Lab1Milestone3

Jump to bottom

sethnielson edited this page on 23 Oct · 1 revision

Lab 1 Milestone 3: Error-Correcting Delivery

Assigned	10/21/2019
Due	10/28/2019
Points	75

Overview

This is where the rubber meets the road! This is where you take your reliable layer and make it reliable. Your layer needs to make sure that data sent from one end of the playground network *actually* arrives at the other end.

Once this layer is ready, playground servers such as the bank will be switched over to using it and the playground switches will be set up to introduce errors into the data.

Requirements

You are responsible for developing the protocol. But the protocol must meet the following requirements:

1. You must be able to transmit and receive data over Playground switches that have 3 errors per 102400 bytes. Some bytes will cause packets to be completely dropped.
2. You must also be able to transmit and receive data out of order. Playground switches will reorder about 1 in every 100 packets.
3. You must be able to close a session. That is, if either side calls `transport.close` that should trigger a close on the other side as well.

Please make sure you understand the guarantee to the application layer. From the application layer's perspective, all data sent between `connection_made` and either `transport.close` or `connection_lost` should be guaranteed to arrive so long as the error rate is not above a threshold and the other side is still alive. Think carefully about shutdown because if you're not careful data you send may not be delivered.

Optional Components

The following are not requirements but are either helpful to you or worth extra credit

1. Consider implementing *optional* features. There are almost always things you can do with *your* code that isn't in the specification that is still compatible with everyone. There are even ways you can put things in your handshake to recognize your own code and then do optional additions that are only compatible with your own implementation. If you can't see how to do this, let me know and I'll show you.
2. The fastest implementation gets extra credit. This will be measured as the average throughput under various error conditions as measured against all other teams.
3. There is also extra credit for good performance above the 3 errors in 102400 bytes threshhold
4. Please remember that you have to run your code over this protocol. So try not to make it too terribly slow...

Writing the PRFC and Standardizing by the Due Date

Each team will be taking their own approach to this. At any time you can start to propose to other teams how you think the class should do the assignment. But you will be more persuasive if you have a working prototype. As you get your lab working, you need to discuss over slack how you think the PRFC should be written and what should be the standard. Unlike the Escape Room exercise, this will involve protocol design and not just packets.

The PETF can vote, at any time, on any proposal. BUT the handshake must be standardized and the whole class conforming to it by the due date (10/28/2019).

Grading

On the due date, I will post an auto-grader on the SAFE network that conforms to the proposed PRFC. It's going to do some "interesting" logic to test with error conditions. I will provide you with the auto-grade client and you will not need to write it yourself. But it will require that the connector for your network layer is installed in your own personal playground with the name assigned by the PETF.

There will also be a special autograder mode that will be for testing the fastest implementation. We will try out a few trial runs and then test the real thing in class.

The address and port of the autograder will be:

```
20194.0.0.19100 and port 19101
```

The auto-grader will test error-recovery delivery in both directions

- 25 points for data delivery and shutdown with 1 error per 102400 bytes
- 25 points for data delivery and shutdown with 2 errors per 102400 bytes
- 25 points for data delivery and shutdown with 3 errors per 102400 bytes
- 25 extra credit points for data delivery and shutdown with 6 errors per 102400 bytes
- 25 extra credit points for data delivery and shutdown with 10 errors per 102400 bytes
- 50 extra credit points for the fastest implementation

Clone this wiki locally

https://github.com/CrimsonVista/20194NetworkSecurity.wiki.git