Test Case: 1

Summary: Verify that the ExternalServer (simple file server) is able to handle a client request from the same machine, and serve requested file if it exists.

Requirement: ExternalServer must be able to accept a client request, and reply by sending the file requested. The file must be received by the client and become available in said client's file directory.

Prerequisites:

1. ExternalServer has file to be requested in its directory
2. A simple file receipt client that follows the same expected message send/receive procedure as the ExternalServer

Procedure:

1. Execute ExternalServer using port 6900
2. Execute Client using localhost as IP, port 6900 as remote port, and test.docx as the fileName

Expected Result: test.docx appears in the client directory

Actual Result: test.docx appeared in the client directory

Status: success

Remarks: Executing this test case multiple times while continuing to refine the procedure for communicating with a client

Test Case: 2

Summary: Verify that the ExternalServer (simple file server) is able to handle multiple clients in succession

Requirement: ExternalServer must continue to run after serving a client, and continue to serve subsequent clients in the same manner

Prerequisites:

1. ExternalServer has file to be requested in its directory
2. A simple file receipt client that follows the same expected message send/receive procedure as the ExternalServer present in 3 separate directories

Procedure:

1. Execute ExternalServer using port 6900
2. Execute Client in the first client directory using localhost as IP, port 6900 as remote port, and test.docx as the fileName
3. Execute Client in the second client directory using localhost as IP, port 6900 as remote port, and test.docx as the fileName
4. Execute Client in the third client directory using localhost as IP, port 6900 as remote port, and test.docx as the fileName

Expected Result: test.docx appears in all three of the client directories

Actual Result: test.docx appeared in all three of the client directory

Status: success

Remarks: This test case was important in verifying that the ExternalServer was able to run continuously and serve many clients in succession. This proves valuable for developing the other components as we know the ExternalServer will continue to operate while we modify the other components.

Test Case: 3

Summary: Verify client is able to receive and save file from external file server

Requirement: Client must be able to receive a file and save it in its file directory

Prerequisites:

1. File to be requested is not present in client's directory
2. ExternalServer is running on the same machine on port 6900
3. ExternalServer's directory contains test.docx

Procedure:

1. Execute Client using localhost:6900 and request test.docx

Expected Result: test.docx appears in Client's directory

Actual Result: test.docx appeared in Client's directory

Status: success

Remarks: This test was used to confirm that the Client could save a file received from a server.

Test Case: 4

Summary: Verify Server is able to act as a proxy between Client and ExternalServer. In this case it will forward a String request from client to ExternalServer, and forward the expected replies from ExternalServer back to the Client.

Requirement: The cache server must handle Client requests, forward them to ExternalServer, and forward reply back to Client

Prerequisites:

1. ExternalServer is running on same machine on port 6900
2. ExternalServer has test.docx in its directory
3. Client's directory does not have test.docx

Procedure:

1. Execute Server using localhost:6900 as the address for which it will act as a proxy
2. Execute Client using localhost:6968 (cache Server runs on port 6968) with test.docx as requested file

Expected Result: test.docx appears in Client's directory

Actual Result: test.docx appeared in Client's directory

Status: success

Remarks: This test was useful for verifying that the cache server is able to act as an intermediary between a client and a server

Test Case: 5

Summary: Verify that Server is able to use CacheManager to store request/IP pairs in the cache by having Server reply to any secondary clients with the address of the Client that has previously requested the same file

Requirement: Server must store any requests it forwards to ExternalServer along with the requester's IP in a cache

Prerequisites:

1. ExternalServer is running on same machine on port 6900
2. ExternalServer's directory contains test.Docx

Procedure:

1. Execute Server using localhost:6900 as the address for which it will act as a proxy
2. Execute first Client using localhost:6968 (cache server runs on port 6968) with test.docx as requested file
3. Execute second Client using localhost:6968 with test.docx as requested file

Expected Result: ExternalServer only receives one request. Second Client receives address of Client that previously requested file

Actual Result: ExternalServer only received one request. Second Client received "127.0.0.1"

Status: success

Remarks: This test was useful because it shows that the Server is able to store requests in the cache and behave differently depending on if a received request is present in the cache

Test Case: 6

Summary: Verify Client is able to host file after receiving it from ExternalServer through the cache Server and serve new Clients that are sent by the cache server

Requirement: After receiving file from ExternalServer via cache server, the Client must host the file and serve Clients that request the file.

Prerequisites:

1. ExternalServer is running on same machine on port 6900
2. ExternalServer's directory contains test.docx
3. Test.docx is absent from all Client directories

Procedure:

1. Execute Server using localhost:6900 as the address for which it will act as a proxy
2. Execute first Client using localhost:6968(cache server runs on port 6968) with test.docx as requested file
3. Execute second Client using localhost:6968 with test.docx as requested file.

