Markov Chain Approximations to Evaluate CCN Caching Systems

Jatin Tarachandani, Taha Adeel Mohammed

Indian Institute of Technology Hyderabad

April 19, 2024

Reference

Title

A versatile Markov chain model for the performance analysis of CCN caching systems

Authors

- Hamza Ben-Ammar - University of Rennes, IRISA, France
- Yassine Hadjadj-Aoul - University of Rennes
- Gerardo Rubino - University of Rennes
- Soraya Ait-Chellouche - University of Rennes

Year of Publication

- 2018

Contents

Introduction

Introduction

CCN?

- **C**ontent **C**entric **N**etworking
- Here, network of nodes (graph representable)
- Nodes have their own distinct caches
- Some nodes have attached content repositories (final destination)
- Requests transmitted across network via *interest packets*, and corresponding *data packets* cached (or not) at nodes along which response is sent
- Above point depends on caching strategy

Introduction

Why CCN?

- Allows to focus more on content itself rather than location
- Intermediate caching reduces round-trip time and server load
- Achieves significant improvements [1]

Other Related Terminology

- LCE: **L**eave **C**opy **E**verywhere - caching strategy that basically means whenever a data comes as a response through a node, cache a copy
- LRU - **L**east **R**ecently **U**sed - individual nodes' cache replacement algorithm considered in this work

[1] G. Rossini and D. Rossi, "A dive into the caching performance of content centric networking," in Proc. of IEEE CAMAD, Sep. 2012, pp. 105–109.

Introduction

Markov Chain Analysis of this System (High-level)

- We consider a r-ranked set of content $\{c_r\}, r \in \{1, 2, ...R\}$. Probability of requesting any one $\rightarrow p_r$.
- Goal - Approximate the cache hit ratio for a content c_r and validate against simulation numbers
- DTMCs used to model a *single cache node*
    - One DTMC constructed per content item
    - Transition probabilities dependent on p_r
- Further generalization to a system of cache nodes forming the network

System Description

System Description

Notations

- $G = (V, E)$: Graph representing the network of caches
- $V = \{v_1, \ldots, v_M\}$: Set of nodes, each with a cache
- $E \subseteq V \times V$: Set of links between nodes
- $C = \{c_1, \ldots, c_R\}$: Set of content items
- p_r: Probability of requesting content c_r
- N: Size of the cache for each node, where $N << R$
- $\beta(r)$: Probability that c_r is cached at a node upon a cache miss

System Description

- All content items c_r have an identical size and are stored permanently at some nodes (servers) in the network.
- The clients generate independent and identically distributed sequence of requests from the catalog of content items. (**IRM** - **I**ndependent **R**eference **M**odel) [2]
- **Shortest Path Routing:** Each node has a **F**orwarding **I**nformation **B**ase (FIB) table that contains the next hop for each content item.
- **LRU Replacement:** The cache replacement policy is Least Recently Used (LRU).

[2] A. Dan and D. Towsley, "An approximate analysis of the LRU and FIFO buffer replacement schemes," *SIGMETRICS*, vol. 18, pp. 143-152, Apr. 1990

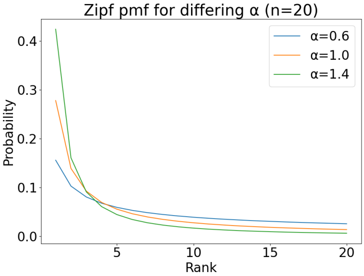
- **Zipf's Law:** The content catalog C has an inherent popularity ranking following the Zipf distribution[3].

$$p_r = \frac{r^{-\alpha}}{\sum_{r=1}^{R} r^{-\alpha}}, \text{ where}$$

$c_r \rightarrow$ content with popularity rank r

$p_r \rightarrow$ probability of requesting c_r

$\alpha \rightarrow$ skewness of the Zipf distribution



[3] F. Guillemin, T. Houdoin, and S. Moteau, "Volatility of youtube content in orange networks and consequences," *2013 IEEE International Conference on Communications*

Single Node Cache Analysis

Single Node Cache Analysis

Given an LRU Cache with size N, when a content c_r is requested and recieved at a node, for any $c_{r'}$ (with $r' \neq r$) at position i in the cache, the following cases arise:

- $c_{r'}$ is moved down by one position if c_r is not in the cache and c_r is being cached currently (with prob. $\beta(r)$), or if c_r is at j with $j > i$.
- $c_{r'}$ will remain at the same position if c_r is at j with $j < i$ or if it has been decided to not cache c_r (with prob. $1 - \beta(r)$).
- $c_{r'}$ will be evicted if it occupies the N^{th} position and c_r is not in the cache and c_r is being cached currently (with prob. $\beta(r)$).

