e);
   }
  }
}
```

```
D:\McCarthy\www\95-702\examples\JNDI>java LookUp LookUp.java
Using a file system (FS) provider
Found a file object
LookUp.java is bound to: D:\McCarthy\www\95-702\examples\JNDI\LookUp.java
D:\McCarthy\www\95-702\examples\JNDI\LookUp.java is 1255 bytes long
```

ListCurrentDirectory.java

```
// Use JNDI to list the contents of the current
// directory

import javax.naming.Context;
import javax.naming.InitialContext;
import javax.naming.NamingException;
import javax.naming.NamingEnumeration;
import javax.naming.NameClassPair;
import java.util.Hashtable;
import java.io.File;
```

```
public class ListCurrentDirectory  {

  public static void main(String args[]) throws NamingException {

   try {

     Hashtable env = new Hashtable();
     env.put(Context.INITIAL_CONTEXT_FACTORY,
                "com.sun.jndi.fscontext.RefFSContextFactory");
     env.put(Context.PROVIDER_URL,
                 "file:D:\\McCarthy\\www\\95-702\\examples\\JNDI");
```

```
Context ctx = new InitialContext(env);

NamingEnumeration list = ctx.list(".");

while (list.hasMore()) {
    NameClassPair nc = (NameClassPair)list.next();
    System.out.println(nc);
 }
ctx.close();
}
catch(NamingException e) {
    System.out.println("Naming exception caught" + e);
}
  }
}
```

```
D:\McCarthy\www\95-702\examples\JNDI>java ListCurrentDirectory
ListCurrentDirectory.class: java.io.File
ListCurrentDirectory.java: java.io.File
LookUp.java: java.io.File
SimpleJNDI.java: java.io.File
x: javax.naming.Context
```

x is a DOS directory

```
// Use JNDI to change to a sub directory and list contents

import javax.naming.Context;
import javax.naming.InitialContext;
import javax.naming.NamingException;
import javax.naming.NamingEnumeration;
import javax.naming.NameClassPair;
import java.util.Hashtable;
import java.io.File;

public class ChangeContext  {

  public static void main(String args[]) throws NamingException {

   try {
     Hashtable env = new Hashtable();
     env.put(Context.INITIAL_CONTEXT_FACTORY,
                "com.sun.jndi.fscontext.RefFSContextFactory");
     env.put(Context.PROVIDER_URL,
                "file:D:\\McCarthy\\www\\95-702\\examples\\JNDI");
```

```
    Context ctx = new InitialContext(env);

    // a subdirectory called x contains a file f.txt and a subdirectory t
    Context sub = (Context)ctx.lookup("x");

    NamingEnumeration list = sub.list(".");

    while (list.hasMore()) {
        NameClassPair nc = (NameClassPair)list.next();
        System.out.println(nc);
     }
    ctx.close();
    sub.close();

   }
   catch(NamingException e) {
       System.out.println("Naming exception caught" + e);
   }
  }
}
```

```
D:\McCarthy\www\95-702\examples\JNDI>java ChangeContext

f.txt: java.io.File
t: javax.naming.Context
```