A tutorial on Jini

Jan Newmarch
jan@ise.canberra.edu.au
University of Canberra
DSTC (Distributed Systems Technology CRC)

OVERVIEW

 © Jan Newmarch

What is Jini?

- Jini brings ‘‘network plug and play’’ to hardware devices and software services
- Jini is in pure Java
- It makes hardware and software available as Java networked services
- ‘‘Legacy’’ systems can be wrapped in a Java proxy
- Jini has competitors including CORBA Trader, DCOM, HP’s Chai, Enterprise Java Beans, JNDI, etc

Jini Federation

- A Jini federation or ‘‘djinn’’ consists of a number of clients, services and service locators
- Typically connected by TCP/IP
- Code is moved around from service to locator to client using Java 1.2 extensions to RMI

 © Jan Newmarch

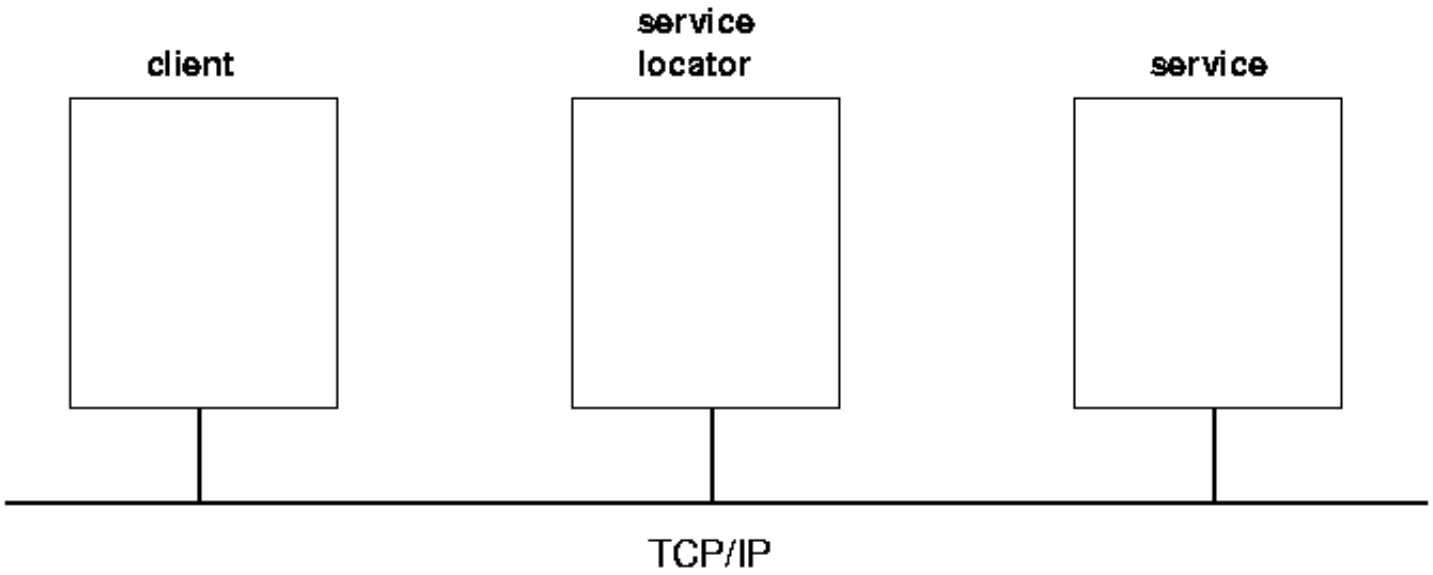


Figure 1: Components of a Jini federation

Service Registration

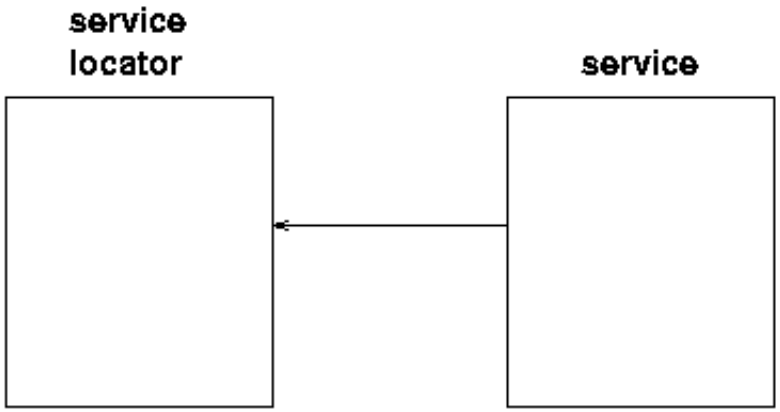


Figure 2: Querying for a service locator

 © Jan Newmarch

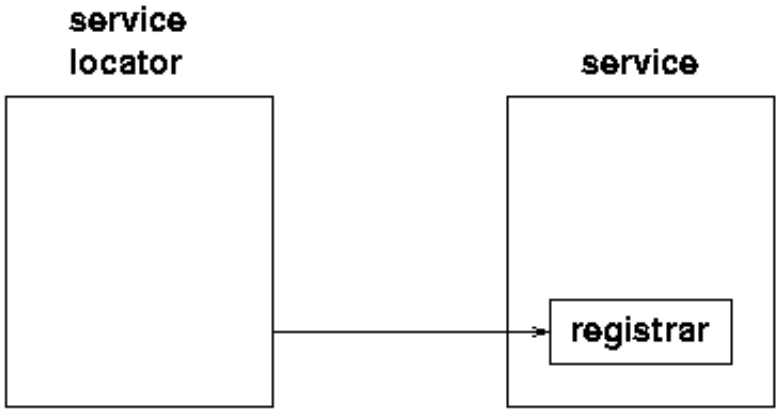


Figure 3: Registrar returned

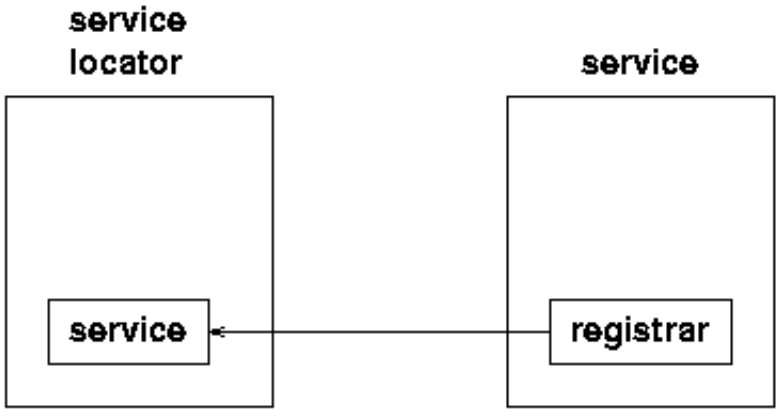


Figure 4: Service uploaded

Client Lookup

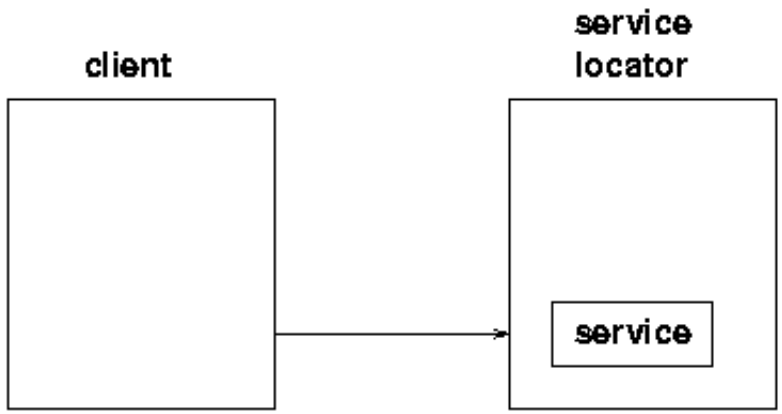


Figure 5: Querying for a service locator

 © Jan Newmarch

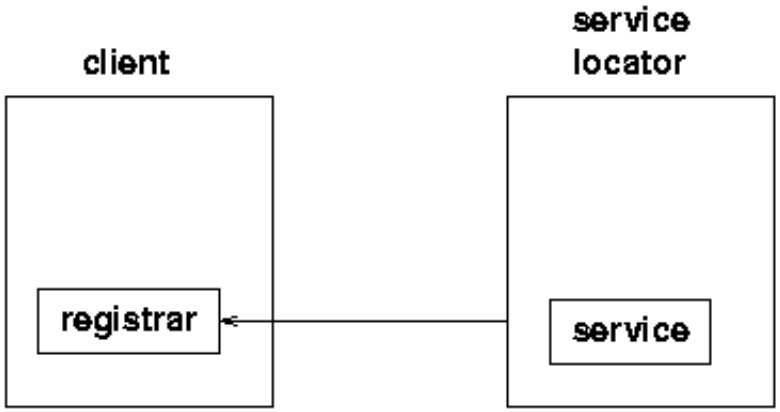


Figure 6: Registrar returned

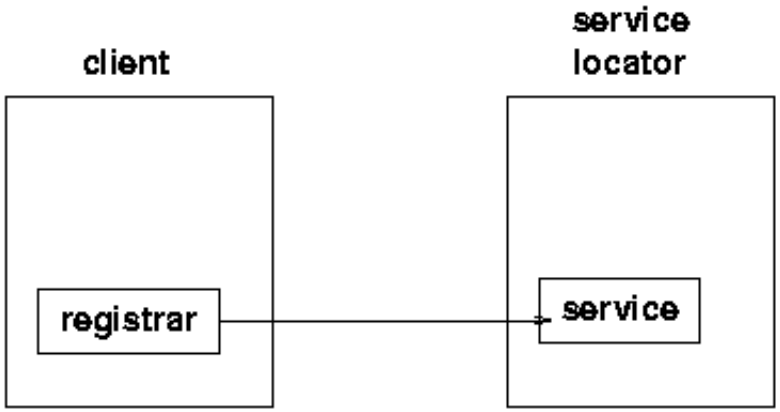


Figure 7: Asking for a service

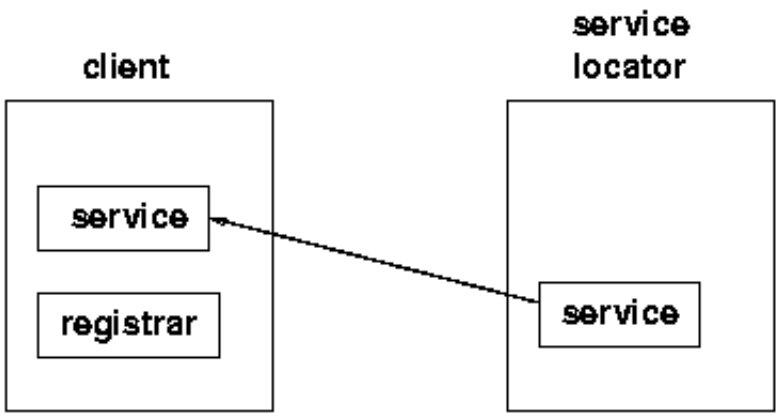


Figure 8: Service returned

 © Jan Newmarch

