Probability based, two-step caching in Named Data Network

Himanshu Mendhe
hpm7152@rit.edu

ABSTRACT

Named Data Networking is a novel communication system, which relies upon caching of content to provide data on request. The currently used caching systems like LRU, which has been studied here, tend to under-utilize the cache-capacity. A new system of caching has been proposed here, it is probability based, two-step caching and as the name suggests, consists of incorporating probability in decision making of caching. This project uses an open-source implementation of NDN simulation called NdnSim and modifies its source-code to test its concepts.

1. INTRODUCTION

Internet and its underlying communication network has been around for quite a long while now. Initially, the traffic on the internet used to be for achieving communication needs like chat, instant messaging and for document transfer. This need has also evolved to include video chat and virtual private networks for enterprises. However, in the past decade or so, the traffic on internet has moved towards satisfying the user's entertainment and social media needs. Video and audio streaming, online multi-player games are taking up larger portion of internet traffic. The companies which provide for these services, like Youtube, Netflix and Spotify, have data centers located at certain locations on earth and sometimes they are far away from the location of a user who intends to access content from these data centers. The distance, coupled with the 'best-effort' nature of internet, results in the users experiencing problems like increased latency, connection drops and request denial. The obvious solution to this would be a system where the required content is brought as close to the user as possible. This is where the Content Distribution Networks[6] (CDNs) comes into play. Traditionally, the CDNs have been implemented on the IP-address based networks. This approach has worked over the years, however, when the requirement is only of content, the IP address, which denotes location, should not be the metric on which network is based. This is why Named Data Network[9] (NDN), where the named-content is the primary identifier, is the basis of research in this paper.

It is very important in a CDN, that the required content be available to the end user as soon as the user requested for it. Caching, will go a long way in speeding up this content delivery process. While there has been a lot of research already done about caching in CDNs, there hasn't been much in the NDN domain which is the motivation behind the research described in this paper.

There are many algorithms available for caching[10], the most popular being Least Recently Used (LRU)[4], Most Recently Used (MRU), Random Replacement (RR), etc. In this paper, there will be a discussion of use of LRU with a NDN. This discussion will focus on the distribution of content in the NDN based upon the type of requests the content gets and the popularity of content. Along-with the aforementioned algorithms, over the years, probability and popularity have also been used as metric to determine which content to remove and which to keep. So, in this paper, a new method of cache replacement and caching in general with a probability-based, two-step caching process has been proposed.

2. SIMULATION OF NAMED DATA NETWORKING: NDNSIM

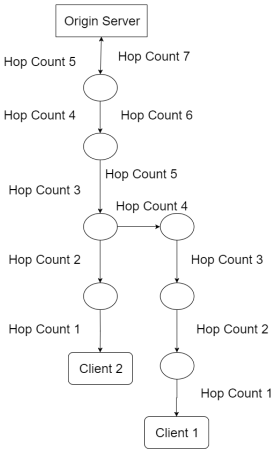
As mentioned above, the architecture of an NDN is based upon replacing IP address with content-name as primary identifier of nodes in a network. However, in today's world, almost all the network is IP-based. Thus, NDN will have to co-exist with the old IP-based networking. In other words, NDN runs as an overlay on the old IP-based networks. It, thus, became a requirement of this project to have NDN simulated. To achieve this, an open-source software called NdnSim[5] was used. NdnSim implements an overlay network over another open source software called NS3[8]. Both software have been written in C++ 11 and because of their open-source nature, one is able to modify their source-code to implement theoretical concepts and test them to see how they might work in real world.

This project has been implemented by modifying the source-code of NdnSim to change its caching policy. Description of the modifications made to the source-code has been given in section 5.

