Front. Comput. Sci., 2015, 9(6): 846–859
DOI 10.1007/s11704-015-4064-7

RESEARCH ARTICLE

Hierarchical caches in content-centric networks: modeling and analysis

Zixiao JIA[1,2], Jiwei HUANG[1], Chuang LIN (✉)[1]

1 Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China
2 National Computer Network Emergency Response Technical Team/Coordination Center of China, Beijing 100029, China

Abstract Content-centric network (CCN) is a new Internet architecture in which content is treated as the primitive of communication. In CCN, routers are equipped with *content stores* at the content level, which act as caches for frequently requested content. Based on this design, the Internet is available to provide content distribution services without any application-layer support. In addition, as caches are integrated into routers, the overall performance of CCN will be deeply affected by the caching efficiency.

In this paper, our aim is to gain some insights on how caches should be designed to maintain a high performance in a cost-efficient way. We try to model the two-layer cache hierarchy composed of CCN routers using a two-dimensional discrete-time Markov chain, and develop an efficient algorithm to calculate the hit ratios of these caches. Simulations validate the accuracy of our modeling method, and convey some meaningful information which can help us better understand the caching mechanism of CCN.

Keywords CCN, cache, model, analysis

1 Introduction

With the flourishing of Internet application and the development of computing techniques, the concise pattern of current IP-based network contributes to the most noticeable scalability, but also becomes a shackle for the further development of Internet services. Under such background, reforming the Internet architecture has become one of the most popular issues in the study field of computer network.

Nowadays, many researchers pay much attention to the Content Network, and more significant projects focusing on the future Internet architecture design have been funded in recent years. Content-centric network [1] (CCN) provides a clean-slate design for the Internet, where content becomes the primitive of communications. In CCN, each piece of content contains a name that uniquely identifies it, and is transmitted in a receiver-driven way. When the requested content is returned from the source, it is remembered/cached by intermediate routers (termed as *content routers*). Accordingly, the subsequent requests for the same content can be served later by these content routers without resorting to the source again.

CCN becomes a good alternative for Internet service providers (ISPs) to offload their IP backbone traffics, which are currently overwhelmed by over-the-top (OTT) content like Internet videos. Figure 1 shows a possible deployment of CCN in an ISP's network, where content routers are organized in layers, with the lowest layer accepting requests from customers and the topmost layer connecting to the IP backbone. In virtue of the built-in caching capability, CCN enables the Internet to support content distribution services directly in its network layer, without any application-layer solutions, e.g., CDN.

As caches will be integrated into routers, the network performance will be influenced by the caching efficiency. Thus, the deployment of CCN in ISPs' networks is not that straightforward. For instance, the benefits of employing CCN can even be offset if the storage volumes of these caches are too

Received February 17, 2014; accepted April 1, 2014

E-mail: chlin@tsinghua.edu.cn

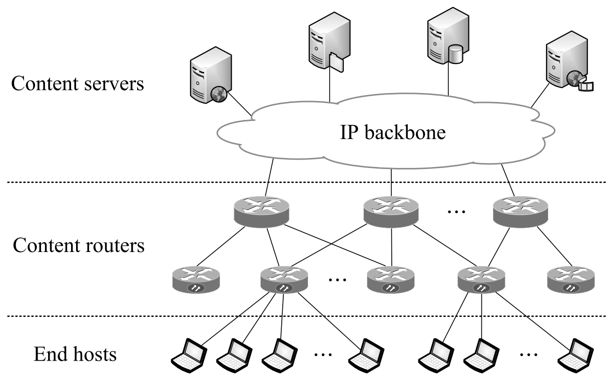

Fig. 1 The hierarchical structure of content routers in CCN

small. On the other hand, due to their high demand on both data access speed and storage volume, caches stand as a major infrastructure investment for ISPs. Therefore, how these caches should be designed cost-efficiently, while still maintaining a high performance is a problem to be examined.

The caching problem in the area of web caching has been extensively studied. However, caches in CCNs are actually different from web caches in the following respects. Firstly, web caches are application-specific (for web browsing), while CCN caches are for general content distribution services. Secondly, Web caches use single-path routing, while CCN allows content to be retrieved and distributed along multiple paths. Thirdly, web caching is based on objects/files, while caching in CCN is based on chunks/packets.

Some efforts have also been made on studying the caching performance in CCN. Nevertheless, most of them are based on trace-driven simulations or experiments. Existing modeling methods for CCN caches are either complicated to be solved or fall short in giving valuable guidance on how to design the caches.

This paper studies the performance issues of caches in CCN. The layers of content routers are treated as a cache hierarchy (Fig. 1), and we contrive to develop models to guide the design of caches at different layers. Specifically, we use discrete-time Markov chains (DTMC) to capture the dynamics of content occupancy in hierarchical caches, which can be used to efficiently calculate the cache hit ratios. Additionally, the variables used in our model, which convey some meaningful information, can help us better understand the caching mechanism of CCN. The main contributions of this paper are as follows:

1) We propose probability models for two-layer cache hierarchy. These models are based on two-dimensional Markov chains, each of which contains a relatively large number of states. To obtain analytical solutions, we introduce a key variable R to decompose the two-dimensional Markov chain into a series of one-dimensional Markov chains, which can be iteratively solved.

2) Based on our models, we develop an efficient algorithm to calculate the hit ratio for each cache. Simulations show that the results are accurate with error less than 5%.

3) By analyzing the numerical results obtained from our models, we gain some understanding on (a) the impact of cache size on hit ratio, (b) the contribution of each cache parts to the hit ratio of root node, (c) size requirements for different layers of caches in CCN, etc.

4) We also present some interesting findings through the variables used in our model: (a) the caching characteristics for content of different popularities, (b) the relation between two adjacent layers of nodes to explain the "filter effect" imposed by lower-layer caches.

The rest of this paper is organized as follows. In Section 2, we briefly introduce the background of CCN and make some assumptions. Section 3 presents the leaf node model, the layer-1 node model and the extended layer-1 model, respectively. Section 4 validates our models in terms of simulations, and reports some numerical results obtained from our models. Section 5 surveys some related work and Section 6 concludes this paper.

2 Background and assumptions

2.1 Background

In CCN, there are two types of packets: interest and data. A data packet contains not only the content, but also the name that identifies the content. A user who is interested in a data packet broadcasts an interest packet over its connections.

Having received the interest request, the router (termed as *content router*) will check its *content store*, which is a cache of data packets, according to its content name. If the requested data packet is already cached in the content store, the data packet would be returned directly by that router. On the other hand, if the data packet is not found, the content router will look up in its *forwarding information base* (FIB), and forward the interest request to a list of outgoing interfaces. Here, FIB is a data structure similar to the routing table in a traditional IP router, with the difference that the FIB lookup is based on content name rather than IP address. The router also keeps track of the pending request by inserting a record to its *pending interest table* (PIT).

When the corresponding data packet is returned, the router checks its PIT to decide which interface to forward this data packet. After forwarding, the router also keeps a replica of the data packet in its content store. Then, if interest packets for the same content arrive again, the router can directly return the content.

As seen above, data requests are not constrained to flow along a single path as in IP networks, but can be routed in a multi-path fashion. In addition, interest packets for the same content will be aggregated as a single PIT entry containing a list of interfaces. When the content is returned, it will be sent over all these interfaces. This enables CCN to inherently support multicast transmission.

