Prof. Dr. Thomas Schmidt
HAW Hamburg, Dept. Informatik
Raum 780, Tel.: 42875 - 8452
Email: schmidt@informatik.haw-hamburg.de
Web:
http://inet.cpt.haw-hamburg.de/teaching/ss-2009/internet-technologies

Internet Technologies I

LAB Assignment

Goal: Implementation of a simple peer-to-peer resource sharing service in Java.

1st Step (Basics):

Implement a simple http-server with the following properties (*use Java.net package*):

- The server has a configurable (listener-) port taken from a config file.
- The server communicates via http – check with a standard browser!
- The server reads its 'resource table' (table of files) from a local, pre-configured directory and transfers a directory file listing, when no resource is specified in the URL.
- The server transfers resources (files), if specified in the URL.

Implement a simple http client with the following properties:

- The client has a graphical user interface (*Java AWT or Swing*), which allows for configuration of server port and input of a URL.
- The client can display the simple html-based directory listing and download resources (files) to the file system.

2nd Step (Virtualized Resource Server):

Extend your server data structure to provide the following XML datasets (read from the config or taken from the system data resp., *use org.w3c.dom package*):

- Provider Profile:

```
<provider>
  <name> … </name>                    # Name of the provider
   <descriptor> … </descriptor>       # Text description of provider
   <current-ip> …</current-ip>        # Current IP address
   <current-port> … </current-port>   # port of your tcp-interface
   <bandwidth> … </bandwidth>         # Bandwidth of connectivity
   <res-locator> … </res-locator>     # Name of resource locator file
   <res-loc-date> … </res-loc-date>   # Current date of resource locator file
</provider>
```

- Resource Table:

```
<rtable>
  <resource>
```

```
        <title> … </title>
        <provider> … </provider>      # provider name as in profile
                ….
        <provider> … </provider>
    </resource>
      ….
    <resource>
    </resource>
</rtable>
```

- Make the resource table available via http (test with standard browser!)

3rd Step (Simple Overlay Network):

Extend your server to include a multicast resource announcement and reception function, which will allow every server to learn about all other resources within the multicast domain.

Proceed along the following sub-steps:

- Choose the common multicast group 239.238.237.17 and common port 9017 for your announcement channel.
- Add a multicast sender function to your server, which announces your own provider profile every 5 minutes.
- Add a multicast listener function to your server, which reads profile announcements and updates/adds these to its provider profiles.

Check for coexistence/interoperability of your servers.

4th Step (Integrated Application & Testing):

Extend the user interface support of your server, such that a user will be able to retrieve any resource of the overlay network through information provided by your server. Proceed as follows:

- Complement your server's resource listing (as of 1st Step) to include all resource providers (by name). (All data transferred to standard browser as HTML via http, including the obvious hyperlinks!)
- On click have your server retrieve and display the resource table of any known provider.
- Test and experiment with other groups on interoperability and correctness!

Optional: (Overlay Network Routing):

Extend your servers to operate in several networks (different multicast groups/ports!) simultaneously. Forward provider data you know between networks. Autonomously evaluate all resources present in all networks and providers known to your server and determine optimal shortest path access to any specific resource you know of.