3. MOTIVATIONAL STUDY OF LRU

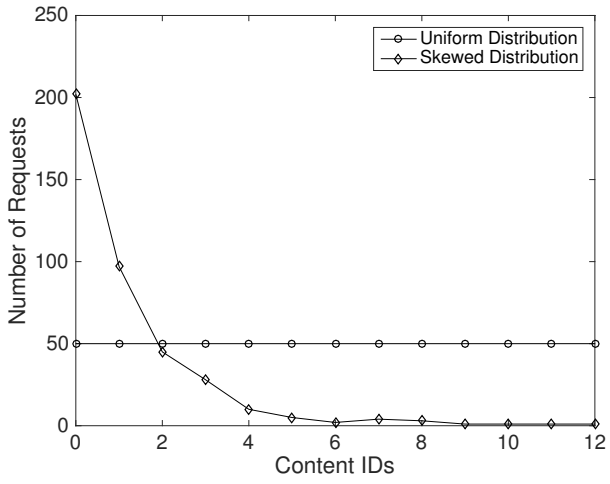
As mentioned above, LRU is one of the most widely used cache replacement policy in traditional IP-based networking. Thus it would also makes sense to have LRU implemented on a vast level in novel networks like NDN. Hence, performance of LRU in NDN was evaluated. The topology that was used for evaluating LRU was also used later to evaluate probability-based, two-level caching method. Figure 1 shows the said topology.

Figure 1: Topology: "Hop Counts" on right are for Client 1 while that on left are for Client 2. The ellipses denote cache servers



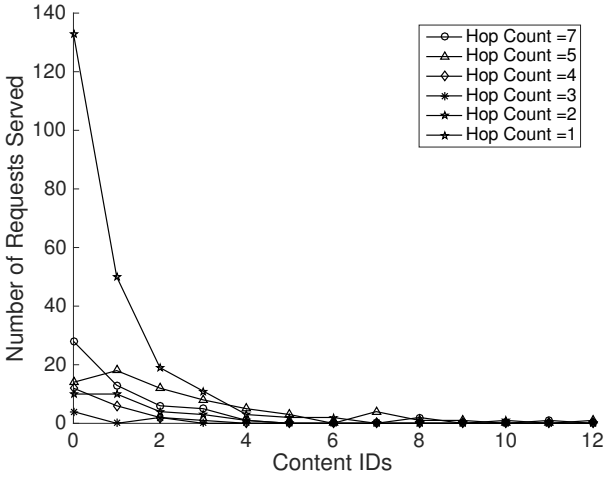
One of the most important thing about testing cache-replacement policy was to test it against popular and non-popular content. To make such traffic which simulate real-world traffic, the requests have to be random. The requests were for contents which had IDs ranging from 0 to 12. The distribution of these requests was skewed, i.e. content ID 0 was the most popular one while content-ID 12 was one of the least popular one. Figure 2 shows distribution of content-ID's frequency of requests. Log-normal distribution[1] was used to skew the distribution of random requests towards certain contents.

Figure 2: Distribution of requests: Lognormal and uniform



Requests with the aforementioned distribution, were sent over the topology shown in Figure 1. Following things were observed: 1. What kind of effect it has on the content-store of the cache servers? 2. How many times is a content stored in which cache server? 3. Which cache server is it served from? Figure 3 shows the number of requests and the hop-counts serving each requests. The numbers on the y-axis represent the number of requests for corresponding content-IDs. Example: content-ID 0 is being served around 135 times by hop-count 1, while around 24 times by hop-count 7. Similarly, content-ID 1 is being served around 50 times by hop-count 1.

Figure 3: Content IDs vs Hop count for LRU



From Figure 3, it can be observed, that most of the requests are being served, either from the edge servers or from the origin server and the middle servers are remaining unused. This was the motivation behind finding a method where utilization of the middle servers could be optimized for at least some requests which would result in better usage of cache capacity of the network.

4. PROPOSED CACHING POLICY

The LRU caching policy studied in the last section does not take into factor the popularity of the content. Neither does it care about the cache-capacity being wasted. For the proposed caching policy, both these factors play a crucial part. Another thing that plays an important role is the fact that not all cache servers should behave the same. Hence, in this project, certain cache servers will behave differently after certain amount of requests have been received by them. Such cache-servers will be called as "hub" for the purpose of this project.