In CCN, all contents are first splitted into packet-sized chunks (4KB as specified in current CCNx implementation[1)]). To request a file, users send out a sequence of interest packets, one for each chunk. Note that these interest packets can be pipelined to reduce the response time.

Because of the limited capacity, cache need to discard the old chunk when inserting a newer content, and how to choose the discarded chunk causes the cache replacement policy.

2.2 Assumptions

In our model, the system consists of three components: content servers, content routers and end hosts. Content routers are equipped with content stores (caches), and are organized in layers, as shown in Fig. 1. We term routers of lowest layer as *leaf routers* or *leaf nodes*[2)]. Leaf routers can accept data requests from end hosts that are directly connected to them. If data requests cannot be satisfied locally, they would be forwarded to the upper-layer routers, which we term as *layer-1 routers*. The key notations of our model are shown in Table 1.

• **Content popularity** Let C be the collection of all chunks in the system, and these chunks are classified into K ranks of popularity. Chunks in lower ranks will be requested by end hosts with higher probabilities. For each leaf node v_i, let C_k^i be the set of rank-k chunks that can be requested by its end hosts, and define the *request ratio* α_k^i as the probability that a requested chunk belongs to rank k.

• **Request arrival** We assume that the arrival of chunk requests at any leaf node conforms to the independent reference model (IRM), which is widely adopted to model cache accesses [2, 3]. Specifically, let random variable X_j be the rank of the jth requested chunk at a leaf router v_i, then $X_1, X_2, \ldots$ are independent and identically distributed (i.i.d.). According to the request ratio defined above, we have $P(X_j = k) = \alpha_k^i$.

Table 1 Summary of key notations.

Term	Description
I_A	Indictor function which takes 1 if predicate A is true, and 0 otherwise
C	Set of all chunks in the system
K	Number of popularity ranks
C_k^i	Set of rank-k chunks requested by end hosts connected to v_i
$\alpha_k^i(\beta_k^i)$	Request ratio of rank-k chunks (a specific chunk) at node v_i
g_k^i	Size of set C_k^i
S_i	Cache size of node v_i
$p_k^i(j)$	Probability that any rank-k chunk is stored at the jth slot of v_i
$b_k^i(j)$	Expected number of rank-k chunks stored in the first j slots of v_i
$h^i(h_k^i)$	Cache hit ratio of node v_i (for a specific rank k)
$\sigma_{m,n}$	Probability that node v_m forwards missed chunk requests to node v_n
δ_i	Probability that when a request comes to the system, it arrives at v_i
$R_k^{m,n}(j)$	Probability that a chunk c of rank k is not present in the cache of v_n condition that it is present in the jth slot of v_m
$D_k^{m,n}(j)$	Expected number of rank-k chunks in the first j positions of v_m but not existing in node v_n
$P_x(i, j)$	Transition probability used in the (extended) layer-1 node model
$P_x^+(i, j)$	Transition probability used in the extended layer-1 node model contributed by the node itself
$P_x^-(i, j)$	Transition probability used in the extended layer-1 node model triggered by other nodes
$P_x'(i)$	Alias for transition probability defined in the leaf node model
$P_x''(i, j)$	Alias for $P_x(i, j)$ defined in the layer-1 node model

• **Multi-path routing** We assume that the routing policy for chunk requests is *multi-path*. Formally, let N be the number of nodes in the system, then the routing policy can be characterized using a matrix $F = (\sigma_{i,j})_{N\times N}$, where $\sigma_{i,j} \in [0, 1]$ is the probability that node v_i forwards missed requests to node v_j. Since a request can be forwarded to multiple nodes, $\sum_j \sigma_{i,j}$ can be larger than 1.

• **Node cache** We consider the cache of a router as a sequence of slots, and each slot can only contain one chunk. To simplify the model analysis, we use slot as the unit to depict the cache size in the following modeling and experiments.

• **Cache replacement** We assume a simple LRU strategy for cache replacement. If a requested chunk is not in the cache, it would be brought from other routers and placed at the first slot of the cache. All other chunks in the cache are pushed down one position, and the chunk in the last slot is discarded. On the other hand, if the requested chunk is already in the cache, it would be brought to the top slot, and all chunks that are previously ahead of it will be pushed down to one position.

[1)] CCNx project: http://www.ccnx.org

[2)] In our model, we would use routers and nodes interchangeably.

3 Model analysis

This section presents models to evaluate the performance of hierarchical caches in CCN. We characterize the caching performance with *overall hit ratio* h^i, defined as the probability that a requested chunk is stored in the cache of node v_i. In addition, we are also interested in evaluating *rank-k hit ratio* h^i_k, defined as the probability that a requested chunk of rank-k is stored in the cache of node v_i. Before moving on to the models, we introduce some more notations that will be used later.

Let S_i be the number of slots in node v_i's cache. Then, for any $j \in [1, S_i]$, define $p^i_k(j)$ as the probability that any rank-k chunk is stored at the jth slot of v_i. It easily follows that $b^i_k(j) = \sum_{t=1}^{j} p^i_k(t)$ is the expected number of rank-k chunks stored in the first j slots of v_i. For simplicity of notation, let $g^i_k = |C^i_k|$, and define $\beta^i_k = \alpha^i_k / g^i_k$ as the request arrival ratio of a specific rank-k chunk at node v_i.

3.1 Leaf node model

Let us consider the caching model for a single leaf node v_1, and develop an expression of hit ratios h^1 and h^1_k.

By the definitions of $p^i_k(j)$ and $b^i_k(j)$, the rank-k hit ratio is:

$$h^1_k = \frac{\sum_{j=1}^{S_1} p^1_k(j)}{\sum_{j=1}^{M} p^1_k(j)} = \frac{b^1_k(S_1)}{g^1_k}, \tag{1}$$

and the overall cache hit ratio is:

$$h^1 = \frac{\sum_{k=1}^{K} h^1_k \cdot \alpha^1_k}{\sum_{k=1}^{K} \alpha^1_k}. \tag{2}$$

To determine $b^1_k(j)$, we construct a discrete-time Markov chain (DTMC), which captures the dynamics for a specific chunk $c \in C^1_k$ in the cache of v_1, as shown in Fig. 2. In this Markov chain, each state represents the slot that c occupies: state j means that c is in the jth slot; state $M = S_1 + 1$ means that c is outside of the cache. A transition is triggered when a request arrives to node v_1. Let $P_{i \to j}$ be the transition probability from state i to state j. In the following, we will determine all these transition probabilities.

$P_{i \to i},\ i \in [2, S_1]$ corresponds to the probability that any chunk in the first $i-1$ slots is requested:

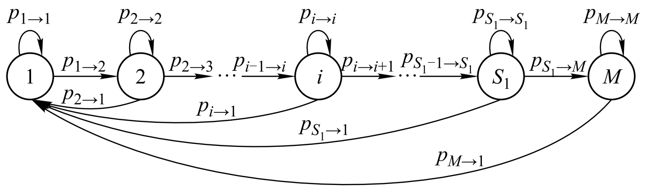

Fig. 2 The Markov chain of the leaf node model

$$P_{i \to i} = \sum_{t=1}^{K} \beta^1_t \cdot b^1_t(i-1),$$

and $P_{M \to M}$ is the probability that c is not requested:

$$P_{M \to M} = 1 - \beta^1_k.$$

$P_{i-1 \to i},\ i \in [2, M]$ corresponds to the probability that neither c nor any other chunk in the first $i-2$ slot is requested:

$$P_{i-1 \to i} = 1 - \sum_{t=1}^{K} \beta^1_t \cdot \left[b^1_t(i-2) + I_{t=k}\right],$$

where I_A is a indictor function which takes 1 if predicate A is true, and 0 otherwise.