MACS

- *Configuration:* $\vec{x} = (x_1, x_2, \ldots, x_N)$, where x_i is the content item at position i in the cache.
- $|Configurations| = R!/(R-N)!$ Huge!!

Markov chain-based Approximation of CCN Caching System (MACS)

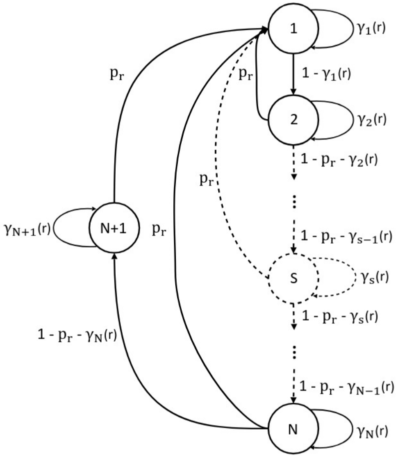
- Consider a Markov chain $X(r)$ with N+1 states.
- The chain represents the evolution with time of the position occupied by c_r in the cache.
- State 1 means that c_r is at the top of the cache, ... , state N means that c_r is at the bottom of the cache.
- State $N+1$ means that c_r is not in the cache.
- The chains $X(1), X(2), \ldots, X(R)$ are independent of each other.

Transition Probabilities

Definition: $\gamma_s(r)$

$\gamma_s(r) \rightarrow$ probability that content c_r stays at the same state s in $X(r)$, when $\beta(r) = 1$

$$\begin{cases} \gamma_1(r) = p_r, \\ \gamma_s(r) = \sum_{i=1, i \neq r}^{R} p_i \sum_{j=1}^{s-1} \pi_j(i),\ s \in [2, N], \\ \gamma_{N+1}(r) = 1 - p_r. \end{cases} \quad (1)$$



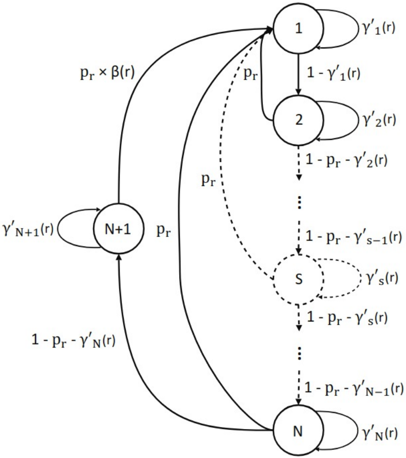
Transition Probabilities

Definition: $\gamma_s'(r)$

$\gamma_s'(r) \rightarrow$ probability that content c_r stays at the same state s in $X(r)$

$$\begin{cases} \gamma_s'(r) = \gamma_s(r) + \sum_{i=1, i\neq r}^{R} p_i \pi_{N+1}(i)(1-\beta(i)), \\ \gamma_{N+1}'(r) = 1 - p_r\beta(r). \end{cases} \tag{2}$$

where $\pi_{N+1}(i)$ is the probability that c_i is not in the cache.

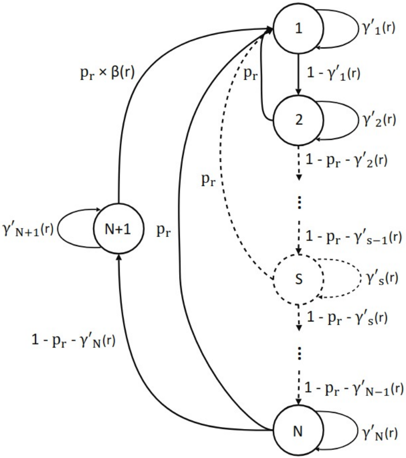


Transition Probabilities

Definition: $\mathbf{T_{i,j}}$

$T_{i,j} \rightarrow$ Transition probability from state i to state j in $X(r)$

$$\begin{cases} T_{i,i} = \gamma'_i(r), & i \in [1, N+1], \\ T_{i,1} = p_r, & i \in [2, N], \\ T_{N+1,1} = p_r\beta(r), & \\ T_{1,2} = 1 - \gamma'_1(r), & \\ T_{i,i+1} = 1 - p_r - \gamma'_i(r), & i \in [2, N], \\ T_{i,j} = 0, & j \notin \{1, i, i+1\}. \end{cases} \tag{3}$$



Stationary Distribution

Stationary Distribution Existence

- From the markov chain, we can see that the states form a SCC and hence the chain is irreducible.
- Also, due to self loops, the chain is aperiodic. $(0 < \gamma_s'(r) < 1)$
- Hence, the chain is ergodic and has a unique stationary distribution $\pi(r) = (\pi_1(r), \pi_2(r), \ldots, \pi_{N+1}(r))$.
- The cache miss probability would be given by $\pi_{N+1}(r)$ and the cache hit rate by $(1 - \pi_{N+1}(r))$.

