1. File Transfer Protocol
   1. Description of Existing Application

FTPClient requests FTPServer for a list of available files and then sends a file download request to the server. The server sends the requested file in small chunks to the client.

* 1. Current Design

Figure 1 shows an overview of the current architecture for this system. It only contains two main classes i.e., FTPClient, FTPServer and three protocol messages FileTransferRequet, FileTransferResponse and FileTransferAck. Both the client and server run as separate processes, and maybe even on separate machines. The UML Sequence Diagram in Figure 2 shows this client-server interaction in more details.

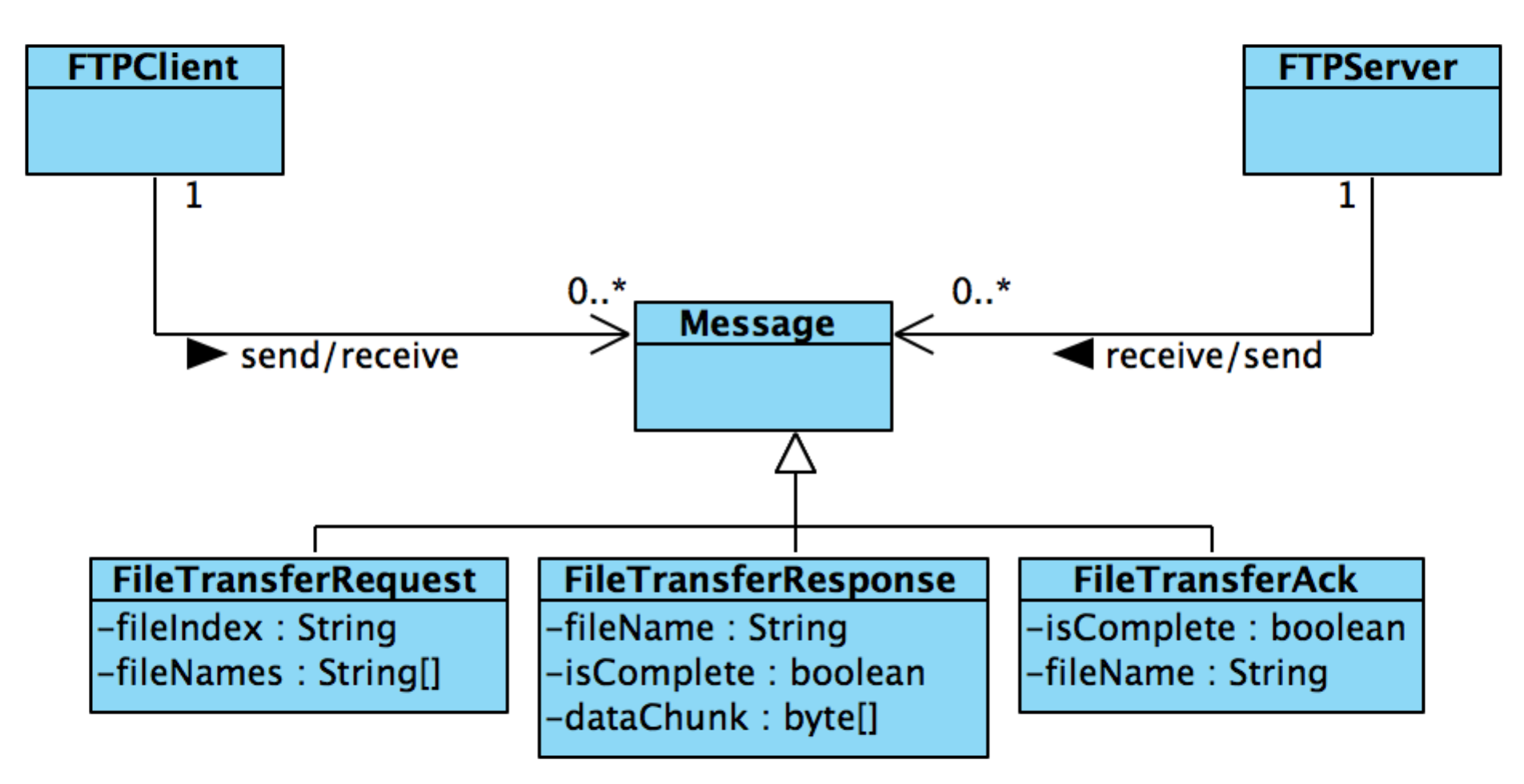


Figure 1: Architecture Diagram for FTP

FTPClient communicates with the FTPServer and establishes a TCP connection. The client sends a FileTransferRequest to the server to ask for the list of available files on the server. FTPServer sends back the list of available file names, encapsulated in FileTransferRequest. FTPClient then allows the user to enter the selected file index, using console input. Then it creates an instance of FileTransferRequest, encapsulated with selected file index, and sends to the server. FTPServer receives the request, and starts transferring the selected file contents in fixed-length data chunks, encapsulated in FileTransferResponse. Once the file has been successfully transferred, client sends an acknowledgement message, FileTransferAck, to the FTPServer. FTPClient process automatically opens the file after successful transfer and terminates itself. FTPServer terminates itself after the file has been transferred successfully and has received an acknowledgement.