$P_{i \to 1},\ i \in [1, M]$ is the probability that c is requested:

$$P_{i \to 1} = \beta^1_k.$$

Let $\vec{\pi} = [\pi(1), \pi(2), \ldots, \pi(M)]$ be the steady state distribution, then the balance equations are as follows:

$$\begin{cases} \pi(1) = \beta^1_k \cdot \sum_{i=1}^{M} \pi(i), \\ \pi(i) = P_{i-1 \to i} \cdot \pi(i-1) + P_{i \to i} \cdot \pi(i), \quad i \in [2, M], \\ \sum_{i=1}^{M} \pi(i) = 1. \end{cases}$$

According to our definition, $p^1_k(j) = g^1_k \pi(j)$. Combining the above results, we have the following recursive equations:

$$p^1_k(j) = p^1_k(j-1) \cdot \frac{1 - \sum_{t=1}^{K} \beta^1_t \cdot \left[b^1_t(j-2) + I_{t=k}\right]}{1 - \sum_{t=1}^{K} \beta^1_t \cdot b^1_t(j-1)},\ i \in [2, M], \tag{3}$$

with the initial value $p^1_k(1) = \alpha^1_k$.

Using Eq. (3), we can solve $p^1_k(j)$ for $j \in [1, S_1]$, and obtain $b^1_k(S_1)$. Then, h^1_k and h^1 can be calculated using Eq. (1) and Eq. (2), respectively.

3.2 Layer-1 node model

Let us consider the caching model for a single layer-1 node v_0, which is connected with one leaf node v_1. We aim to evaluate the values of h^0 and h^0_k.

Different from the previous leaf node model, a cache hit of chunk c at node v_0 requires not only that c is in the cache of

v_0, but also that c is not in the cache of v_1. In other words, the hit ratio of v_0 not only depends on its own state, but also depends on v_1's state. To this end, we introduce another variable to express hit ratios of node v_0.

Let $C_i(c) = j$ be the event that the chunk c is present in the jth slot of node v_i; $C_i(c) = 0$ when c is not present in node v_i. Define $R_k^{m,n}(j)$ as the probability that a chunk c of rank k is not present in the cache of v_n condition that it is present in the jth cache slot of v_m.

$$R_k^{m,n}(j) = \Pr\left\{C_n(c) = 0 \mid C_m(c) = j\right\}, \quad j \in [1, S_m].$$

The variable R defined above characterizes the relation between two nodes. Specifically, $R_k^{m,n}(j)$ reflects how useful a rank-k chunk stored in the jth slot of node v_m is for node v_m. Using R, we can express the rank-k hit ratio of node v_0 as:

$$h_k^0 = \frac{\sum_{j=1}^{S_0} p_k^0(j) \cdot R_k^{0,1}(j)}{g_k^0}. \tag{4}$$

The overall hit ratio of node v_0 can be derived in the similar way as the leaf node model using Eq. (2), with α_k^0 expressed as:

$$\alpha_k^0 = \frac{\alpha_k^1(1 - h_k^1)}{\sum_{t=1}^{K} \alpha_t^1(1 - h_t^1)}.$$

Then, we define $D_k^{m,n}(j)$ as the expected number of rank-k chunks present at the first jth slots of v_m, but not present in v_n's cache. $D_k^{m,n}(j)$ would be expressed as:

$$D_k^{m,n}(j) = \textstyle\sum_{i=1}^{j} p_k^m(i) \cdot R_k^{m,n}(i). \tag{5}$$

By modeling the cache of v_0 using a one-dimensional Markov chain, we can derive the following recursive equation in a similar way as the leaf node model:

$$p_k^0(j) = p_k^0(j-1) \cdot \frac{1 - \sum_{t=1}^{K} \beta_k^1 \cdot \left[b_t^1(S_1) + D_t^{0,1}(j-2) + R_k^{0,1}(j-1) I_{t=k}\right]}{1 - \sum_{t=1}^{K} \beta_t^1 \cdot \left[b_t^1(S_1) + D_t^{0,1}(j-1)\right]}, \tag{6}$$

where $p_k^0(1) = \alpha_k^0$.

To determine $R_k^{0,1}(j)$, we construct a two-dimensional discrete-time Markov chain, as shown in Fig. 3. In this Markov chain, state (i, j) means that c is present in the ith slot of node v_1 and jth slot of node v_0, where $i \in \{1, 2, \ldots, S_1, M\}, j \in \{1, 2, \ldots, S_0, M\}$. Let $\vec{\pi} = [\pi(1,1), \pi(1,2), \ldots, \pi(1,M), \pi(2,1), \ldots, \pi(M,M)]$ be the steady state probabilities. Then, $R_k^{0,1}(j)$ can be expressed as:

$$R_k^{0,1}(j) = \frac{\pi(M, j)}{\sum_{i=1}^{S_1} \pi(i, j) + \pi(M, j)}. \tag{7}$$

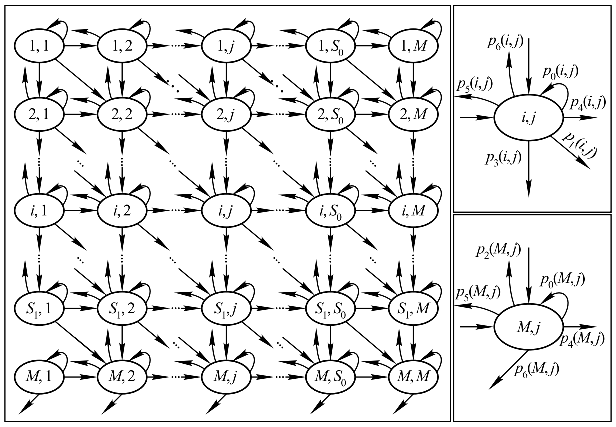

Fig. 3 The two-dimensional Markov chain of the layer-1 node model. On the right side, we list two states (i, j) and (M, j) to illustrate different state transitions

The remaining problem is how to solve this Markov chain (please refer to Appendixes A and B for the transition probabilities and balance equations). Since the number of states in this Markov chain is $(S_0+1)\times(S_1+1)$, solving it numerically can be difficult when the cache sizes are very large. Even worse, the computation will become more difficult when there are multiple leaf nodes, as to be seen in Section 3. On the other hand, analytical solutions are more desirable since they allow more efficient computation, and can be extended.

Solving this Markov chain analytically is also tricky since the transition probabilities contain variables $R_k^{0,1}(j)$, which are functions of $\vec{\pi}$. This causes a loop which prevents a straightforward solution. To address this problem, we first transform the balance equations (see Appendix C), and define $\vec{\pi}_j = [\pi(1, j), \pi(2, j), \ldots, \pi(M, j)]$ which corresponds to the steady state probabilities of the jth vertical chain. Then, we have the following observations:

1) $\vec{\pi}_j, j > 1$ can be solved once $\vec{\pi}_{j-1}$ is determined;
2) The value of $R_k^{0,1}(j)$ only depends on the ratio of elements in $\vec{\pi}_j$.

