Verifying The DDoS Resiliency Of Content Centric Networks

Jack Wilburn and Leo Kell and Dalton Clift

1. Introduction

After reading "Networking Named Content" [1] by Jacobson et al. for homework 4 in this class, we were intrigued to see how the network would perform while mitigating some of the attacks that it theoretically can. The main attack vectors discussed in the homework questions were a bad actor sending malicious data to an endpoint or trying to overwhelm the endpoint with a DDoS attack. We chose to focus on the DDoS resiliency of the network and created a python implementation of CCN to let us experiment. We used SimPy [2] for our simulation and generated a network to see how it would behave with different initial conditions and in different scenarios. We compared a network where each node had a large cache size, a small cache size, and one that mimics IP routing. Our comparison showed that a singular server with a piece of information would be shielded from a DDoS attack by the caches that are closer to the clients and that in IP, there is absolutely no shielding. Our findings support the theoretical resiliency shown in the Jacobson paper, that "Data-based distributed denial of service (DDoS) attacks are simply not possible" when cache sizes are large enough.

2. Related Work

The CCN (Content-Centric Network) protocol is a new network protocol designed as to improve traditional IP networking. The concept was first introduced in 2014 in a paper by Van Jacobson et al. [1] and at an accompanying Google tech talk conference. It has not seen widespread adoption yet, but there may be more adoption on the horizon, with web 3.0 becoming a more widely talked about paradigm.

In traditional IP routing, users must look up the entity they'd like to connect to and then send a packet to that address. Since it's unknown to other nodes on the network what data that entity has and what is being requested, they must forward all traffic to the entity. This leaves nodes on the network susceptible to being overloaded, especially by bad actors. It also adds significant latency to the content being retrieved since users have to wait first for a DNS resolution.

In contrast, CCNs are more resilient to DDoS since they develop a content store of data cached at locations close to the end-users. Therefore, even a distributed attack should be mitigated by nodes on the network close to end-users, and the traffic shouldn't make its way to the server. Additionally, CCNs don't need to do any name resolution since they track search terms and the closest entity that has them.

CCNs also better reflect how people actually use the internet. In today's internet, millions of users request specific data, and they don't particularly care where it comes from or how it gets there; they just want it, and quickly. Having caches closer to end-users allows for extreme performance improvements for high demand content and inherently can reduce strain on the backbone of the internet by eliminating the distances content needs to travel to reach the end-user and allowing reduced load on end servers as every client will not need to communicate directly to the hosts.

CCN protocols revolve around two types of packets involved in a request/response communication. First the request packet, referred to as the Interest Packet. This contains the information that is requested, represented as a unique identifier. The response is known as the Data Packet, which contains both the information requested and additional data used to verify the integrity of the cache. As mentioned above, a key part of CCN nodes is the ability to cache content, and serve that data in response to future requests from any clients. When a computer requests data, it first reaches out to its router and from there the router will check its cache. If there is a cache miss, the node uses its FIB (forwarding information base) to locate the node containing the data it needs closest to its location. That router then goes directly to the node and retrieves that cached data.

Each node in the network keeps track of pending interests in a PIT (Pending Interest Table). In the PIT, each node tracks who has requested data that its already waiting to hear a response for, and it makes a note to forward the data onto that new requesting node. This means that for many client requests, a node needs to only send one packet upstream to get the data. Also, since each node has a PIT, the network uses the PITs as a trail back to the requester when the data is sent back downstream.

3. Threat Model

The threat model that we're considering for this analysis is a DDoS attack on a server node. The idea is that one server holds a unique piece of data, and we'll have many nodes on the network send requests for that packet. If we have a good implementation of a CCN, we expect that the server will be insulated from requests and that after the responses are cached close to the requesters, the server will see a minimal load.

We will contrast this against an IP-like network, where we use the same CCN implementation without the CCN features such as the content cache and the PIT. Disabling these should mean that each packet gets forwarded onto the

server and that it gets crushed, like in a traditional DDoS attack on an IP network.

4. Methodology

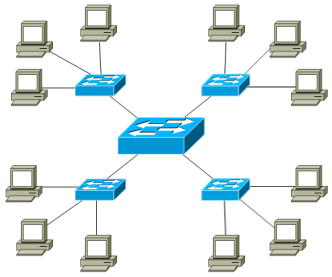

Figure 1. Star Network Topology [4]

The best way to evaluate the claims in the paper is to run a simulation because then we have to deal with the overhead of networking and handling multiple machines. The protocol lends itself well to discrete event simulation, where each event covers a node dealing with an incoming or outgoing packet in its event queue. We leveraged SimPy [2] to handle the event queues, yielding time delays between nodes, and coordinating the scheduling. We assigned a certain amount of time to each event to simulate the processing delay when handling the packets.

