

**Basics of Computer Networks**  
**COMP-435B, Winter 2003**  
**Assignment 4**

**Due date: April 11, 2003**

Note: Assignments must be handed in in class. Late assignments will only be accepted with prior **written** permission of the instructor.

## 1 Assignment Questions

Note: You must explain all answers and show all work for full marks.

1. Pick a source host you have access to (eg `mimi` or `willy`) and an arbitrary destination host on the internet (try to pick a different destination host than any of your classmates; it must be at least 10 hops away from your source). Use the unix command `traceroute` (or equivalent, eg `tracert` on Windows NT/2000) to find the route taken from the source to the destination. Show the route; how many hops does it take to reach your destination? Many of the host names listed will suggest companies and/or geographic locations; as best as you can discern, how many separate organizations and which physical locations does your route traverse? **1**
2. A router has the following routing table for IP addresses:

Entry	Next Hop	Entry	Next Hop
160.0.0.0/4	A	56.0.0.0/5	E
144.0.0.0/6	B	129.0.0.0/8	F
128.0.0.0/2	C	62.0.0.0/7	G
64.0.0.0/2	D		

Does the router know where to forward packets for each of the following destinations? In each case explain what the router will do with a packet destined for the given address:

- (a) 130.202.6.44   (b) 128.44.12.9   (c) 126.100.0.11   (d) 63.63.0.199   **3**  
(e) 53.224.66.88   (f) 147.204.6.33
3. Using classic IP rules (ie classes A,B,C,D,E), how many actually usable, public host addresses are there? **2**

## 2 Programming

In this assignment you will implement a distributed network of concurrent, non-caching file servers. Concurrent servers avoid blocking behaviour by spawning multiple processes or threads to handle each incoming request; blocking in one process or thread then does not affect the ability to receive or process other requests.

Define a program that does the following:

1. A base directory, a port to listen on, as well as a list of other file servers (IP address and port) are all given on the command line.
2. Creates and listens to a TCP server socket on the given port. Incoming requests are formatted as:

"FILE:" (5 bytes)	name-of-file (variable length)	newline (1 byte)
-------------------	--------------------------------	------------------

where name-of-file is at least one character long, starts with a letter (upper or lowercase) or underscore, and then includes only elements of {A-Z,a-z,0-9,-,+,.,}.

- An incoming request should first be examined. If the file described in the packet is found in the given base directory then it will need to be sent to the originator of the request. If it is not found then a search will have to be performed to first find and store the file locally, then send it to the originator of the request. Each incoming request should spawn a new thread or process to handle it. Examples of doing this in C and Java are on the course website.
- Requests for a file that is found locally should result in the file being sent to the requesting server. The file is sent as a sequence of transmissions, formatted as:

data length (2 bytes)	file data (0-65000 bytes)
-----------------------	---------------------------

That is, the file is sent in at most 65000 byte chunks, so while small files will be in one packet, large files will require many packets. A data length of 0 indicates the end of the file data sequence. Your program should handle any file data and total length. Once the file is completely sent, the sending process/thread should sleep for 1 second before exiting.

- Requests for files not found locally will result in searches. For each other server in the command line list in turn, send a file request as described in 2. If the file is returned (as described in 4) then it should be first stored locally (ie fully received) as a temporary file, then sent as a normal local request. Note that since this is a non-caching file server the temporary file is not retained after sending.

If a server does not have the file and cannot find the file from other servers it responds with: 

65535 (2 bytes)
-----------------

Your program should be launched as:

```
prog -d basedir -p port [serverip serverport]*
```

where `basedir` is the directory to use for local file storage, `port` is a port number, and the `serverip serverport` pairs is a variable length list of IP's (eg "132.206.50.4") and port numbers for other servers to search.

In order to test your application you will need 4 instances of your application running, each with its own separate local repository of files. In the servers' directories create files as described in figure 1; the test files are available on the course website.

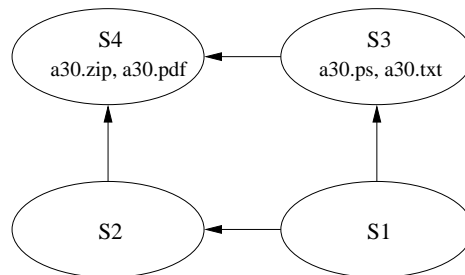


Figure 1:

Launch servers s2–s4 with command lines specifying other servers as indicated by the arrows in the figure. s3 and s4 should be run on machines with different architectures (eg s3 on Sun and s4 on Intel). Once all other servers have been started, server s1 should be started (with an appropriate command line); s1 will then start a test by concurrently searching for all the given files plus a request for a non-existent file, e.g. `.dat`.

19

Hand in your code on paper and electronically as normal. Make sure your code compiles and executes properly on the teaching system (mimi, willy etc).

Output from the servers should also be included; this should consist of lines describing each message received or sent (but not actual file data; eg, "FILE:somefilename", "sent 65000 bytes to 132.206.51.3:54030", "received 52303 bytes from 132.206.51.3:34001").

Total marks:

25