Based on the first observation, we can solve this two-dimensional Markov chain by iteratively solving $\vec{\pi}_1, \vec{\pi}_2, \ldots, \vec{\pi}_M$ in sequence. The second observation allows us to bootstrap this iteration by setting the right side of the first transformed balanced equation (see Appendix C) to 1. In the following, we will give details of this solution.

First, we vertically divide this two-dimensional Markov chain into S_0+1 one-dimensional Markov chains $\{MC_1, MC_2, \ldots, MC_{S_0}, MC_M\}$. Figure 4 shows the transition graph for MC_j, where state i means that c is in the ith slot of v_1; state M means that c is not in v_1. We add an absorbing state O to aggregate all the other states that c is not present in the jth slot of v_0. Once the system reaches state O, the probability of moving out of O is 0. The transition probabilities of this Markov chain can be derived easily.

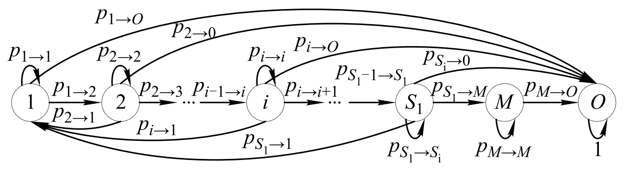

Fig. 4 MC_j, the jth vertical Markov chain

Then, we show how to iteratively solve Markov chains from MC_1 to MC_{S_0}. As seen in Fig. 4, the Markov chain has no steady state, since there is an absorbing state. Here, we adopt the approach given in Ref. [4] as follows.

Let φ_j^i denote the average number of times that state i is visited before the chain reaches the absorbing state O. Then, we have:

$$\varphi_j^i = q_j^i + \sum_u \varphi_j^u P_{ui}, \quad i, u \in \{1, \ldots, S_1, M\}, \tag{8}$$

where q_j^i is the probability that MC_j starts at state i, and P is the transition probability matrix of MC_j. Let $\vec{q}_j = \{q_j^1, q_j^2, \ldots, q_j^M\}$ represent the initial state distribution of MC_j. In our case, the initial-state distribution of MC_1 is $\vec{q}_1 = \{1, 0, \ldots, 0\}$; $\vec{q}_j (j > 1)$ can be calculated using the steady state probability of MC_{j-1} and the transition probabilities from MC_{j-1} to MC_j:

$$q_j^i = \begin{cases} 0, & i = 1; \\ \varphi_{j-1}^{i-1} \cdot P_1(i-1, j-1), & i \in [2, S_1]; \\ \varphi_{j-1}^{S_1} \cdot P_1(S_1, j-1) + \varphi_{j-1}^M \cdot P_4(M, j-1), & i = M. \end{cases} \tag{9}$$

Thus, we can determine $\varphi_j^1, \ldots, \varphi_j^{S_1}, \varphi_j^M$, and solve $R_k^{0,1}(j)$ using:

$$R_k^{0,1}(j) = \frac{\varphi_j^M}{\sum_{i=1}^{S_1} \varphi_j^i + \varphi_j^M}. \tag{10}$$

With $R_k^{0,1}(j)$, we can calculate $p_k^0(j)$ according to Eq. (6). Repeat this process until we have all the values of $R_k^{0,1}(j)$ and $p_k^0(j)$. Finally, we can obtain h_k^0 and h^0 according to Eq. (4) and Eq. (2).

3.3 Extended layer-1 node model

This subsection extends the previous layer-1 node model by considering the case that v_0 is connected to multiple leaf nodes $v_1, v_2, \ldots, v_n$. In addition, we also incorporate multipath routing in our model, using variables $\sigma_{m,n}$ defined in Section 2.

As there are multiple leaf nodes, we need to specify their request arrival rates. Let λ_i be the request arrival rate at leaf node v_i, and define $\delta_i = \lambda_i / (\sum_{j=1}^n \lambda_j)$ as the probability that when a request comes to the system, it arrives at node v_i.

Now, we introduce the equation for cache hit ratio of rank-k chunks at v_0:

$$h_k^0 = \frac{\sum_{l=1}^n \dfrac{D_k^{0,l}(S_l)}{g_k^l - b_k^l} \cdot \dfrac{g_k^l}{g_k^0} \cdot \alpha_k^l \cdot (1 - h_k^l)}{\sum_{l=1}^n \alpha_k^l \cdot (1 - h_k^l)}. \tag{11}$$

Similar to previous models, we have the following recursive relation for the calculation of $p_k^0(j), j \in [1, S_0]$:

$$p_k^0(j) = p_k^0(j-1) \sum_{l=1}^n \delta_l \frac{1 - \sum_{t=1}^K \beta_t^l \Big[b_t^l(S_l) + D_t^{0,l}(j-2) + R_k^{0,l}(j-1) I_{t=k} \Big]}{\dfrac{1}{\sigma_{l,0}} - \sum_{t=1}^K \beta_t^l \Big[b_t^l(S_l) + D_t^{0,l}(j-1) \Big]},$$

with $p_k^0(1) = \alpha_k^0$ defined as:

$$\alpha_k^0 = \frac{\sum_{l=1}^n \sigma_{l,0} \cdot \alpha_k^l (1 - h_k^l)}{\sum_{l=1}^n \sum_{t=1}^K \sigma_{l,0} \cdot \alpha_t^l (1 - h_t^l)}.$$

Note that $\sigma_{l,0} > 0$, otherwise, it is not necessary to include v_l in this model.

As seen above, we need to calculate $R_k^{0,i}(j)$ and $p_k^i(j)$ for each node i and slot j, and use the above recursive relation to compute $p_k^0(\cdot)$. That is, we need to solve n two-dimensional Markov chains for $(v_0, v_1), (v_0, v_2), \ldots, (v_0, v_n)$ respectively. Each chain is similar in structure to the layer-1 node model, but with different transition probabilities. The difference comes from the fact that each chain, say (v_0, v_i) would be affected by all leaf nodes other than v_i.

To illustrate, we consider one such Markov chain for (v_0, v_1). To capture the influences from other leaf nodes, we express the transition probabilities $P_x(i, j)$ for $x \in \{0, 1, \ldots, 6\}$ in the following form, with x represents seven transition probabilities out of its current state in the two-dimensional Markov chain, as shown in the right of Fig. 3.

$$P_x(i, j) = \delta_1 P_x^+(i, j) + \sum_{l=2}^{n} \delta_l P_x^-(i, j),$$

where $P_x^+(i, j)$ corresponds to the transition triggered by a request at v_1 itself, and $P_x^-(i, j)$ corresponds to the transition triggered by a request at node other than v_1.