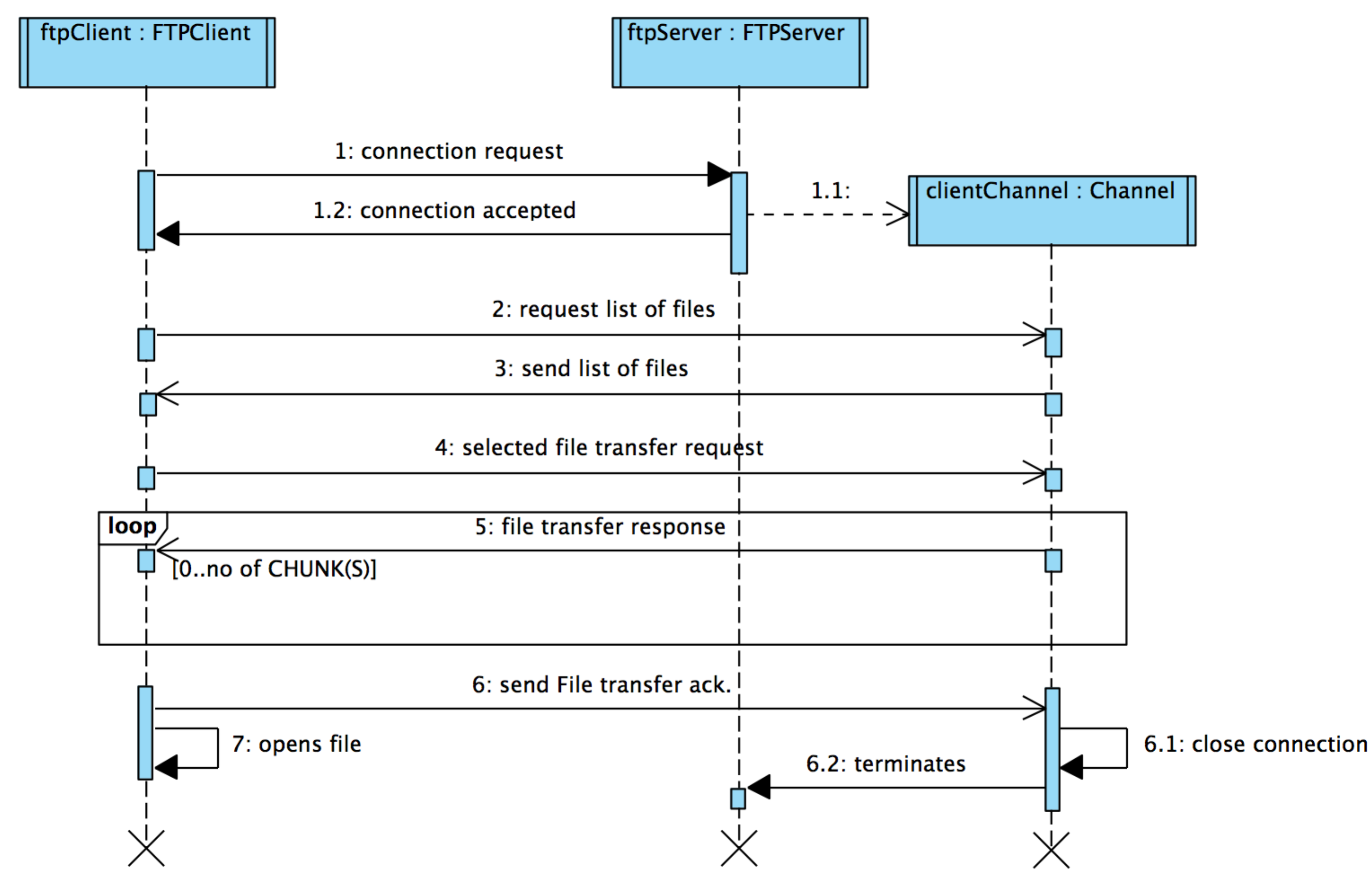


Figure 2: Interaction diagram between FTPClient and FTPServer

Note, that the interaction is asynchronous from both the client and server perspective. In other words, both the client and server does not block while waiting for a protocol message.

1. Measuring Performance
   1. Introduction

It measures some performance related statistics for message-based communications between a sender and receiver.

* 1. Description
* The extension logs following performance related statistics:
  + Total numbers of conversations, which occurred in the system where a conversation can be defined with any combinations of, sends or receives. Different types of conversations are one-way send, one-way receive, request-reply and multi-step conversations
  + Total time for all conversations
  + Average turnaround time for a request to be processed where average turn-around time is the average of a timespan from conversation start time to conversation end time
  + Maximum turnaround time for any conversations
  + Minimum turnaround time for any conversation
* The program logs the time when a conversation starts
* It logs and calculates the above statistics when the conversation ends
* Note that a conversation can be a simple request-reply type exchange of messages or a complex combination of send and receive events. We are defining the conversations for sample applications as follow:
  + *Levenshtein Edit-Distance Calculator:* A conversation is when a client sends a request and receives a response from the calculator
  + *File Transfer Protocol:* A conversation is when a client sends a request for a file download and when it receives the last response of data chunk for that file from the server
  + *Weather Station Simulator:* A conversation is when a receiver sends a request to get weather related data readings and receives the first response from the transmitter
  1. Application Material
* Developers would be provided with the following classes:
* *Stats:* A data structure containing elements to measure performance
* *PerformanceMeasure:* It logs performance measure using sliding window