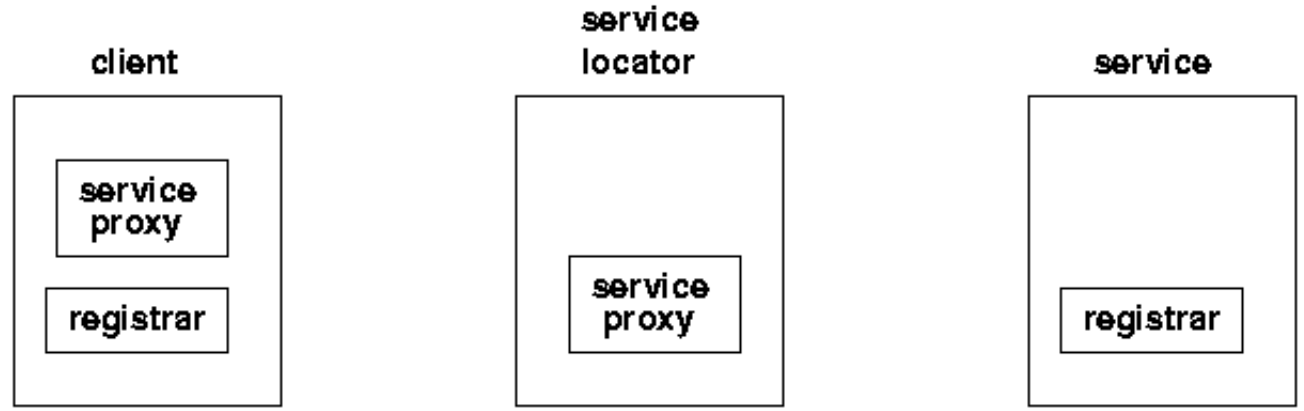
Support Services

- Jini objects are moved in ‘‘marshalled’’ form
- Class files may need to follow these for proper activation
- The class files may need to be available from HTTP servers
- RMI Naming can be used to locate remote objects
- RMI Naming may need an rmiregistry or rmid running
- Security permissions will need to be set using JDK 1.2 security model

Proxies

- A toaster will be controlled by some service software
- The service software will be moved across the network to the client
- The mobile code must be a proxy for the real service code
- The proxy may be explicitly written or be an RMI proxy stub
- Remote calls from the proxy may be RMI Remote calls, or by some other mechanism such as private TCP connection