First, for $P_x^+(i, j)$, if we do not consider multi-path routing, they are the same with $P_x(i, j)$ in our layer-1 model. When we consider multi-path routing, they would be defined as:

$$\begin{aligned} P_x^+(i, j) &= \sigma_{1,0} \cdot P_x''(i, j) + (1 - \sigma_{1,0}) \cdot P_x'(i), & x \in \{0, 3, 6\}; \\ P_x^+(i, j) &= \sigma_{1,0} \cdot P_x''(i, j), & x \in \{1, 4, 5\}, \end{aligned}$$

where $P_x''(i, j)$ takes the same value as $P_x(i, j)$ ($x \in \{0, 1, \ldots, 6\}$) defined in the previous layer-1 node model, and $P_x'(i)$ is the alias of the transition probability in the leaf model, expressed as follows:

$$P_x'(i) = \begin{cases} P_{i \to i}, & x = 0; \\ P_{i \to i+1}, & x = 3; \\ P_{i \to 1}, & x = 6. \end{cases} \tag{12}$$

Apart from the above cases, we have another new transition $P_2(M, j)$ from (M, j) to $(1, j)$, which is 0 in the previous model. Here $P_2(M, j)$ is defined as:

$$P_2^+(M, j) = \beta_k^1 \cdot (1 - \sigma_{1,0}).$$

Then, let us determine the $P_x^-(i, j)$ for each x:

1) For $P_1(i, j)$, $P_2(i, j)$, $P_3(i, j)$, and $P_6(i, j)$, the position of chunk c at node v_1 is changed, therefore the corresponding events only taken place when content requests arrive at node v_1. Thus, we have:

$$P_1^-(i, j) = P_2^-(i, j) = P_3^-(i, j) = P_6^-(i, j) = 0.$$

2) For $P_0(i, j)$, $P_4(M, j)$, and $P_5(M, j)$, we have:

$$P_0^-(i, j) = \sigma_{l,0} \cdot \sum_{t=1}^{K} \beta_t^l \cdot \left[b_t^l(S_l) + D_t^{0,l}(j - 1) \right] + (1 - \sigma_{l,0}),$$

$$P_4^-(i, j) = \sigma_{l,0} \left\{ 1 - \sum_{t=1}^{K} \beta_t^l \cdot \left[b_t^l(S_l) + D_t^{0,l}(j-1) + R_t^{0,l}(j) I_{t=k} \right] \right\},$$

$$P_5^-(i, j) = \sigma_{l,0} \cdot \beta_k^l \cdot R_k^{0,l}(j).$$

Unlike the previous layer-1 node model, this model has more states other than (1, 1) which have incoming transitions from other chains. Thus, we cannot simply assign $(1, 0, \ldots, 0)$ to the initial state of MC_1. Through estimating, we found that $\varphi_1^i, i \in [2, S_1]$ are negligible compared to φ_1^1 and φ_1^M. Thus, we only need to determine $R_k^{0,1}(1)$, and set $\varphi_1^M = R_k^{0,1}(1)$, $\varphi_1^1 = 1 - R_k^{0,1}(1)$. To solve $R_k^{0,1}(1)$, we utilize an important property of the R value: independence of the cache size of the level-1 node. That is, the probability that chunk c is not present in the cache of v_1 condition c is present in the jth slot of v_0 is irrelevant to the cache size of v_0. Thus, we can assume the cache size of v_0 is 1, and construct a simplified two-dimensional Markov chain with $2 \times (S_1 + 1)$ states, as shown in Fig. 5.

In this Markov chain, the transition probability $P_4(i, j)$ and $P_5(i, j)$ contain undetermined R variables, but we can express them as:

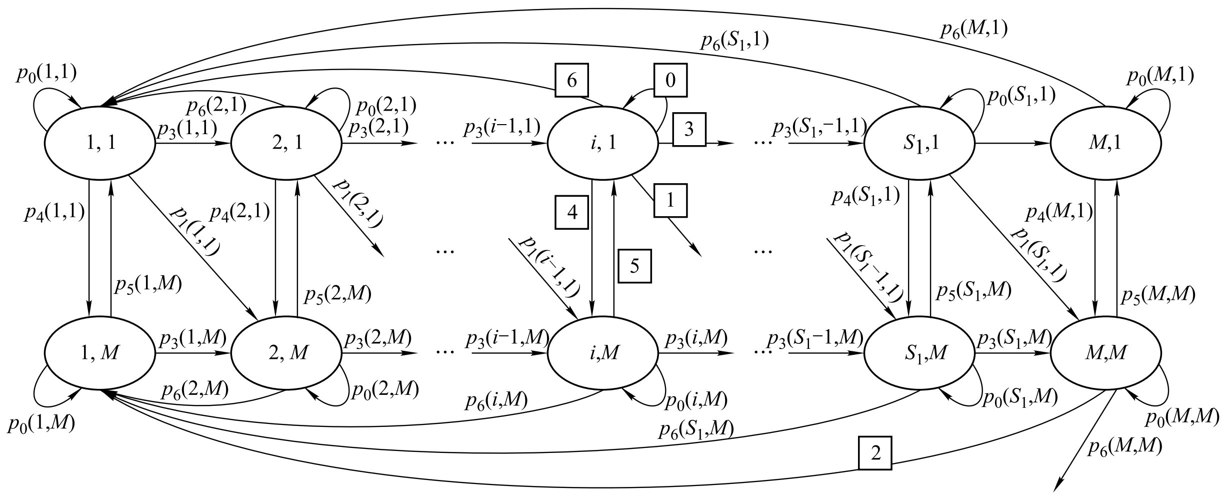

Fig. 5 The simplified Markov chain for solving $R_k^{0,1}(1)$

$$P_4(i,1) = 1 - P_0(i,1) - P_1(i,1) - P_3(i,1) - P_6(i,1),$$
$$P_5(i,M) = 1 - P_0(i,M) - P_2(i,M) - P_3(i,M) - P_6(i,M).$$

We sum up the procedure of calculating h_k^0 and h^0 in Algorithm 1.

Algorithm 1 ExtLayer1NM(V_L,$\{\lambda^L\}$)

```
Input: $V_L$: Leaf nodes set.
Input: $\{\lambda^L\}$: Request arrival rates of leaf nodes.
Output: The overall hit ratio of node $v_0$.
1  /* Step 1: Calculate $\{p_k^i(j), b_k^i(j), h_k^i\}$ for each leaf node $v_i \in V_L$ */
2  foreach $v_i \in V_L$ do
3      Initialize $p_k^i(1)$ and $b_k^i(1)$ with $\alpha_k^i$, $k \in [1,K]$;
4      foreach $j \in [2, S_i], k \in [1,K]$ do
5          $p_k^i(j) \leftarrow$ Eq. (3);
6          Update $b_k^i(j)$ with $b_k^i(j-1) + p_k^i(j)$;
7      $h_k^i \leftarrow$ Eq. (1) for $k \in [1,K]$;
8  /* Step 2: Calculate $b_k^0(j)$ and $R_k^{0,i}(j)$ */
9  /* Initialization */
10 foreach $v_i \in V_L$ do
11     foreach $k \in [1,K]$ do
12         Initialize $p_k^0(0)$, $b_k^0(0)$ and $D_k^{0,i}(0)$ with zero;
13         Get $R_k^{0,i}(1)$ by solving Markov chain in Fig. 5;
14         $\{\varphi_j^1, \varphi_j^2, \ldots, \varphi_j^M\} \leftarrow \{1 - R_k^{0,i}(1), 0, \ldots, R_k^{0,i}(1)\}$;
15         $\{p_k^0(1), b_k^0(1), D_k^{0,i}(0)\} \leftarrow \{\alpha_k^0, \alpha_k^0, \alpha_k^0 \cdot R_k^{0,i}(1)\}$;
16 /* Iteratively solve $b_k^0(j)$ */
17 foreach $j \in [2, S_0], k \in [1,K]$ do
18     foreach $v_i \in V_L, k \in [1,K]$ do
19         Solve $MC_j$ for $\vec{\varphi}_j$;
20         $R_k^{0,i}(j) \leftarrow \varphi_j^M / \sum_{i=1}^{M} \varphi_j^i$;
21         /* Get the initial states of $\vec{\pi}_j$ */
22         $\vec{q}_{j+1} \leftarrow \vec{\varphi}_j$ using Eq. (9);
23     $p_k^0(j) \leftarrow$ Eq. (6) for $k \in [1,K]$;
24     $D_k^{0,i}(j) \leftarrow D_k^{0,1}(j-1) + p_k^0(j) \cdot R_k^{0,i}(j)$;
25     Update $b_k^0(j)$ with $b_k^0(j-1) + p_k^0(j)$;
26 /* Step 3: Evaluate the overall hit ratio of root node $v_0$ */
27 $h_k^0 \leftarrow$ Eq. (11) for $k \in [1,K]$;
28 $h^0 \leftarrow$ Eq. (2);
29 return $h^0$;
```

