Esteban Roncancio
Capstone Abstract
CSCI 1760

Packet Firewall

The goal of this capstone project was to design a packet filtering system with two main functions: access control and filtering. Access control was determined by configuration packets that both whitelisted communication between two address and outright blacklisted addresses from sending any packets. Filtering was applied to accepted data packets by maintaining a histogram of the checksum of the body. Throughout the semester, the project evolved from a nominal single-threaded serial firewall to a high performant, scalable, concurrent design.

An important aspect to consider was to ensure the data structures were linearizable. I opted to use Java's ConcurrentHashMap (wait-free implementation) to keep track of the blacklisted source addresses and to map concurrent skip lists (to denote ranges of permitted source addresses) to destination addresses. For the SkipList, I implemented the LazySkipList algorithm outlined in *Art of Multiprocessor Programming* by Prof. Herlihy. This provided me with a wait-free contains() method. Finally, I opted to use a concurrent Cache from Google's Guava library, which allowed me for faster decision making for recurring packets between repeated addresses.

By using wait-free and lock-free data structures, the processing of data packets was very fast. The thread model consisted of 3 pools to handle different tasks of the system. Majority of threads were in charge of reading the cache and maps to make quick decision. Since configuration packets needed to be handled atomically, one thread handled those and because filtering need not be linearizable multiple threads grabbed the bodies that were ready for processing from an AtomicQueue.