Since we're not using a physical model, our simulation can run on a singular machine quickly, which allowed us to iterate quickly and verify our results. We can also scale to different sizes and topologies. We can generate multiple types of topologies using our simulation code. We initially started with a line of routers connected to one node upstream and one downstream. We used this for our initial testing to ensure that we could send and receive packets between clients and servers. After experimenting with this topology, we generated a star network with a more typical network layout, as shown in figure 1. This allowed us to simulate a more normal looking network when it came to getting our final results.

In our simulation, clients send continuous requests with a configurable delay. This lets us simulate normal network loads or a DDoS by simply tuning the parameters. We ultimately chose to have them send continuous requests to simulate a DDoS scenario. After simulating these networks, we plotted our results using the python port of ggplot2, plotnine [3].

5. Implementation/Experimentation

Please find our code at https://github.com/lkell/cs6490-project.

5.1. Experiment

We developed a python model of a CCN network to simulate various scenarios. We wanted to compare a fully functional CCN network to a traditional IP based network under a DDoS load.

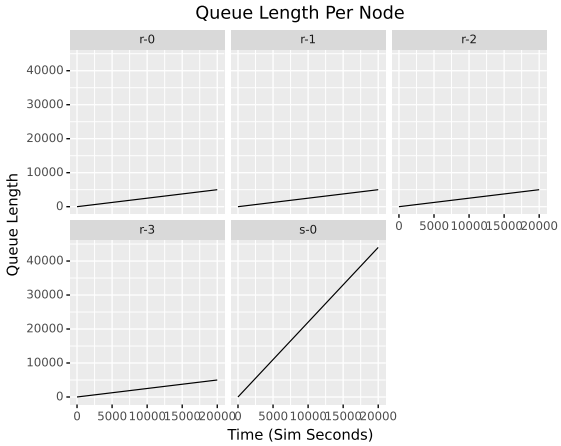

Figure 2. Network Node Load For IP-like Network

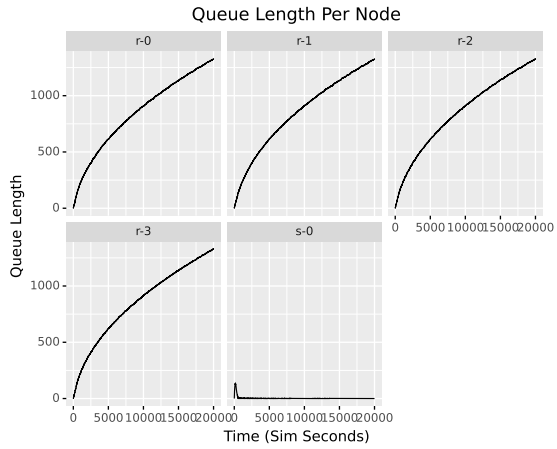

Figure 3. Network Node Load For CCN With Small Cache

Each network object is defined as an instance of a general Node python class. Each node contains a queue of packet requests that allow the node to address one request at each time-step of the simulation in FIFO order. To cache retrieved data, each router contains a dictionary of request paths and response data. The cache size is configurable in the simulation, and the LRU policy is used to evict data once the cache becomes full. To search for data not contained in its cache, each router forwarded requests to neighboring routers according to its routing table.

An important distinction between IP and CCN is the usage of PITs (pending interest tables) by CCN. Each node starts with an empty PIT, and as requests for certain data search paths come in, the node updates its PIT to track that a client has asked for a piece of information. As more requests for the same search path come to the node, we add the requester to the PIT and don't generate a new packet upstream. This reduces network load as requests are aggregated by intermediary nodes. In order to contrast this

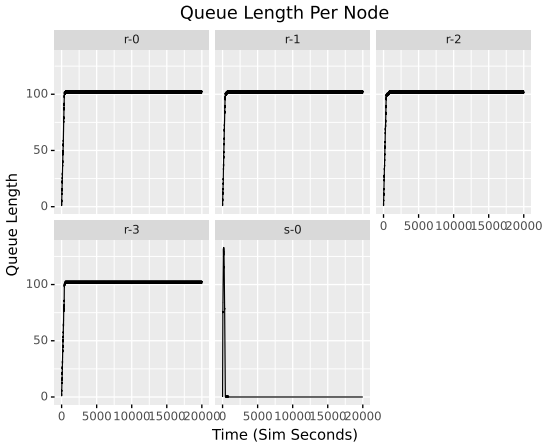

Figure 4. Network Node Load For CCN With Big Cache

model with a simple IP networking model, we included an option to bypass the PIT. This would cause the routers to re-forward every request they received, regardless of whether the content path was already contained in the PIT.

