Appendix B: Selected sample applications

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.

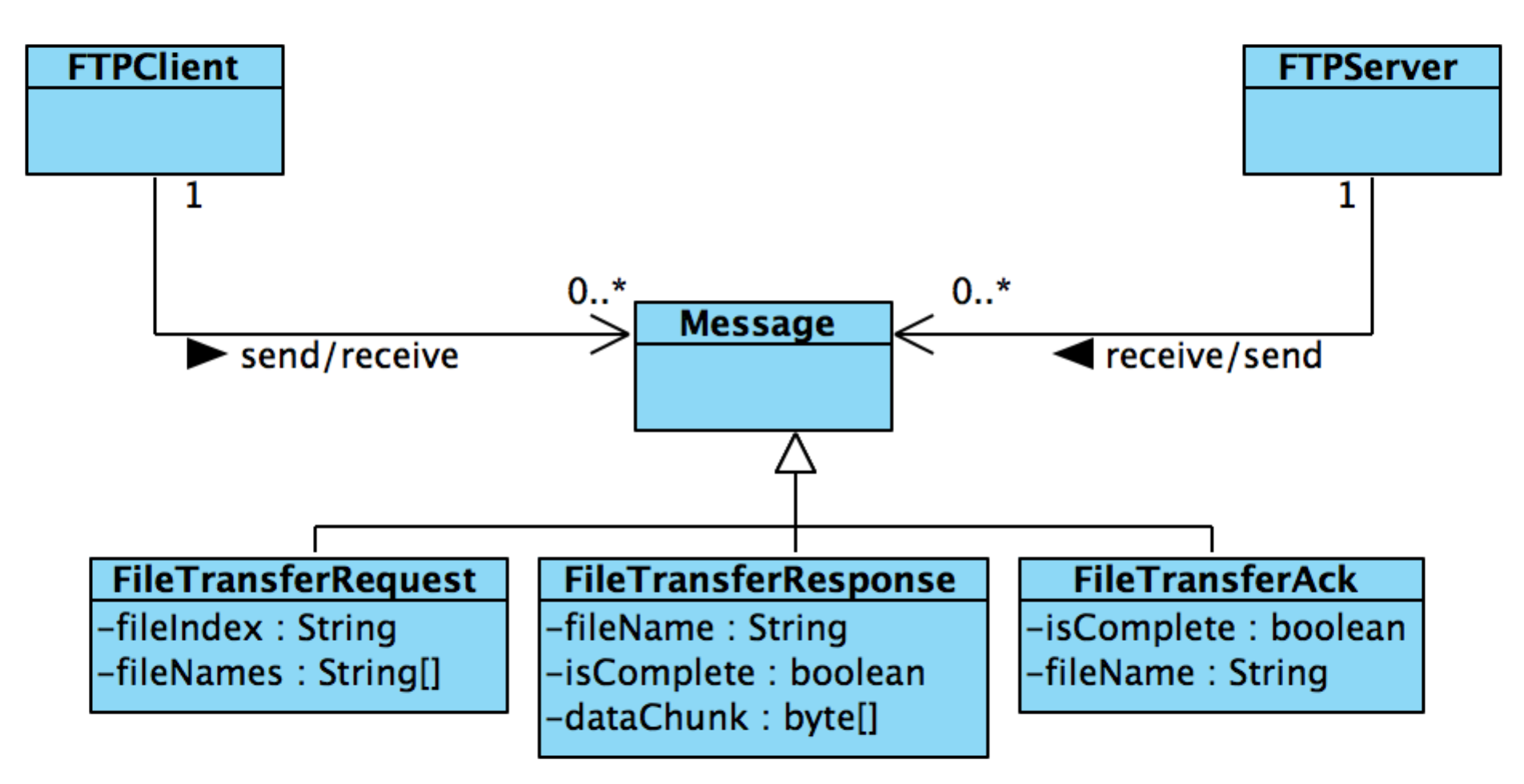


Figure 1: Architecture Diagram for FTP

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.

