# Fault model for HTTP/3 and related network protocols

15-300, Fall 2021

Shannon Ding

November 19, 2021

## 1 Project Description

I will be working with Professor Eunsuk Kang from the Institute for Software Research to formally describe the HTTP/3 protocol. The HTTP protocol is used in the World Wide Web which has applications in online banking, sensitive data transfer, and communication in general. The next iteration of the protocol is HTTP/3, which will improve performance and privacy. HTTP/3 has been an Internet Draft since 2018, which means there is a paper describing how to implement the protocol by building on HTTP/2. It recently (in August 2021) entered the Request for Comments phase where the protocol is more rigorously defined. Already, many libraries (Python, Node) and network infrastructure services (Cloudflare) have implemented support for HTTP/3. Since so many types of devices use HTTP, there are many implementations of HTTP/3 being put into use. Thus we must ensure their correctness and safety before they are deployed and interconnected on such a widespread level.

The main gap in the field we hope to resolve is the lack of thorough analysis over a wide range of implementations. There are more papers about how to implement and protocol performance than correctness. Furthermore, while most papers test the protocol against common inputs, we will be targeting the protocol's response toward unexpected inputs.

We will begin by first looking at TCP, an established and widely studied protocol. As an exercise, I'll start by defining an abstract state machine for TCP using a to-be-decided descriptive language. Afterward, we'll run the abstract state machine through a set of scenarios to catch any contradictions or errors. Finally, we'll use the abstract state machine definition to generate a set of tests to run on actual implementations of TCP.

To further split this task up, we will be analyzing components of TCP at a time before analyzing the protocol as a whole to see how different components act.

We will repeat this process for HTTP/3, although it will take longer to formally define an abstract state machine as there is less literature on the subject. For HTTP/3 in particular, we will attempt to generate a set of representative unexpected inputs and test for an HTTP/3 implementation crashing.

The primary challenge is defining the abstract state machine, as it requires thorough reading of the protocol specification and very exact behavior.

## 2 Project Goals

### 2.1 75% Project Goal

- Read TCP specification and code an abstract state machine for TCP
- Generate set of tests using TCP abstract state machine and run them on several implementations

- Identify a reasonably self contained component of HTTP/3 and code an abstract state machine for the component. Log crash data.

### 2.2 100% Project Goal

- Split HTTP/3 protocol into several components and code an abstract state machine for each one
- Generate tests for each component and run them separately on several implementations. Log crash data.

### 2.3 125% Project Goal

- Code abstract state machine for behaviors that involve more than one component of HTTP/3
- Generate tests for the HTTP/3 protocol as a whole and run them on several implementations
- Identify the nature of unexpected inputs that cause crashes. Log crash and performance data.

## 3 Project Milestones

### 3.1 First Technical Milestone

Finish reading background literature on TCP. Mathematically write out the abstract state machine for TCP.

### 3.2 First Biweekly Milestone

Mostly done with coding abstract state machine for TCP.

### 3.3 Second Biweekly Milestone

Complete TCP abstract state machine. Begin generating tests from abstract state machine.

### 3.4 Third Biweekly Milestone

Run generated test suite on several implementations of TCP. Read background on HTTP/3 and identify a component of HTTP/3 to first tackle.

### 3.5 Fourth Biweekly Milestone

Complete the code for an abstract state machine of one component of HTTP/3. Generate tests and successfully run on several implementations of HTTP/3. Collect crash data.

### 3.6 Fifth Biweekly Milestone

Identify and begin coding an abstract state machine for the next component of HTTP/3. Generate tests, run on implementations, and collect crash data.

### 3.7 Sixth Biweekly Milestone

Identify and begin coding an abstract state machine for the next component of HTTP/3. Generate tests, run on implementations, and collect crash data.

### 3.8 Seventh Biweekly Milestone

Synthesize results and come to conclusion on the types of unexpected inputs causing crashes.

## 4 Literature Search

I will need to identify papers that define abstract state machines for TCP. This research was primarily inspired by this paper [1], but rather than testing on common inputs, we will be testing robustness by providing unexpected inputs. I'll also be concurrently taking Bug Catching: Automated Program Verification, which is in line with this research topic.

## 5 Resources Needed

The majority of the work, defining an abstract state machine, can be done on a laptop using some kind of formal verification language. Testing may require computer cluster use.

## References

[1] Kenneth L. McMillan and Lenore D. Zuck. 2019. Formal specification and testing of QUIC. In Proceedings of the ACM Special Interest Group on Data Communication (SIGCOMM '19). Association for Computing Machinery, New York, NY, USA, 227–240. DOI:https://doi.org/10.114