Network Sockets 101

Warren Christian

SEL Intern

Initial Goal

- Goal is to build out an FTP client application

- Broke it down into 3 parts
  - Network logic
    - Network sockets
    - Send-receive packets
  - Parsing data
  - UI/UX

Sockets

- RFC 147 coined the term 'Socket' in the context of computer networks
- First implemented in ARPANET, which heavily relies on socket architecture
- UC Berkeley created its own socket API, called the BSD sockets API in C
- Became open-sourced in 1989
- Almost all networks are designed with some variation of sockets, whether they are abstracted or not

Socket client within BSD paradigm

1. **socket()** - attaches IP/hostname to be used later, along with other specificities (UDP, TCP, etc)
2. **connect()** - socket information is passed in, then the client attempts to connect to node
3. **send(), recv(), read(), write()** - used to translate data along network socket pipeline.
4. **close(), shutdown()** - kills connection

Sidenote oversimplification: Main difference between server and client, is that a server "listens" for traffic, and receives packets

Swift socket architecture

- Every framework has a BSD/POSIX implementation "behind the curtains"
- Differences in design lie in how this is abstracted away
- Swift all but requires concurrency (ie. multi-threading)

Weighing the options

- Began thinking about forking pre-existing remote file system applications. Network sockets were not on the radar at first
- It became more apparent that sockets would become a core part of the inner-workings of an FTP application
- Came across native (Apple) and third-party socket implementations.
- Found that there are tons of options with many flavors for every kind of use-case!
- Narrowed it down to a handful

- **BlueSocket** – Open source IBM framework
  - Lacks concurrency by design, but can be implemented later
  - Implementation closely resembles BSD socket architecture
  - Deprecated, but functional
- **NWConnection** – Apple networking implementation
  - One of Apple's network frameworks
  - Lots of modularity for developers
  - Can be used in conjunction with other frameworks because of native support
- **SwiftNIO** – Apple networking framework
  - Concurrent by default
  - Obfuscates socket manipulation, beginner friendly
  - (It's the best)
- **GoldRaccoon** - An FTP-specific framework
  - Focuses on FTP usage
  - Long deprecated
  - Relies on objective-c

Learnings as an intern

- There are many paths to solving a challenge
- Often there is no clear "correct" solution, but one that is balanced between many factors
  - Factors such as time, skills, complexity, cost
- Priorities can change as you are working
  - Something can appear to be a feasible solution, but when implementing it, you find that there are unforeseen shortcomings
    - GoldRaccoon!
- Fun challenges can send you down rabbit-holes
  - Pushes the envelope on your current skills, highlights new knowledge areas, and you gain an appreciation for expertise in those domains
  - Compartmentalized knowledge is very important in large organizations