We also had to tackle another issue, how would nodes know where to send packets to get valid data responses. We decided that we would use a modified version of the Bellman-Ford/RIP (routing information protocol) algorithm where each node would broadcast its distance to the target, and every other node would update its distance accordingly. If a node updated its distance, it would rebroadcast until all nodes had the shortest path to the target data. We ran this step before simulating the requests and responses so that there wouldn't be any packets without a route.

Once the model network was initialized in python, different experiments were run for different router cache sizes. Each experiment was run for 20,000 simulated seconds, providing us with ample data for analysis. Within the experiment, each event (e.g., processing an incoming request packet) was configured to take up one second. Data requests from each client were sent out once for every three simulated seconds.

5.2. Results

We see from figures 2, 3, and 4 that in our CCN network we're able mitigate DDoS attacks very well, and that in general, across all tests, our network load is reduced. When our cache size is as large as the number of data items that exist on the network (as in figure 4), all responses are cached, and the router load never increases. The routers are not swapping out cache items and requesting from upstream while more requests are coming, they are simply able to respond with cached values. When we have a smaller cache (as in figure 3), we see that the load on the routers does increase, but slowly, as the first layer of routers has to ask the second layer for content that is cached there. When we have no cache and no PIT (as in figure 2), we see that the server is slammed and is unable to respond to all the requests that its receiving. There is also a high load on all other nodes as they're managing requests and responses traveling from the clients to the server. This would result in network disruptions, and the server experiencing a DDoS.

6. Conclusion

With our simulation, we were able to accurately model both an IP network and a CCN network and their behaviors during a DDoS attack. Using SimPy, we were able to handle the discrete events that each node was generating for requests and responses and we were able to show how the queue size grew over time under heavy load.

As shown by our results, CCNs are able to mitigate DDoS attacks effectively. However, CCN networks on their own are not necessarily adequate to completely remove the threat of DDoS attacks. This is indicated in Figure 3, where the router cache size is smaller than the total set of content that can be requested. Even though this particular simulation showed improved load reduction compared to simulations with smaller cache sizes and IP networks, there is still a clear growth in load over time that can be further limited by other DDoS prevention techniques such as rate-limiting. Note, however, that CCN's use of PIT acted as an effective load insulator for the centralized server, which saw minimal traffic after the caches were filled.

If we were to do any future work in this area, we could look at how to mitigate interest flooding attacks where a node on the network is constantly asking for similar but different content and overwhelming routers on the network as the nodes scramble to fulfill those requests. The suggestions in the paper are to rate-limit nodes and by having nodes ask downstream nodes to rate-limit interests. In this way, the network can be adaptive to load. It would be interesting to simulate this behavior to see if it would work as advertised and reduce network load for intermediary routers during times of heavy traffic. Additional future work could also involve experimenting with more complex and realistic network topologies, in addition to running the experiments on physical hardware capable of running the actual CCN protocol.

References

[1] Van Jacobson, Diana K. Smetters, James D. Thornton, Michael Plass, Nick Briggs, and Rebecca Braynard. 2012. Networking named content. Commun. ACM 55, 1 (January 2012), 117–124. https://doi.org/10.1145/2063176.2063204

[2] Ontje Lünsdorf and Stefan Scherfke, SimPy, (2002-2022), GitLab repository, https://gitlab.com/team-SimPy/SimPy/

[3] Hassan Kibirige, et al. (2021). has2k1/plotnine: v0.8.0 (v0.8.0). Zenodo. https://doi.org/10.5281/zenodo.4636791

[4] Kelson Lawrence, Kailin Acheson, and Delana Hallstedt. Back to the Basics: Networks and Topologies. https://blog.boson.com/bid/87993/Back-to-the-Basics-Networks-and-Topologies