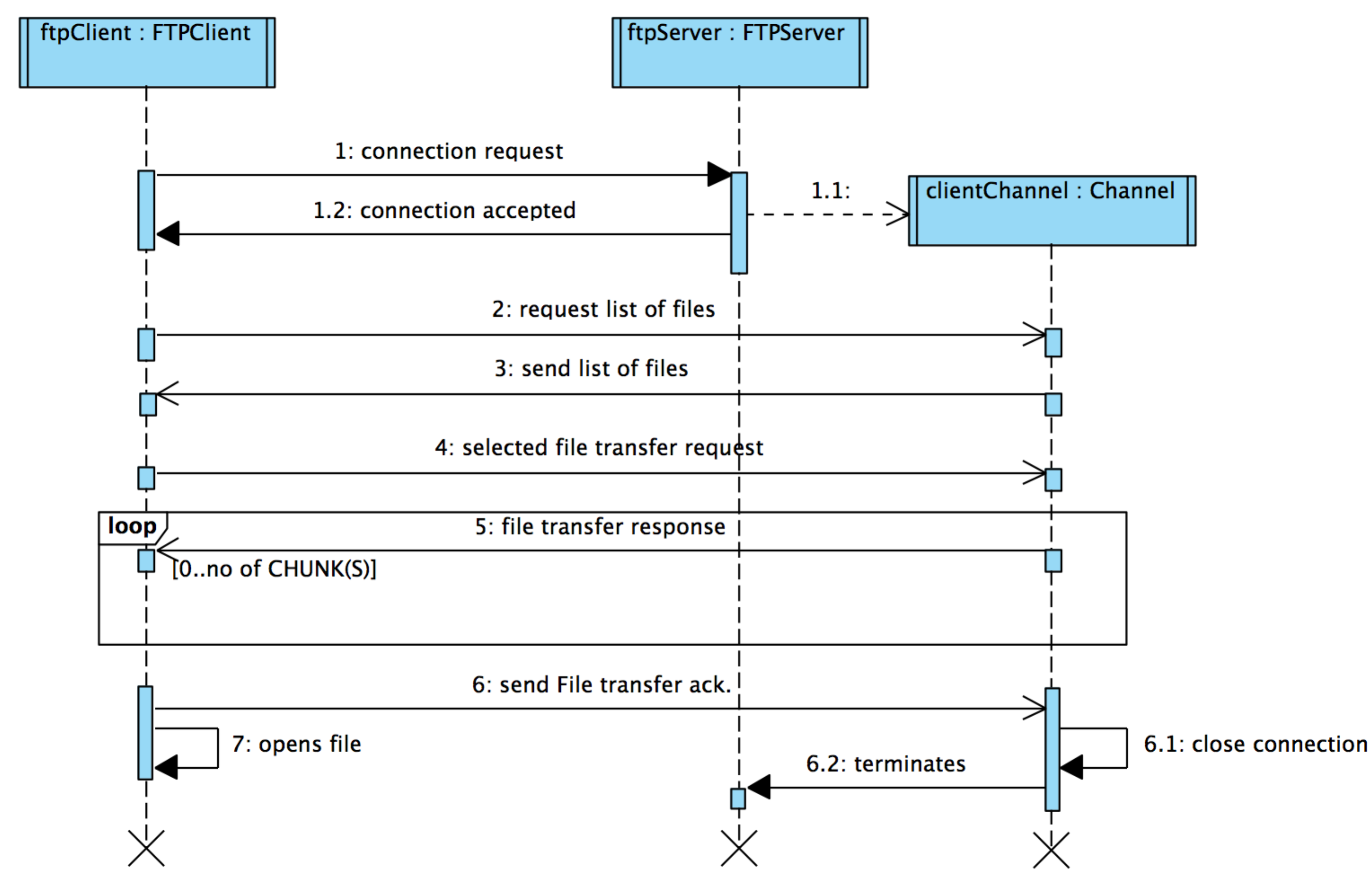


Figure 2: Interaction diagram between FTPClient and FTPServer

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.

Appendix C: Selected inter-process Extensions

1. Version Compatibility
   1. Introduction

This extension adapts one version of the message to another, so processes running different versions can still communicate with each other.

* 1. Description
* Each application process knows its version number.
  + Client will be using Version 1
  + Server will be using Version 0
* Each message contains that version number.
* Before sending the message to receiver, it always converts the message to its application version at sender side.
  + On Client Side, aspect will convert Client ver1 messages to Server ver. 0 message
  + On Server Side, aspect will convert Server ver. 0 messages to Client ver. 1 messages
* After receiving the message, it always ensures that the received message is matched with the application version at receiver side.
  + Aspects on both sides will just check that the correct message has been received.