Stationary Distribution

Stationary Equations

Using the Chapman-Kolmogorov equations ($\pi = \pi P$ and $\sum \pi_i = 1$) and the transition probabilities derived earlier, we obtain:

$$\begin{cases} \pi_1(r) = \pi_1(r)\gamma_1'(r) + \sum_{i=2}^{N} \pi_i(r) p_r + \pi_{N+1}(r) p_r \beta(r) \\ \pi_2(r) = \dfrac{\pi_1(r)(1-\gamma_1'(r))}{1-\gamma_2'(r)} \\ \pi_i(r) = \dfrac{\pi_{i-1}(r)(1-p_r-\gamma_{i-1}'(r))}{1-\gamma_i'(r)},\ i \in [3, N+1] \\ \pi_1(r) + \pi_2(r) + \cdots + \pi_{N+1}(r) = 1 \end{cases} \tag{4}$$

Stationary Distribution for LCE

For the LCE caching strategy, $\beta(r) = 1$ for all r.

$$\implies \pi_1(r) = p_r \tag{5}$$

Hence, the system of equations derived earlier can be solved to get

$$\begin{cases} \pi_1(r) = p_r \\ \pi_2(r) = \dfrac{p_r(1-p_r)}{1-\gamma_2(r)} \\ \pi_i(r) = \dfrac{p_r(1-p_r)\prod_{j=2}^{i-1}(1-p_r-\gamma_j(r))}{\prod_{j=2}^{i}(1-\gamma_j(r))}, \ i \in [3, N+1] \end{cases} \tag{6}$$

Stationary Distribution Computation for Generic Case

- In the general case of $0 \leq \beta(r) \leq 1$, $\pi_1(r)$ cannot be computed directly since it depends on all other $\pi_i(r)$.
- And each $\pi_{i\geq 2}(r)$ is a function of $\pi_{i-1}(r)$ (and hence $\pi_1(r)$).
- Hence to compute $\pi_1(r)$, the equations can be rearranged to get:

$$\pi_1(r) = \frac{p_r(1 + (\beta(r) - 1)\pi_{N+1}(r))}{1 - (\gamma_1'(r) - \gamma_1(r))} \tag{7}$$

- Now $\pi_1(r)$ depends only on $\pi_{N+1}(r)$.
- **Fixed Point Iteration:** Successive approximations of $\pi_{N+1}(r)$ can be used to get $\pi_1(r)$ and then the rest of the values.

Multiple Nodes System

Multiple Nodes System

Extending to Multiple Cache Nodes

- So far, we derived for the states of a single cache node.
- For multi-node network, we must consider request forwarding as another addition to p_r.
- Under SPR, we must consider requests from other nodes due to cache misses - miss stream or MS_r

Definition

The outgoing miss stream rate for a content c_r from a node u is:

$$MS_r(u) = req(r, u) \times \pi_{N+1}(r, u) \tag{8}$$

where $req(r, u)$ is the proportion of requests for content c_r received by u.

Incoming Miss Stream

Definition

The incoming miss stream rate for content c_r at node v is:

$$\eta_r(v) = \sum_{u:NH(u)=v} \left(MS_r(u) \prod_{w \neq u:NH(w)=v} (1 - MS_r(w)) \right) \tag{9}$$

- $\{u : NH(u) = v\}$ is the set of nodes s.t. u's **N**ext **H**op on the shortest path towards the repo location of c_r is v.
- We take the $(1 - MS_r(w))$ terms due to request aggregation (duplicate requests collated, then forwarded)

Adjusting p_r

- As a result, node's content requests = client-origin requests + cache-miss-origin forwarded requests
- Thus, value p_r used in DTMC's $\pi_i(r)$ formula is no longer a valid probability of incoming requests
- It thus becomes (for each node v):

$$p'_r = \frac{p_r + \eta_r(v)}{\sum_{k=1}^{R} (p_k + \eta_k(v))} \tag{10}$$

and resubstitute p'_r in the original MC equations to get the adjusted π.

Experimental Simulations

Simulator and Fixed Params

- Simulations were conducted on *ccnSim* [4], which is a discrete-event CCN simulator.
- Initially requests are sent until steady state (ccnSim checks this using batched Coeff. of Variation on p_{hit}), post which 10^6 requests are sent.