4 Numerical results

In this section, we present numerical results obtained based on our models. We aim to show the accuracy of our model with simulations, and provide some understanding to guide the design of hierarchical caches in CCN. Since we are only studying the two-layer hierarchical cache model, we will term the layer-1 node as root node.

4.1 Experiment setup

We use the OMNeT++[3), a discrete-event simulation package, to construct the content-centric networking environment [1]. Rather than implementing a full-fledged one, we only include the basic functions of CCN: multi-path routing, chunk-based caching and receiver-driven transport protocol. The forwarding table of each node is computed using the CCNd method introduced in the CCNx project.

We developed a simplified version of ProWGen [5] to generate a large pool of data chunks[4). Then, we partition these trunks into ten popularity ranks from 1 to 10. For each leaf router, we randomly sample 500 chunks from the chunk pool. The topology to be used in our experiments is shown in Fig. 6, where each node represents a content router. Nodes at level 1 and level 2 correspond to leaf nodes and root nodes in our model; the node at level 5 acts as the source of all the contents; nodes in levels 3 and 4 are used to simulate the network backbone.

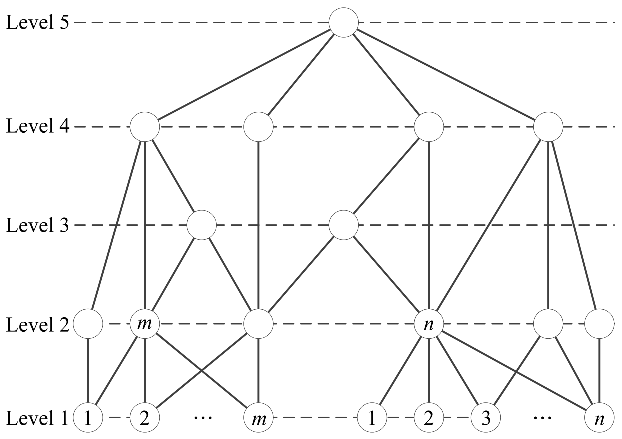

Fig. 6 The router topology used in our experiments

4.2 Model verification

In order to verify the accuracy of our models, we calculate hit ratios for different cache sizes, and compare them to the simulation results. To verify our three models, we consider three different kinds of nodes: single leaf node, root node connected with one leaf node, and root node connected with multiple leaf nodes. For each case, we repeat the simulations for ten times to cover the randomness of chunk requests.

Figure 7(a) reports the hit ratios of a leaf node with cache size ranging from 10 to 150. The error bars reflect the result variance of the ten simulation runs. Note that our modeling results are quite close to the simulative results, with differ-

3) OMNeT++ Network Simulation Framework: http://www.omnetpp.org
4) It is shown that ProWGen can generate workloads that are similar to empirical traces [6]

ence less than 2.5%.

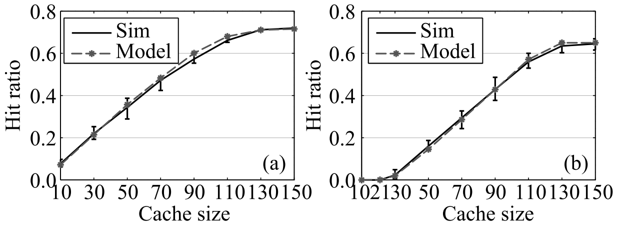

Fig. 7 The relation between hit ratio and cache size for (a) single leaf node, and (b) root node connected with one leaf node

Figure 7(b) shows the relationship between hit ratio and cache size for a root node connected with one leaf node. We observe that: 1) the hit ratio is close to 0 when the cache size is below 21. The reason is that the chunks stored at the first $j \leqslant 21$ slots are also stored in the leaf node with a high probability. As a result, $R_k^0(j)$ is negligible for these small j, and the hit ratio is thus very low according to Eq. (4); 2) after the cache size of the root node becomes larger than 21, the hit ratio climbs quickly, but will not increase much after the size exceeds 130. This is because these slots at the rear of the cache are mostly occupied by cold contents which are seldom requested (see the next experiment for details). These content contributes little to the overall hit ratio according to Eq. (2).

By the above two observations, our model can be employed to find the two threshold values s_1 and s_2 (here $s_1 = 21$, $s_2 = 130$). With the cache size of root node set between s_1 and s_2, we can expect a quick growth of overall hit ratio.

It is also seen that the shape of curves in Figs. 7(a) and 7(b) are remarkably different, owing to the fact their request arrival patterns are different. For the leaf node, requests arrive at it according to the IRM assumption (see Section 2); while the root node is fed with non-IRM requests. This confirms the "filter effect" previously studied in Ref. [7]: lower-level nodes selectively filter out chunk requests, and the resultant requests are no longer independent of each other.

We continue to consider the scenarios of multiple leaf nodes. Figure 8 plots the hit ratio of the root node when the leaf node number $n = 2, 3, 4, 5$. As we apply approximation in the calculation (see Section 3), the accuracy drops a little, but the error is still bounded by 5%. Note that the hit ratios in these cases are all above 0 even when the cache size is very small. This is due to the independence of chunk requests from leaf nodes, i.e., the chunk recently requested by node v_1 can still be requested by v_2 within a short period. Another interesting observation is that as the number of leaf nodes increases, the shapes of curves become more and more similar to that of the leaf node in Fig. 7(a). This may imply that the request arrival at root node tends to follow IRM again, i.e., the "filter effect" becomes less noticeable with the increase of leaf node number.