Expected Result: ExternalServer only receives one request. First Client receives request directly from second Client. Test.docx is saved in second Client's directory

Actual Result: ExternalServer only received one request. First Client received a request directly from second Client. Test.docx was saved in second Client's directory

Status: success

Remarks: this test was very valuable because it verified the very basic functionality of the solution.

Test Case: 7

Summary: Verify that upon Client shutdown, if that client was hosting a file, it's request/IP pair are removed from cache on server

Requirement: If a hosting Client shutdowns for any reason, it must notify the cache Server to remove its information from the cache.

Prerequisites:

1. ExternalServer is running on same machine on port 6900
2. ExternalServer's directory contains test.docx
3. Test.docx is absent from all Client directories

Procedure:

1. Execute Server using localhost:6900 as the address for which it will act as a proxy
2. Execute first Client using localhost:6968(cache server runs on port 6968) with test.docx as requested file
3. Execute second Client using localhost:6968 with test.docx as requested file.
4. Terminate first Client
5. Execute third Client using localhost:6968 with test.docx as requested file.

Expected Result: Second Client will be handled by hosting client. Third Client will be forwarded to ExternalServer instead of being handled by the (now shutdown) hosting Client

Actual Result: Second Client was be handled by hosting client. Third Client was be forwarded to ExternalServer instead of being handled by the (now shutdown) hosting Client

Status: success

Remarks: This test was crucial in handling the case where a hosting client unexpectedly shuts down and preventing the cache server from continuing to send new clients to that address.

Test Case: 8

Summary: Verify that a hosting Client will only handle 5 requests then shutdown

Requirement: A hosting Client will handle a limited number of requests then shutdown

Prerequisites:

1. ExternalServer is running on same machine on port 6900
2. ExternalServer's directory contains test.docx
3. Test.docx is absent from all Client directories

Procedure:

1. Execute Server using localhost:6900 as the address for which it will act as a proxy
2. Execute first Client using localhost:6968(cache server runs on port 6968) with test.docx as requested file
3. Execute second Client using localhost:6968 with test.docx as requested file.
4. Execute second Client again using localhost:6968 with test.docx as requested file.
5. Execute second Client again using localhost:6968 with test.docx as requested file.
6. Execute second Client again using localhost:6968 with test.docx as requested file.
7. (fifth request to client) Execute second Client again using localhost:6968 with test.docx as requested file.
8. Execute second Client again using localhost:6968 with test.docx as requested file.

Expected Result: upon receiving the 5th request, the hosting client will wait for all threads to join, and shutdown. The next request is forwarded to ExternalServer.

Actual Result: upon receiving the 5th request, the hosting client waited for all threads to join, and shutdown. The next request was forwarded to ExternalServer.

Status: success

Remarks: This test verified that the number of clients any individual hosting-client will serve can be limited

Test Case: 9

Summary: Verify that after a set expiry time (5 minutes) an item in the cache will be removed

Requirement: A cached request/IP pair must expire after a predetermined length of time and remove itself from the cache.

Prerequisites:

1. ExternalServer is running on same machine on port 6900
2. ExternalServer's directory contains test.docx
3. Test.docx is absent from all Client directories

Procedure:

1. Execute Server using localhost:6900 as the address for which it will act as a proxy
2. Execute first Client using localhost:6968(cache server runs on port 6968) with test.docx as requested file
3. Wait for 5 minutes
4. Execute second Client using localhost:6968 with test.docx as requested file

Expected Result: When the second Client is executed, its request will be forwarded to ExternalServer because the first Client's info has been removed from cache

Actual Result: When the second Client was executed, its request was forwarded to ExternalServer because the first Client's info had been removed from cache

Status: success

Remarks: This is valuable because it allows us to keep the file being distributed relatively up to date by never allowing a client to download a file older than 5 minutes

Test Case: 10

Summary: Verify that after a set expiry time (5 minutes) if a hosting Client has not received a Client's request, it will shutdown.

Requirement: A hosting client must shut down after not receiving any requests for an extended period of time.

Prerequisites:

1. ExternalServer is running on same machine on port 6900
2. ExternalServer's directory contains test.docx
3. Test.docx is absent from all Client directories

Procedure:

1. Execute Server using localhost:6900 as the address for which it will act as a proxy
2. Execute first Client using localhost:6968(cache server runs on port 6968) with test.docx as requested file
3. Wait for 5 minutes

Expected Result: After the 5 minutes is up, the client will automatically shutdown

Actual Result: After 5 minutes, the client automatically shutdown

Status: success

Remarks: This is valuable because it prevents the client from hosting infinitely despite no more requests incoming.