**Problem**: Implement a parallel file transmission system. This is a client-server system where a file can be transferred from the server to the client. Once the client finishes, an intact copy of the file from the server should be on the client. Client will initiate the connection with the server and requests for the file. We wish to transfer the file as fast as possible, and you should assume that a single connection will not maximize the network bandwidth. Thus, we’d like to use multiple connections to transmit different parts of the file in parallel.

**System Design:**

The implementation is based on micro service architecture. Service Locator pattaren, Dependency injection, networking and immutability patterns wherever possible is followed.

Following are the services :

**File Server Service** – This service provides multiple offsets (start and end combination) to read files in parallel, this service also serves the file. It is a TCP service. Multiple instances of this can be executed. They can be load balanced based on RR by service broker. Each prominent class of this service is extensible.

**Service Broker Service** – This service provides infrastructure to register a service. It can register multiple instances of file server service and any other service, it provides connect string of instance based on RR load balancing. It is a TCP service. As of now it is single instance but can be clustered and load balanced.

**Client Service** – This service downloads the files from File Server service. It first gets the connect string of load balanced File Server Service from Service Broker Service. Then connects to File Server Service to get multiple offsets (start and end combination) to download files in parallel from File Server Service. Client Service downloads files in chunks and store it in download/temp/<filename> folder. Client Service has Merge manager module which merges all chunk files sitting in download/temp/<filename> folder in correct order, saves complete file in download/ folder and purges the chunk files sitting in download/temp/<filename>. This service is standalone module but can be changed or extended to be based on TCP service. It is provided as a library so that anyone can use it as module in the application. There is driver class written to run this service. It is possible to run multiple instances of this service.

**Config Handler-** This interface has been written with intent to extend capability to manage configuration of each service.

**Request and Response Flow:**

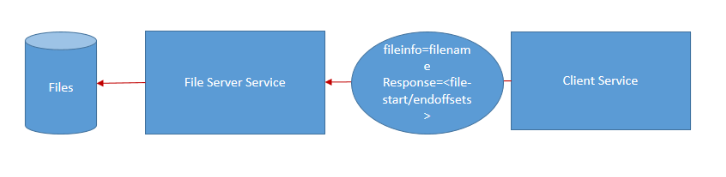
**File Server Service Registers itself to Service Broker:**



**Client Service discovers connect string of File Server Service from Service Broker**

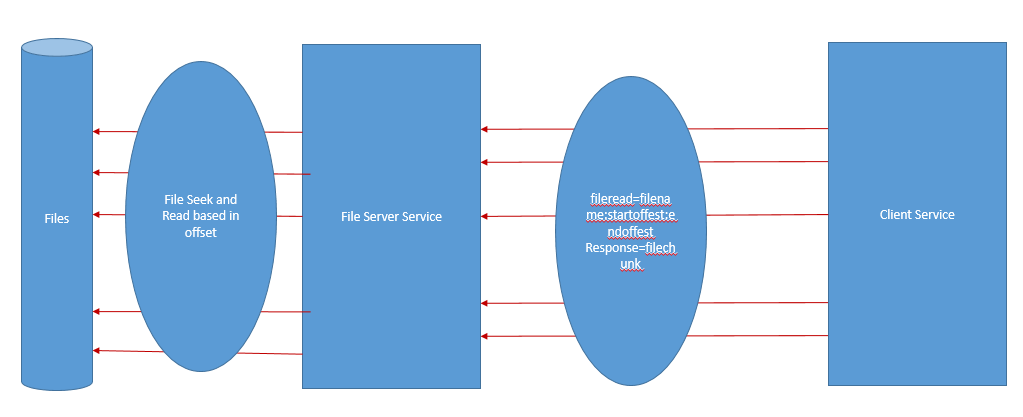


**Client Service asks File Service to send file offsets information**



**Client again discovers file server service from broker** 

**Client Service download files in chunks in parallel socket**



**Deliverables**-

**Source Code** – src folder includes complete code

**Binaries Folder** – Jar files of file server service(fs.jar), service broker(sb.jar), client service (client.jar), test file, startup.bat, properties file

**Setup.bat** – Run this file to run all services. Following explains each input of this batch file:

start java -jar sb.jar <Location of directory of sb.properties, this property file is explained below>

SLEEP 1

start java -jar fs.jar <Location of directory of fs.properties, this property file is explained below>

SLEEP 5

start java -jar client.jar <Location of directory of client.properties, this property file is explained below> <name of file you want to download e.g."test.txt">

