Simple TFTP Server

The goal is to build a simple TFTP server that only stores files in memory. There's no need to write files out to the filesystem.

For more information on the TFTP protocol refer to the RFC:
http://tools.ietf.org/html/rfc1350

The server can bind a port other than 69 but should be implemented using UDP regardless. Only the "octet" mode needs to be implemented. Requests should be handled concurrently, but files being written to the server must not be visible until completed.

Ground Rules

1. The goal is to have functional code that can be the basis of a conversation around design trade-offs during the face-to-face interview process.

2. Expect to take several hours, but this is not a timed assignment. The most important goal is making a functional piece of code you would want to be reviewed by a colleague.

3. Include all source code, test code, and a README describing how to build and run the solution.

4. We think of languages as tools and good engineers can adapt to the right tool for the job. As we primarily program in golang at Igneous (and it happens to be a great language for this type of project), please submit the code in golang.
Hint: OSX and (most) Linux distros include a tftp client that can be used for ad-hoc testing.

Additional Focus - Operations

To provide operational focus, there is one more requirement: aggregate a request log of all requests to the server. There should be one line for each request to read a file, each request to write a file, and also for each unrecognized request. Write the request log to a separate file so that it is not interspersed with other log output from your in-memory TFTP server solution.

Getting Started

A stubbed out main method and a packet parsing library is supplied to you as a starting point to help you ramp up with some basic golang syntax, golang tools, and testing patterns. Beyond that, it's your choice to determine how the solution should be structured and implemented.