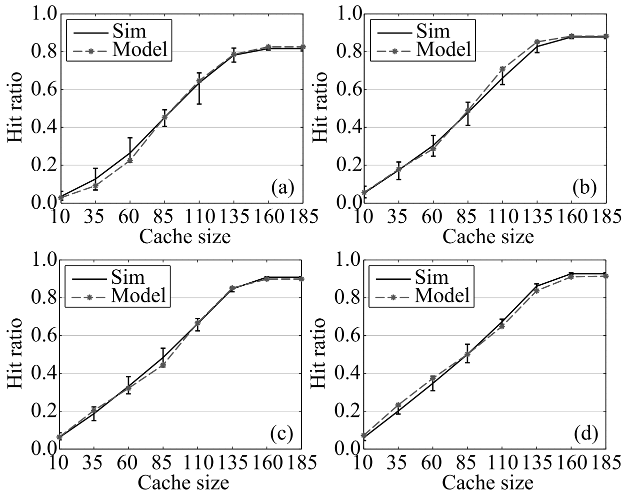

Fig. 8 The relation between hit ratio and cache size of a root node, when the number of leaf nodes is (a) n = 2, (b) n = 3, (c) n = 4, and (d) n = 5

4.3 Which parts of the cache are mostly valuable?

In the previous experiments, we already see that the hit ratio increases as the cache size becomes larger. Also, we locate thresholds s_1 and s_2 within which the increase of hit ratio is the most remarkable. That is to say, investment on cache slots between s_1 and s_2 may be a good strategy, as they contribute a lot to the overall hit ratio. For better illustration, Fig. 9 provides a more fined-grained view of how much each slot contributes to the overall hit ratio of the root node connected with two leaf nodes.

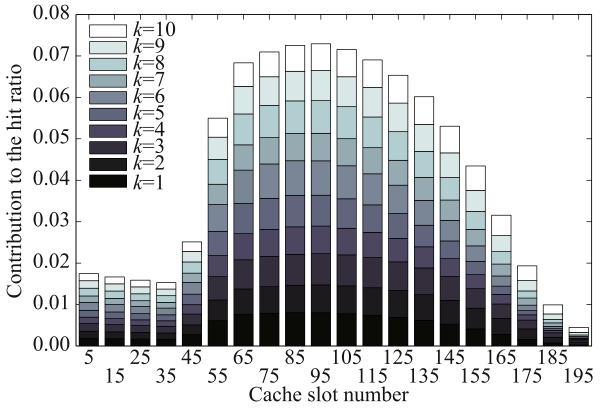

Fig. 9 The contribution of each slot to the hit ratio of the root node, which is connected with two leaf nodes

In Fig. 9, we see that the slots from 55 to 145 contribute most to the cache hit ratio, which agrees with Fig. 8(a). By increasing the memory access speed of these slots, we may expect a better performance for the content router.

Owing the scale of y-axis is not sensitive to different content popularities in Fig. 9, each rank seems to have almost equal contribution to the overall hit ratio. If independently drawing the absolute contribution of these ranks, we can easily find their difference.

4.4 The relation between content popularity and its location — a study of value p

In this experiment, we study the relationship between a chunk's popularity and its location in the cache, by analyzing the value of $p_k^i(j)$. Figure 10 shows the cache occupancy statistics reflected by $p_k^i(j)$, in scenarios of one leaf node, root node connected with one leaf node and two leaf nodes, respectively. Based on Fig. 10, we have the following two major observations:

1) Different ranks of chunks tend to occupy different parts of the cache: higher ranks of chunks tend to occupy around the head of the cache, while lower ranks of chunks tend to occupy near the tail of the cache. Take Fig. 10(a) as an example, rank-1 chunks which are the most popular have the peak probability of 0.32 to be located in the first slot; while rank-10 chunks are most likely (with probability around 0.3) to be located in the last slot of the cache. By this observation, we can design location-aware caching strategies to provide differentiated service according to content popularity.

2) Compared to the leaf node, the distribution of $p_k^i(j)$ for each k is more stable for the root node. This again illustrates the "filter effect" which we mentioned in previous experiments. Here, the leaf nodes filter out many requests for popular content. The more popular the content is, the more it will be filtered by nodes of lower layers. As a result of the filter effect, requests arrived at the root node are more likely for cold content. Thus, we conclude that root node needs more capacity to hold cold content in order to maintain a relatively high performance. As for how much capacity is needed, please refer to Section 6.

Finally, a phenomenon that needs further explanation is in Figs. 10(a) and 10(b), $p_k^i(j)$ drops to 0 when the slot number $j = 160$. This is because in our experiment setting, the total number of chunks that may be requested to each leaf node v_i ($\sum_{k=1}^{10} g_k^i$) is set to 160. When there are two leaf nodes, as the case for Fig. 10(c), the total number of *unique* chunks that may be requested to these two nodes exceeds 180. That's why we do not see $p_k^i(j)$ drops to 0 when $j = 180$.

4.5 Uncovering the relation between two adjacent layers of nodes — a study of value R

In this experiment, we aim to uncover the relation between two adjacent layers of nodes, by studying the value of R defined in Section 2. Specifically, we calculate $R_1^{0,1}(j)$ for the root node v_0, by fixing the cache size of leaf nodes to be 30, and varying the number of leaf nodes from 1 to 5. The results are given in Fig. 11.

As seen in Fig. 11, the values of $R_1^{0,1}(j)$ are relatively small for the first few slots. This is due to the fact that these slots store the most recently requested chunks, which are very likely to be cached by one or more leaf nodes too. For example, when there is only one leaf node, the most recently requested chunks are definitely stored at the leaf node. As a result, the values of $R_1^{0,1}(j)$ at the first 21 slots is 0. This explains why the hit ratio of leaf node is 0 when its cache size is below 21, as seen in Fig. 8(b).

Note that the initial value $R_1^{0,1}(1)$ becomes larger as the number of leaf nodes increases. This is easy to understand since a chunk request from any of the leaf nodes will bring the requested chunk to the front of root node's cache. Thus, the chunks stored at the first few slots of the root node may have been requested by leaf nodes other than v_1. The more the leaf nodes, the more likely this will happen.

The value of $R_1^{0,1}(j)$ will reach 1 as the number of slots exceeds a threshold. However, it does not mean that the slots whose numbers are bigger than this threshold are still useful. The reason is that R is a condition probability, and the value of $p_k^0(j)$ already reaches 0 for these slots since the number of chunks in our system is limited.

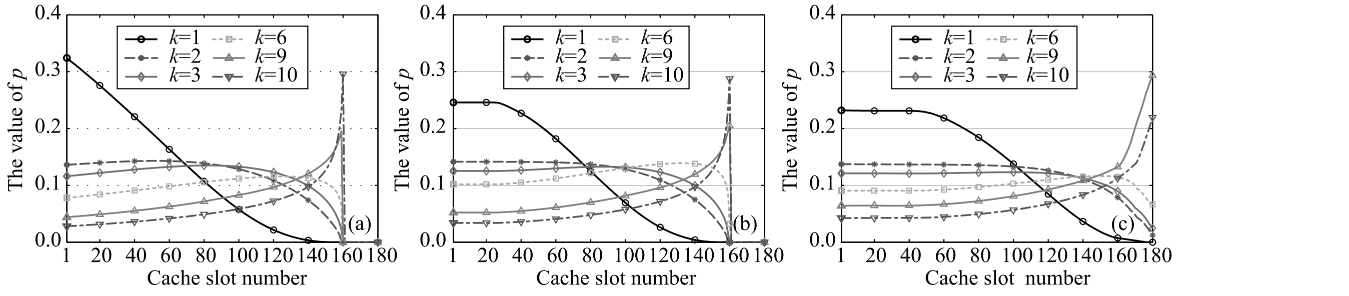

Fig. 10 The value of $p_k^i(j)$ for different slot j in (a) leaf-node v_i, (b) root node v_i connected with one leaf node, (c) root node v_i connected with two leaf nodes