SLEEP 1

**Property files-**

**File Server Service property file – fs.properties**

port=<PORT of File Server Service>

clientprocessingthreadpool=<maximum parallel thread pool size for client socket processing, increase it based on load>

servicebroker=<service broker service host:port>

filedir=<Location of Files>

filechunksize=<File Chunk Size used in case client does not enforce the chunk size>

datapacketsize=<Data packet size which is sent back to client on write on socket, if packet size is less that file chunk size then that size of data is sent on wire at a time else file chunk size of data is sent>

**Client Service Property file – client.properties**

servicebroker=<service broker service host:port>

writefiledir=<Location where files will be written on client side after download>

retry=<Number of times client should re-try to get file chunk>

filechunksize=<File chunk size which clients tells File Server Service and ask it to send packets in that size>

**Service Broker Property file – sb.properties**

port=<PORT of service broker service>

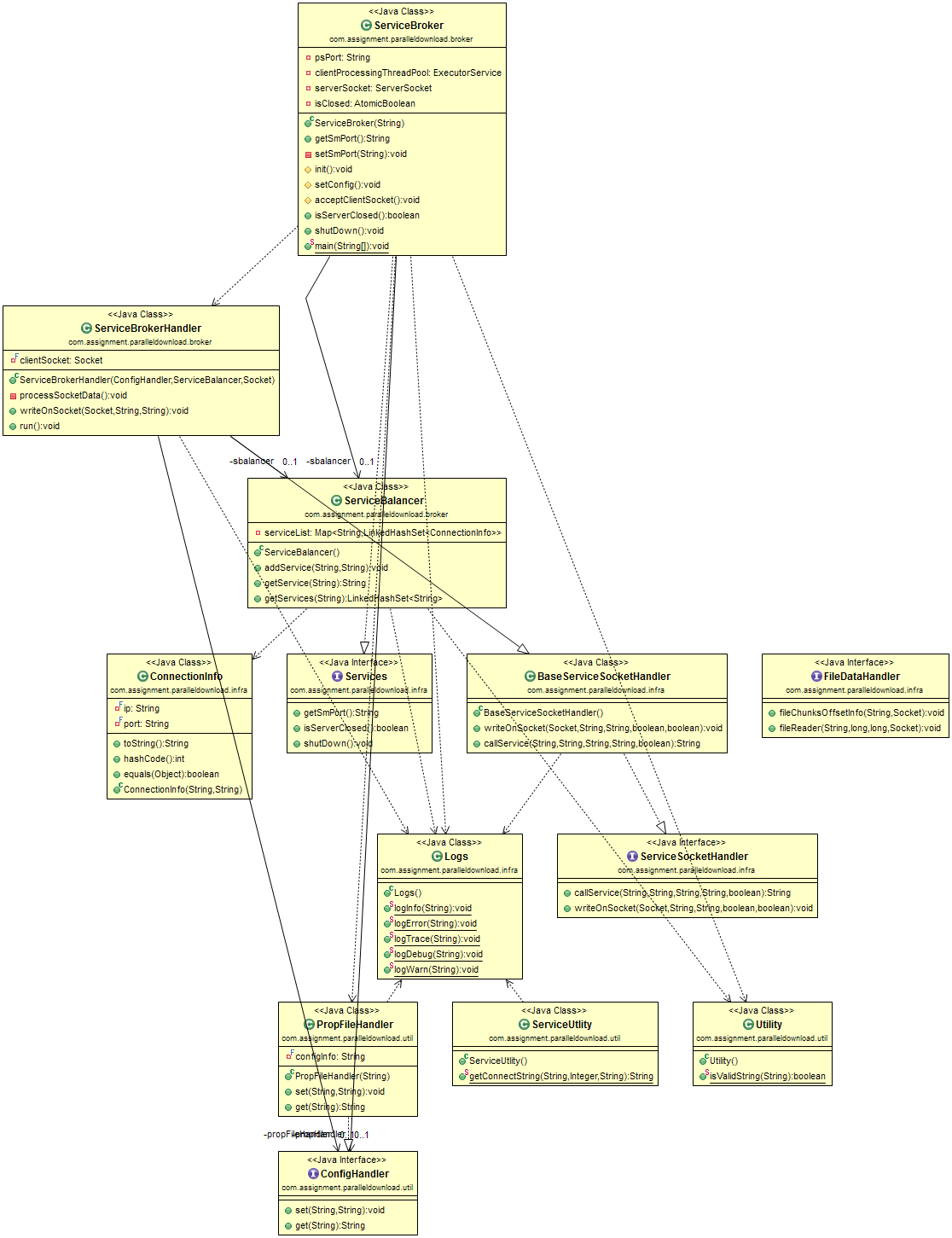
clientprocessingthreadpool=< maximum parallel thread pool size for client socket processing, increase it based on load >