Catalog size	20000
Client model	Poisson, 1req/s
$\beta(r)$	0.5
Cache replacement	LRU

Table: Fixed Parameters

[4] Chiocchetti, Raffaele, Rossi, Dario and Rossini, Giuseppe, "ccnSim: an Highly Scalable CCN Simulator." In IEEE International Conference on Communications (ICC), June 2013.

Variables / Topologies

Variables considered:

- Zipf law skew parameter α
  - Values from 0.8 to 1.2
- Cache size as a percentage of catalog size
  - Values from 0.1% to 1%
- Topology:
  - 15-node binary tree
  - 31-node binary tree

  Here, clients are leaf nodes, repository is the root node

Result graphs

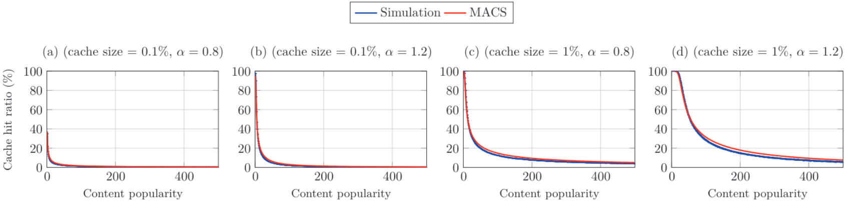
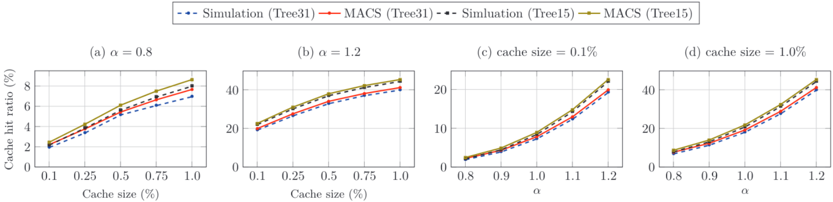


Figure 3: Total hit probability vs content popularity under binary tree topology (31 nodes)



Our Work

Average Number of Hops for a Request

- We build on the equilibrium distribution results $\pi(r)$ of MACS to get the average number of network hops before a request for content c_r is satisfied.
- Let $\phi(u_0, c_r) = \{u_0, u_1, \ldots, u_k\}$ be the shortest route/path from u_0 to a server containing c_r.
- The expected number of hops for a request for c_r from u_0 is:

$$\begin{aligned} \mathbb{E}\left[H(u_0, c_r)\right] &= \sum_{i=0}^{k} i \times \prod_{j=0}^{i-1} Pr(\text{Cache miss at } u_j) \times Pr(\text{Cache hit at } u_i) \\ &= \sum_{i=0}^{k} (i \times \prod_{j=0}^{i-1} \pi_{N+1}(c_r, u_j) \times (1 - \pi_{N+1}(c_r, u_i))) \end{aligned} \tag{11}$$

Evaluating on other graphs

- Tree network topology considered is simplistic
- Average case topologies would be a better indicator of robustness of MACS
- We do this by using random graphs for topologies

Barabási - Albert model[5]

- Method to generate random, scale-free graphs

Barabási-Albert(n, m_0):

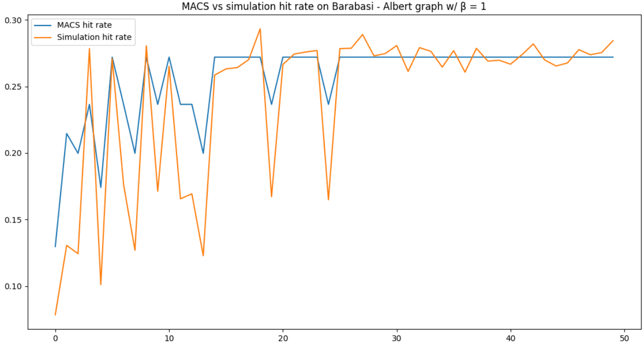
- Start with an initial seed network of m_0 nodes
- Until number of nodes is n do:
  - Add a new node with $m \leq m_0$ links to existing nodes in the graph.
  - Probability that one of these new links connects to any node k given by:

$$Pr(k) = \frac{k_i}{\sum_{j=1}^{n} k_j}$$

where k_i is the degree of vertex k.

[5] Barabási, Albert-László, and Albert, Réka, "Statistical mechanics of complex networks." https://doi.org/10.1103/RevModPhys.74.47

- Using networkx and python scripting these can easily be converted to ccnSim's DSL and simulated upon.
- MACS was computed over every state, vertex, r combination using numpy(for vectorization).



Thank you!