 © Jan Newmarch



Proxy Service

Attribute Registration

- A service is uploaded to a locator as an instance of a class
- Additional information such as address may also be uploaded as instances of Entry classes
- Service registration is done using a leasing system

 © Jan Newmarch

Service Location

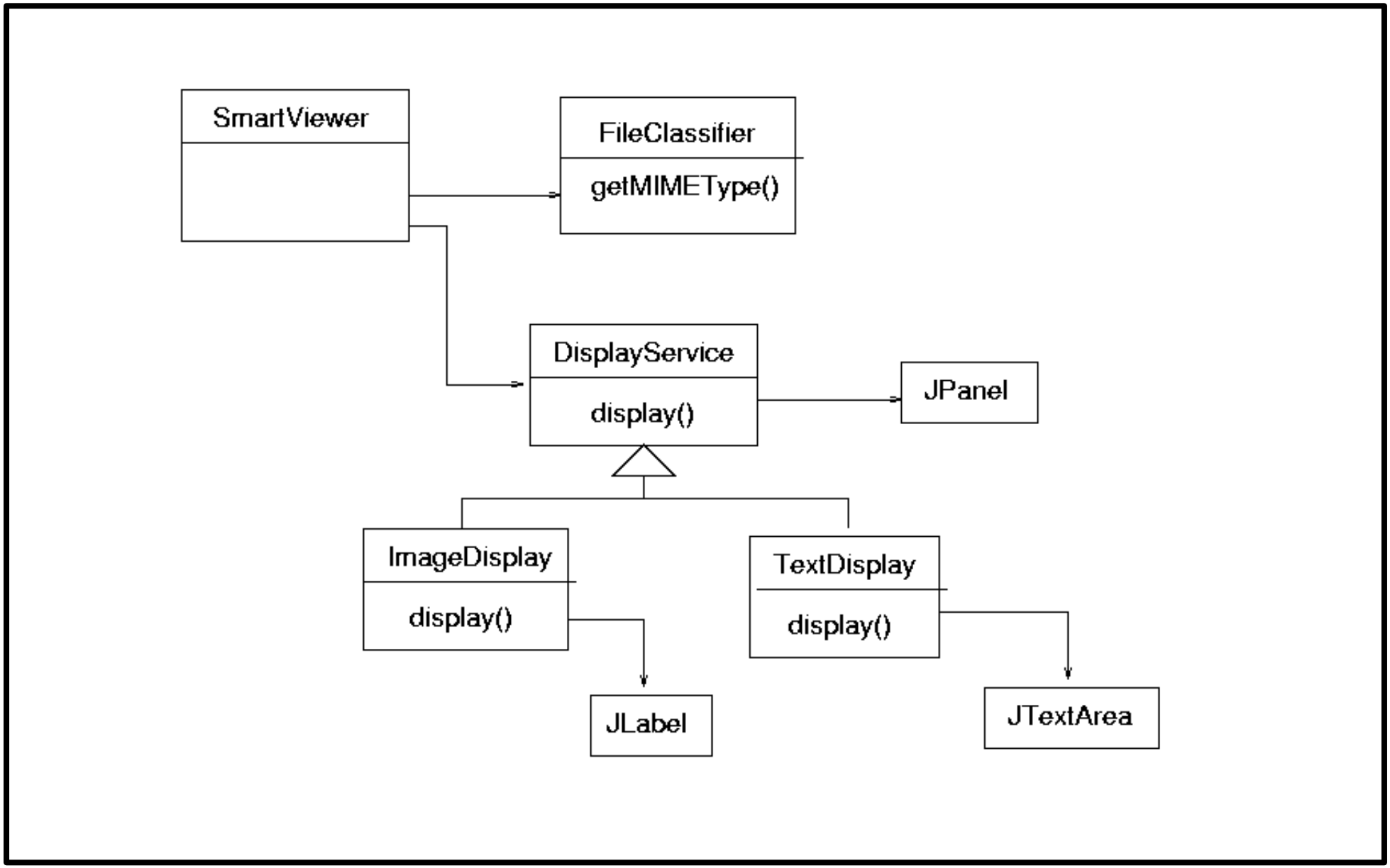
- A client asks for a service as an instance of a class, typically of an interface
- A client may ask for additional Entry attributes
- Matching on fields is *exact*
- You can ask for a printer with a certain speed, but not for one *greater than* a certain speed
- Additional services may be used if complex boolean expressions are needed

