FTP Server Tutorial

FTP

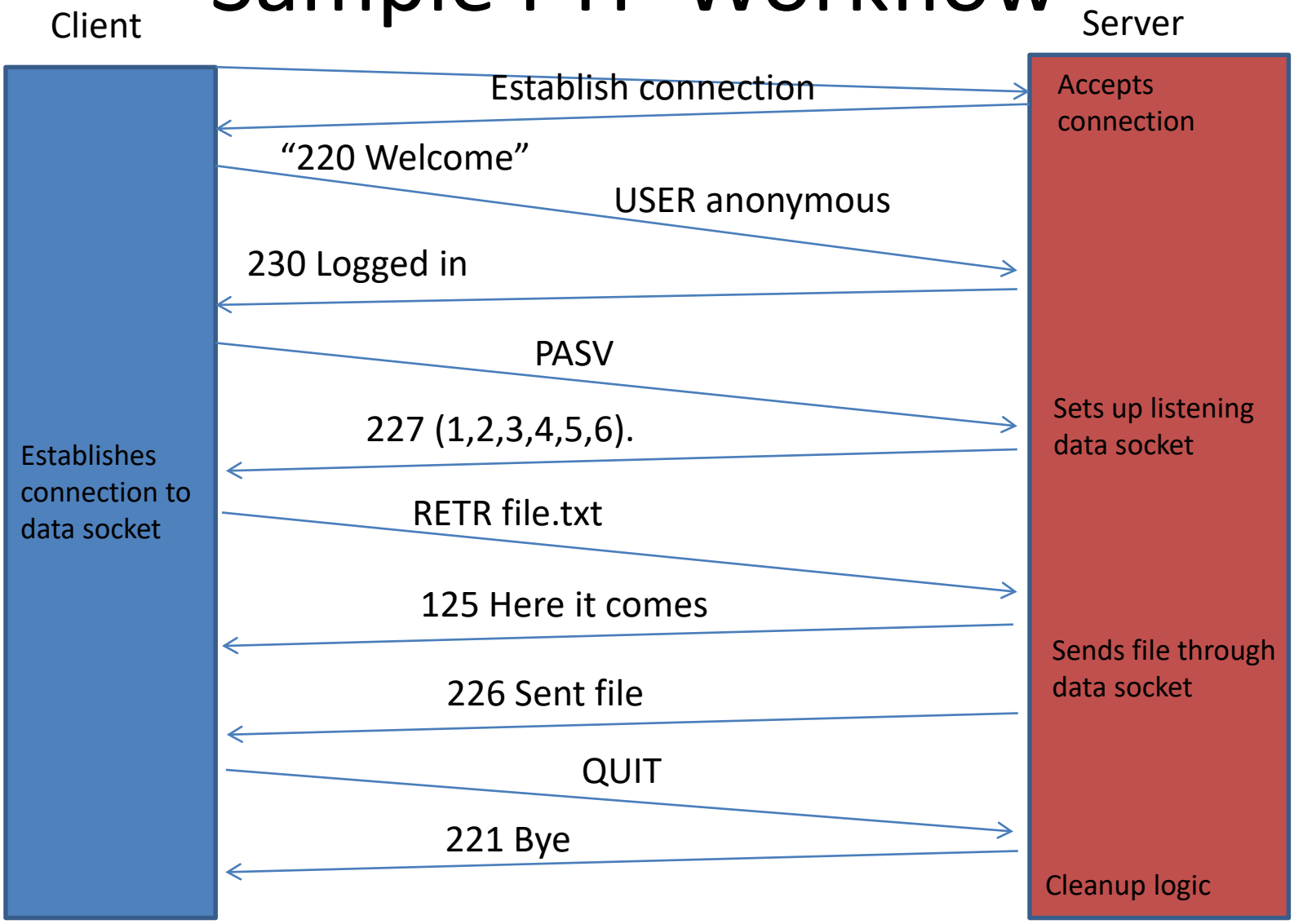
- File transfer protocol
- Allows a client to transfer files from a server rooted at some directory
- In A3 you will make a server able to send files from its root directory

Application Protocols

After connecting:

- Send one or more messages to get one or more responses
- Server loops on its own socket and waits for input
- Messages change state of server based on protocol
- Certain states elicit a response from server
- Client then applies application logic on response

Sample FTP Workflow



Last tutorial you saw how to set up the server and manage the sockets.

Parser Commands from Client

- Format of an FTP response:
  - <command> <arg1> <arg2> … <argn>
  - Space delimited
  - First token is command
  - Each of the next n tokens are arguments

1) Tokenize user input
2) Check that user input is valid (syntactic check)
3) Run command with user input (semantic check – break if arguments don't make sense)

Useful String functions (may be more)

- Strlen
  - String length
- Strtok
  - Tokenize string
- Strcmp
  - String equality
- Strcasecmp
  - Case insensitive string equality
- Strcpy
  - Copy string
- Sprintf
  - Copy formatted string

Strtok

- char* strtok(char* str, char* delimiters)
  - #include “string.h”
  - Str = string to tokenize
  - Delimiters = **characters** to split by (not substrings)
  - It’s a split function, but does splitting one step at a time
  - Each split is done on the next sequence of 1 or more of any arbitrary combination of delimiters
  - Eg. strtok(“**hell**o”, “hlef”) produces “o”

Strtok

- First call loads the string and does the first split
- Subsequent calls expect str=NULL to further split the string
  - If change delimiters, then you change what you split by in the next iteration
  - If you change str, it resets the string
- If there's nothing left to split, returns NULL
- **Warning**: This function mutates your original string

Sprintf

- int sprintf(char* str, char* format, …)
  - #include “stdio.h”
  - This is essentially just printf
  - “…” represent the values that go into the formatted string
  - You write the output into str (a char buffer), rather than standard output (console)
  - Reminder: int printf(char* format, …)

Good Warmup Exercise

Implement a function that takes in user input until “quit” is entered (case insensitive), afterwhich it returns out of the function

For non-quit inputs, tokenize by the space character, print out each token, and print out the # of tokens

```
Enter string: token1 token2 token3 token four
token1
token2
token3
token
four
5
Enter string:
```

Important Note

Strings you read from the socket “probably” end with “\n” and/or “\r”

Make sure you strip them off before you do anything with the string