Hubs behave differently from the other cache servers because they try to cache all the content they are receiving. This is not much different from what happens in the LRU. However, in cases where the cache servers have not become hubs, they are asked to calculate the probability and popularity of the contents arriving at them from the upper servers. The probability is calculated according to the *ProbCache*[7] formula. This calculation of probability has been proved to be

effective in the cited paper. Then, popularity is also calculated using the "Exponentially weighted moving average" *(EWMA)*[3]. Finally, the above two values are normalized and their product taken to come up with a value to determine if the data is to be cached or not. Algorithm 1 describes the aforementioned procedure succinctly.

Result: Cache or not
Content arrives from Origin Server;
Calculate probability;
Calculate popularity;
if *Normalized product of popularity and probability less than max* **then**
  cache data;
else
  don't cache data;
end

Algorithm 1: Algorithm to check if data to be cached or not

In cases where the content-store of the non-hub cache server is full, the decision of which content should be removed from the content-store and which content should be kept is also taken by looking at the normalized product of *ProbCache* and *EWMA*. The content which has the lowest value is removed.

5. IMPLEMENTATION

As the NdnSim's and NS3's source-code was written in C++, the implementation of this project was also done in C++. NdnSim's source-code was modified according to the requirements of this project. The main area of code-change revolved around NFD[5] which stands for Named Data Networking Forwarding Daemon. The NFD is where all the decisions regarding what is to be done when "Interest" (Requests) packets or "Data" packets arrive. Following is a list and description of the main classes which were modified or newly introduced in NdnSim's code:

- **Forwarder:** The forwarder uses the Pending Interest Table (PIT) to determine which interest needs its service. This class controls what happens when interest arrives, checks if the content, for which the interest is requesting data is present with the cache-server. If the content is present, the interest will be satisfied, else a copy of the interest will go back to the PIT while the interest itself will be forwarded to the higher level of node/server. The copy of interest will remain in the PIT until the interest is satisfied by the upper-levels of servers.

  The modification which were done to this class were to log certain counter values when an interest arrives. The code to determine if the cache-server is a "hub" or not was implemented in this class. Also added was the code which initiates the calculation of popularity and probability and the decision to store the incoming data or not.

- **ProbCacheExt Decision Policy:** A new class was created which is a subclass of "Decision Policy" class. The aforementioned class is not part of the core NdnSim code, but was available in an openly available extension of NdnSim from Github[2]. The ProbCacheExt class is called when a decision is to be made whether an incoming data is to be stored in the cache or not. This class calls code to calculate popularity and probability of the content in question to take that decision. The class has been called "ProbCacheExt" to differentiate it from "ProbCache"[7].

- **ProbCacheExtPolicy:** This class is a subclass of "Policy" class in NFD. This class is called to determine which content should be replaced from the cache. This class looks at the normalized product of *ProbCache* and *EWMA* to determine which content to remove.

- **ProbCacheCalc:** This class implements the actual methods which do the calculation of *ProbCache* and *EWMA* and its normalized product.

6. RESULTS

With the above mentioned changes in place, NdnSim's simulation was run on the topology shown in Figure 1. The performance metric into consideration were hit-ratio of the cache servers, and the hop-counts of the satisfied requests. Figure 4 shows the hit-ratio of servers for the two-level caching (this paper's proposed method) and LRU. As can be observed in the said figure the hit-ratio is higher for some servers for probability based, two-level caching while it is more for LRU in some cases. This is not giving a clear picture as to which system is better.