Leasing

- Because services can come and go, network connections can be lost, etc, a distributed garbage collection mechanism is needed
- Services which register are granted a lease
- Registration lasts for as long as the lease
- When the lease expires, the service is removed
- Leases can be renewed

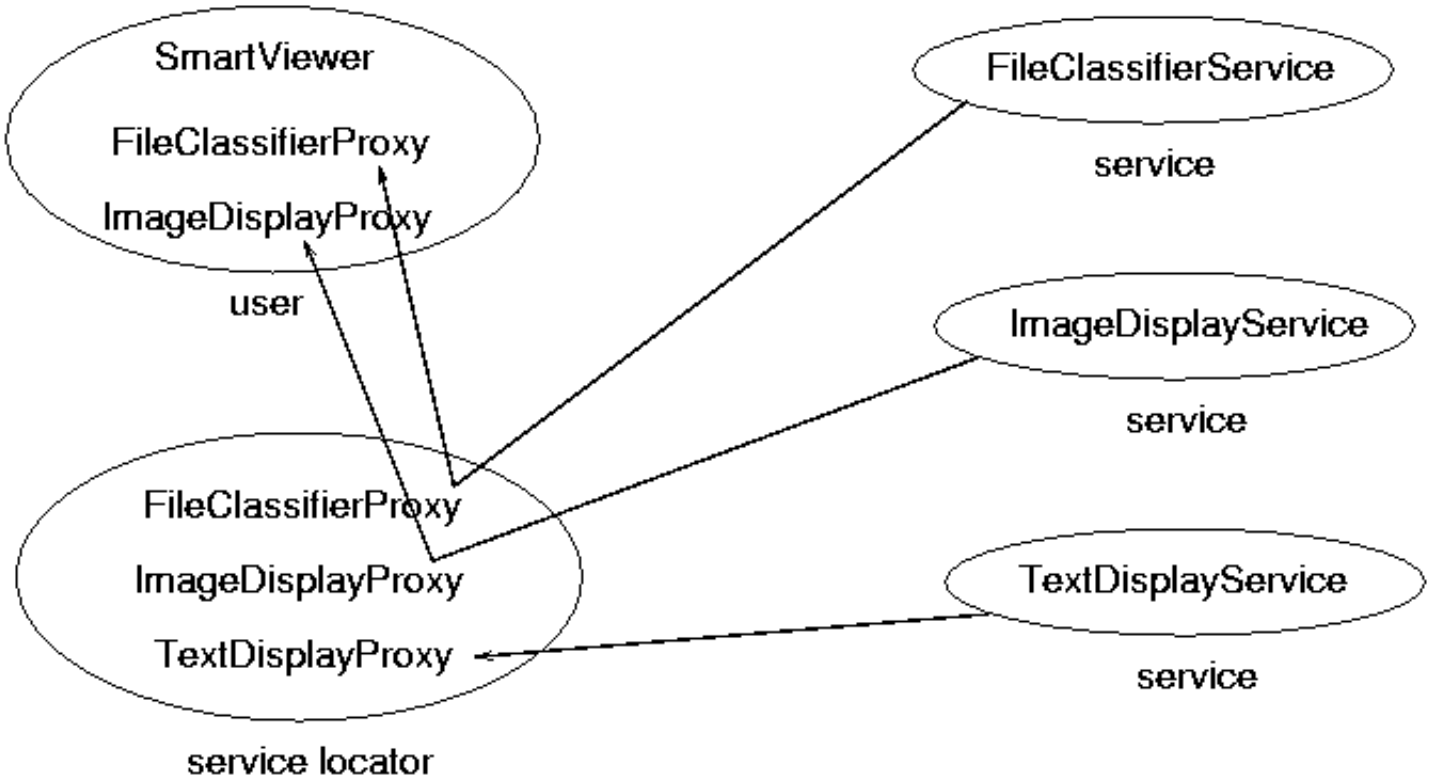
 © Jan Newmarch

Non-distributed Application



 © Jan Newmarch

Distributed Version



Client Structure

Internally a client will look like

```
prepare for discovery
discover a lookup service
prepare a template for lookup search
lookup a service
call the service
```

The following code is simplified from the real case. It attempts to find a `FileClassifier` service, calling the method `getMIMEType()` on this service.

 © Jan Newmarch

```
public class TestUnicastFileClassifier {

    public static void main(String argv[]) {
        new TestUnicastFileClassifier();
    }

    public TestUnicastFileClassifier() {
        LookupLocator lookup = null;
        ServiceRegistrar registrar = null;
        FileClassifier classifier = null;

        // Prepare for discovery
        lookup = new LookupLocator("jini://www.all_about_files.com");

        // Discover a lookup service
        // This uses the synchronous unicast protocol
        registrar = lookup.getRegistrar();

        // Prepare a template for lookup search
        Class[] classes = new Class[] {FileClassifier.class};
        ServiceTemplate template = new ServiceTemplate(null, classes, null);

        // Lookup a service
```

```
        classifier = (FileClassifier) registrar.lookup(template);

        // Call the service
        MIMEType type;
        type = classifier.getMIMEType("file1.txt");
        System.out.println("Type is " + type.toString());
    }
} // TestUnicastFileClassifier
```

 © Jan Newmarch

Server Structure

```
prepare for discovery
