**Evaluation of evidence**

Because the effectiveness of DDoS attacks depend on the deterrence of server performance, data can be gathered and measured by observing the effectiveness the server has in regards to client queries. If we have server access, we can measure performance based on the CPU power and CPU cycles being consumed by the server experiencing a DDoS attack. A server undergoing a DDoS attack would experience a sharp increase in CPU processing power being sacrificed to handle large amounts of queries as the server has to spend more time handling an increasingly high scale of requests to fulfill.

However, in order to measure CPU power, this would require a process to be constantly running in the background occasionally requesting the server to provide a report of CPU usage. This process requires processing time and power from the server which might not be reliably provided when the server is undergoing stress from a DDoS attack.

An alternative method of measurement would be to monitor the network performance shared by the server during a DDoS attack. This way, an additional system can be used to monitor the influx of requests and outgoing responses to and from the server. We can measure network performance by measuring the Round-Trip-Time or time it takes to send a packet and receive a reply to a particular client. If a server is too busy to process a request, the RTT time increases until the server can respond. If we have access to the network where the server is being hosted on, we can measure the influx of packets being sent through the network. The graphs below are an example showing the difference in network performance of a network at rest and undergoing minor stress under the network.

The following two graphs record the performance of a network at rest. The data recorded was obtained by using Wireshark and observing the ethernet port being used to connect to the internet. The RTT is extremely volatile because the network occasionally experiences downtime where no packets are being sent which need to be acknowledged. As such, the RTT time will occasionally drop to 0 when there is no other client to respond to.

Unfortunately, actual data from my simulated network could not be obtained for the time being because the simulated network environment is still being developed.

Chart, line chart

Description automatically generated

Timeline

Description automatically generated

The following two graphs below show the network performance where the network is experiencing Larger than usual workload due to online queries. Note that the number of packets being sent and the is much larger than a network at rest, and the RTT time is more consistent due to the network machine constantly having to send requests.

A picture containing diagram

Description automatically generated

Chart, line chart

Description automatically generated

**Hypothesis**

When handling a sudden influx of incoming requests, the challenge of handling a DDoS attack depends on the scale and resource power from the hardware of the server and the scale of the attack initiated by an attacker. The server faces a challenge of figuring out how to serve all possible users which becomes increasingly difficult the more complex these requests can become. If a server is unable to mitigate or identify a DDoS attack, my proposed solution would be a divide-and-conquer approach using a round-robin queueing system to stress handle a DDoS attack.

What makes this attack different from popular method of load-balancing is that this method would be using techniques from how Operating Systems handle large amounts of tasks for the CPU. Operating Systems use a preemptive scheduling solution to ensure that every tasks gets their share in the CPU on top of a priority system to make sure the most important tasks gets priority. Using this solution, we can use a scheduling algorithm for our servers so that if a server or router becomes unable to handle the large amounts of traffic, it will divide the work into groups and assign priority based on the types of requests being made. Slow, resource-intensive solutions could be given lower priority so that, while it would be inconvenient, these requests wouldn’t be troublesome for the server. Meanwhile, important tasks, such as security checks and network interfacing from an admin, would be given higher priority, allowing users with special privileges to remain in control of the situation. The system would work under a round robin system so that all users, regardless of priority, could still get a chance for server access rather than be blocked out due to a server experiencing overbearing capacity. While it may not stop a DDoS attack, it would allow the server’s security policy to remain in effect even under heavy stress.