**Extensible Design-**

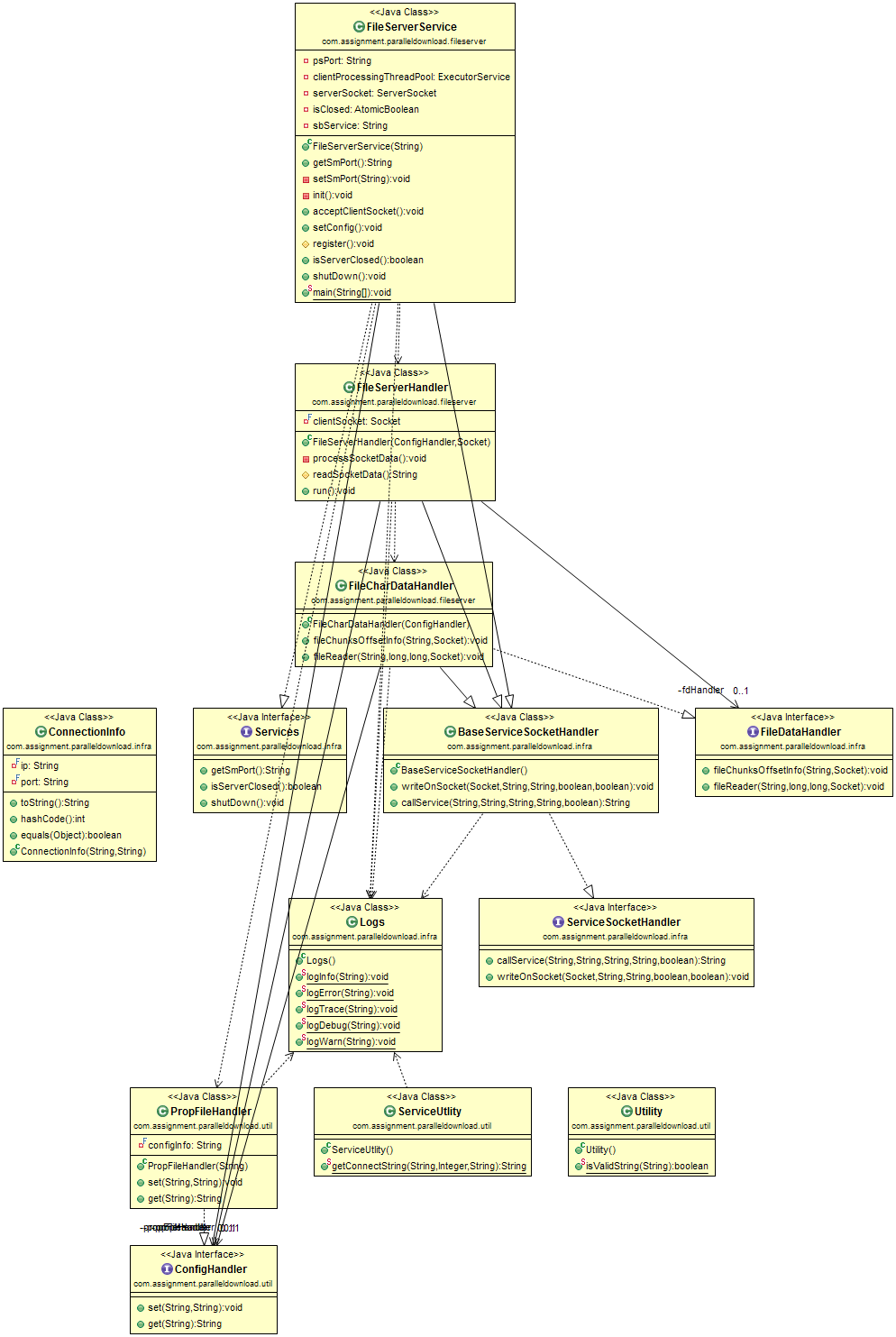
Each class has been written with extensibility in mind.

**Class Diagram-**

**Service Broker Service:**



**File Server Service**



**Client Service**