discover a lookup service
create information about a service
export a service
renew leasing periodically
```

The following code is simplified from the real case. It exports an implementation of a file classifier service, as a `FileClassifierImpl` object.

```
public class FileClassifierServer implements DiscoveryListener {

    protected LeaseRenewalManager leaseManager = new LeaseRenewalManager();

    public static void main(String argv[]) {
```

```
        new FileClassifierServer();

        // keep server running forever to
        // - allow time for locator discovery and
        // - keep re-registering the lease
        Thread.currentThread().sleep(Lease.FOREVER);
    }

    public FileClassifierServer() {
        LookupDiscovery discover = null;

        // Prepare for discovery - empty here

        // Discover a lookup service
        // This uses the asynchronous multicast protocol,
        // which calls back into the discovered() method
        discover = new LookupDiscovery(LookupDiscovery.ALL_GROUPS);

        discover.addDiscoveryListener(this);
    }

    public void discovered(DiscoveryEvent evt) {
```

 © Jan Newmarch

```
        ServiceRegistrar registrar = evt.getRegistrars()[0];
        // At this point we have discovered a lookup service

        // Create information about a service
        ServiceItem item = new ServiceItem(null,
                                           new FileClassifierImpl(),
                                           null);

        // Export a service
        ServiceRegistration reg = registrar.register(item, Lease.FOREVER);

        // Renew leasing
        leaseManager.renewUntil(reg.getLease(), Lease.FOREVER, this);
    }
} // FileClassifierServer
```

Jan Newmarch (http://pandonia.canberra.edu.au)

jan@ise.canberra.edu
Last modified: Sat Sep 4 00:14:09 EST 1999

Copyright ©Jan Newmarch