1. Structure your networking code

2. Rapid development with netcat

3. Simulate real world with Network Link Conditioner

Single Responsibility Principle

Just because you can, doesn’t mean you should

BAAAAAD!

```
@implementation ViewController
...
// Called when user taps 'enter' on the on-screen keyboard
- (BOOL)textFieldShouldReturn:(UITextField *)textField {

  // Write data to output stream.
  const char *data = [textField.text
                      cStringUsingEncoding:NSASCIIStringEncoding];
  NSInteger b = [outputStream write:data
                           maxLength:[textField.text length]];

  // Check for errors.
  if (bytesWritten == -1)  {
    [self disconnect];
  }

  [self displayMessage:textField.text];
  [textField setText:@""];
  return YES;
}
...
@end
```

Good good good!

```
@implementation ViewController
...
// Called when user taps 'enter' on the on-screen keyboard
-(BOOL)textFieldShouldReturn:(UITextField *)textField {

  // Networking code nicely encapsulated
  [[NetworkController sharedInstance] sendMessage:textField.text];

  [self displayMessage:textField.text];
  [textField setText:@""];
  return YES;
}
...
@end
```

NetworkController

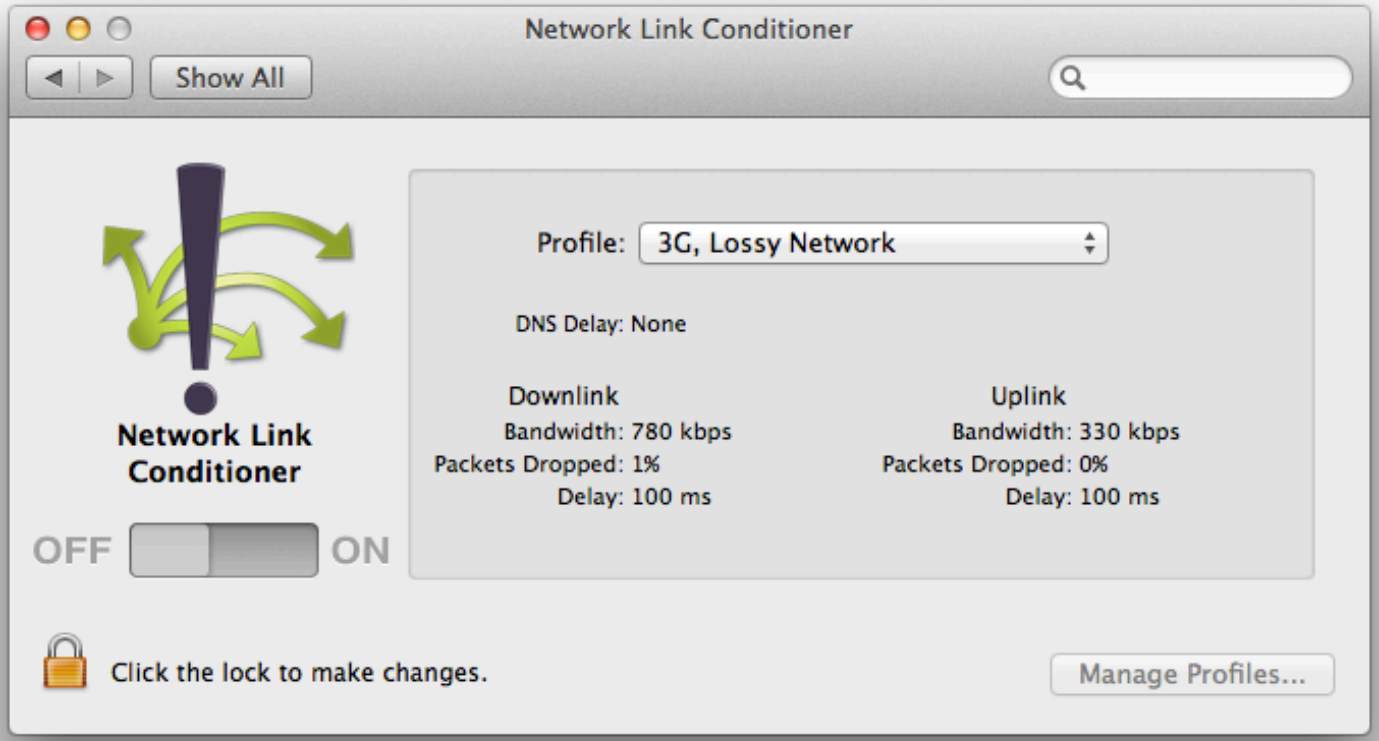
Sockets
Streams
Buffers
Event handling
Protocol

netcat

is awesome

Start server:

```
nc -k -l <port>
```



/Developer/Applications/Utilities/Network Link Conditioner

Summary

1. Encapsulate networking code.
2. Use ‘nc’ to rapidly develop and test.
3. Test on slow network before releasing.

github.com/byteclub/SimpleSocketConnection

@byteclub