Figure 4: Cache hit-rate of servers

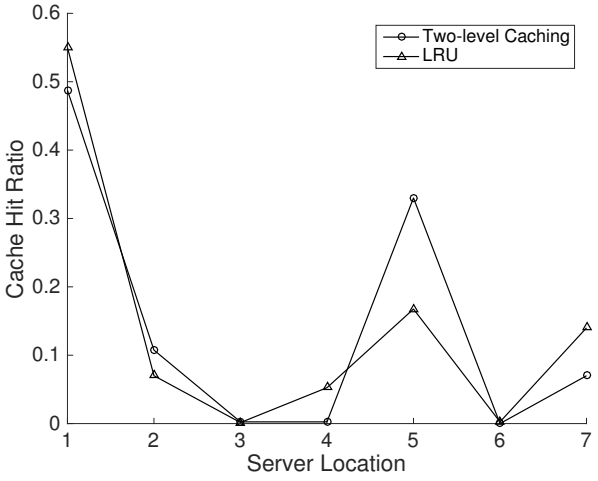
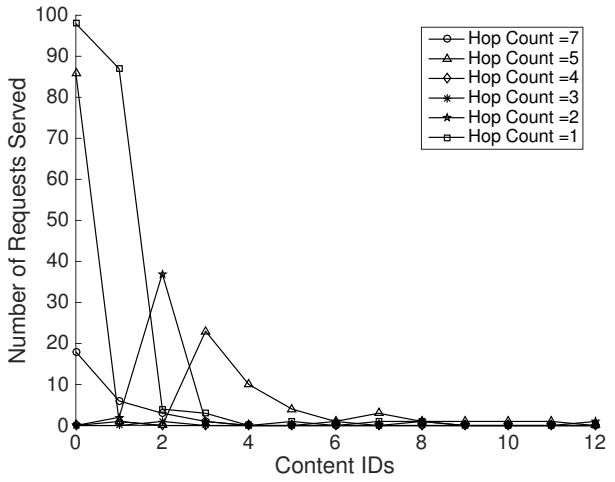


Figure 5 shows the hop-count of all the satisfied requests for corresponding content-IDs. This graph should be compared to the same graph which is present in Figure 3. This graph shows some increased activity in servers at higher-level of hierarchy compared to Figure 3, however this again, as in the Figure 4 doesn't show a conclusive improvement over LRU.

7. FUTURE WORK

As seen in the previous section, the results are such that a definitive conclusion cannot be drawn from it. This can be

Figure 5: Hop Counts of satisfied requests



improved upon. One place to start is to increase the scale of testing. Currently, 7 servers with around 100 requests per client were implemented in testing. This can be massively increased in a tree structure to get a better idea of the performance of the system.

In the experiments performed in this project, it was observed that to create 100 requests, one is required to create 100 clients which send only one request. This is because after sending one request for a content, the client ("Consumer", as NdnSim's developers like to call it), never sends a request for that particular content ever again. This behavior means that there is a limit to how many nodes that can be created to run simulations. This writer observed that after about 500 nodes, the running time of simulation increases drastically. One of the future work could be to to overcome this impediment.

8. CONCLUSION

The current implementation of caching in NDN, i.e. LRU was studied. It was found that it does not utilize cache-capacity in an efficient manner. Hence a new way of caching was proposed which incorporated probability and popularity in its decision making. It also differentiates between types of cache-servers and designates certain servers which handle more requests as "Hub". In the experiments performed, as seen in the section 6 , the proposed caching system did not perform as good enough as expected due to limited data and scale of the experiment, however it did display some promise. The project can be improved upon if the suggestions mentioned in section 7 are implemented.

9. REFERENCES

[1] Lognormal distribution, 2008.

[2] Nfd caching ndnsim. https://github.com/ndncomm/nfd-caching-ndnsim, 2015.

[3] J. S. Hunter. The exponentially weighted moving average. *J. Quality Technol.*, 18(4):203–210, 1986.

[4] T. Johnson and D. Shasha. X3: A low overhead high performance buffer management replacement algorithm. 1994.

[5] S. Mastorakis, A. Afanasyev, I. Moiseenko, and L. Zhang. ndnSIM 2.0: A new version of the NDN simulator for NS-3. Technical Report NDN-0028, NDN, January 2015.

[6] G. Peng. CDN: content distribution network. *CoRR*, cs.NI/0411069, 2004.

[7] I. Psaras, W. K. Chai, and G. Pavlou. Probabilistic in-network caching for information-centric networks. In *Proceedings of the second edition of the ICN workshop on Information-centric networking*, pages 55–60. ACM, 2012.

[8] G. F. Riley and T. R. Henderson. *The ns-3 Network Simulator*, pages 15–34. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[9] D. Saxena, V. Raychoudhury, N. Suri, C. Becker, and J. Cao. Named data networking: A survey. *Computer Science Review*, 19:15–55, 2016.

[10] K.-Y. Wong. Web cache replacement policies: a pragmatic approach. *Network, IEEE*, 20(1):28–34, 2006.