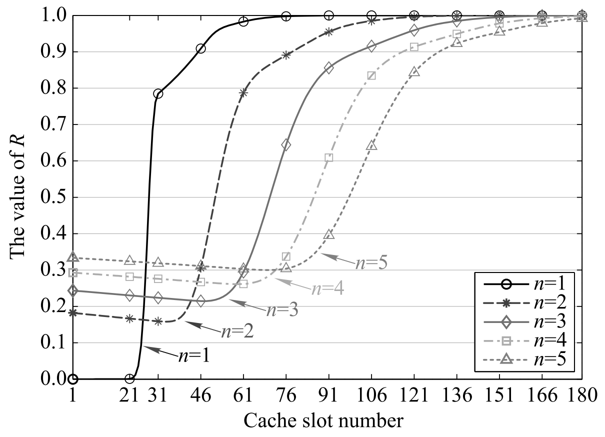

Fig. 11 The values of $R_1^{0,1}(j)$ at different slots j of the root node v_0, when connected with $n = 1, 2, 3, 4, 5$ leaf nodes

4.6 How large cache do we need for the root node?

In the previous experiments, we have mentioned the "filter effect" imposed by leaf nodes, and gained the intuition that the root node should have a relatively large cache to maintain a relatively high hit ratio. In this experiment, we justify this claim by studying the minimum cache size for the root node to maintain a hit ratio above 0.5. We vary the number of leaf nodes connected to the root node from 1 to 6, and the results are shown in Fig. 12. Note that each point is averaged over ten runs by varying the input (randomly generated chunk requests). We have the following two observations:

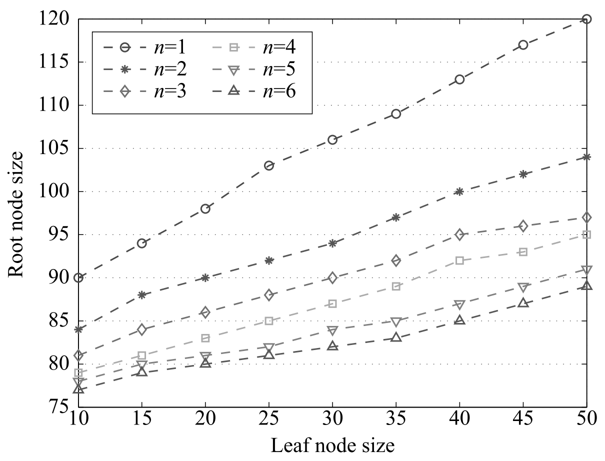

Fig. 12 The cache size requirements for root node when connected with different number of leaf nodes

1) The root node requires a much larger cache size than the leaf nodes. As shown in Fig. 12, The ratio is 3× when there is only one leaf node, and 2× when there are six leaf nodes. This may indicate that using the same replacement policy in hierarchical caches without any cooperations is inefficient. If the root node employs a different replacement policy, we may be able to reduce the required cache size for the root node.

2) The requirement for cache size is lower when there are more leaf nodes. This is because different nodes have different popularity assignment (α_k^i) in our model, and the requests for chunks of different popularity tend to be distributed more uniformly with the increase of leaf nodes.

5 Related work

CCN has received a lot of attention since it was proposed in Ref. [1]. It has been gradually proving its great potentiality in Internet telephony [8], autonomous driving [9], and ad hoc networks [10, 11]. Topics which are still under study include security [12], content router [13], transport protocol [14], etc.

Content caching has been extensively studied in the research area of web cache. Similar to caches in CCN, web caches can store the recently requested file for a short time so that subsequent requests can be satisfied locally. Dan et al. [15] study the LRU and FIFO cache replacement strategies for a single cache node which is fed with IRM file requests. As the computation complexity of directly calculating cache hit probability is very high, they develop an approximated approach to estimate the hit probability. Che et al. [2] propose another approximation approach to evaluate the performance of two-level caches by assuming LRU replacement policy and IRM request arrival. They approximate the evaluation by assuming a constant value for inter-arrival times between two consecutive requests for a same document without cache misses. This assumption is claimed to be good when they are a large number of files. Based on [15], Rosensweig et al. [3] analyze a more general cache network where there is no fixed topology. The problem of multi-cache problem is decomposed into a set of single-cache problems, which can be solved independently. However, they assume that the cache miss stream to be IRM, which would affect the accuracy of their model. Through trace-drive simulations, Williamson [7] examines the filter effect observed in caching hierarchies: caches at one level will filter the data request, and change the workload pattern at its next levels.

There are some efforts on analyzing the caching performance in Content-Centric Networks, but they are mostly based on simulations or experiments [16–19]. However, for a fundamental understanding of the performance of CCN caches, some analytical models are still needed. Towards this goal, Psaras et al. [20] try to use Continuous-Time Markov Chain (CTMC) to capture the caching dynamics. They first introduce a simple model for one node, and then extend the model to multiple nodes. However, as the time is continuous, the model is not easy to solve, and not much results are given.

Carofiglio et al. [21] take another approach by assuming the requests arrivals conform to Markov modulated rate process (MMRP). But to make the chunks misses independent (similar to IRM), they assume the file size to be "memoryless", i.e., geometrically distributed.

To sum up, though web cache has been extensively studied, they can not be directly applied to CCN hierarchical caching systems due to features including chunk-based caching and multi-path request routing. In addition, existing continuous models are complicated and are short in giving valuable guidance for cache design.

This paper is an extension of the work presented in ICCCN 2013 [22]. The additional contributions include two aspects. Firstly, we generalize the root-node model to the extended layer-1 node model by considering the case where one content router is connected to more than one leaf nodes, while the multi-path routing feature of CCN is incorporated. Secondly, we conduct more experiments to get deeper understanding of the cache performance, including 1) which parts of the cache are mostly valuable; 2) the relation between content popularity and its location; and 3) uncovering the relation between two adjacent layers of node. Those additional findings can give more valuable guidance about the caches designed in CCN.

6 Conclusion

This paper models and evaluates the performance of hierarchical caches formed by CCN content routers. The main characteristic that we study is cache hit ratio, including overall hit ratio and rank-k hit ratio. Our models can be used to efficiently calculate the hit ratio for up to two layers of caches in CCN. The accuracy of our models is validated through extensive experiments.

We also present five findings by analyzing the variables numerical results obtained from our models: 1) "filter effect" makes the Size-Hit Ratio curves remarkably different in CCN, and this effect becomes less remarkable when the number of caches at the lower layer increases; 2) each cache part contributes differently to the hit ratio of root node; 3) contents of different popularity tends to gather at different cache slots of CCN nodes; 4) reflect the differentiation-distribution of the contents cached between two adjacent layers of nodes; 5) root node requires much larger cache size to maintain higher hit ratio, and the size required is lower when connected by more leaf nodes.

Our future work includes: 1) conduct more experiments on real NDN networks, which may be deployed in the future; 2) extend our models to consider more than two layers of caches.

Acknowledgements This work was supported by the National Natural Science Foundation of China (Grant No. 61472199) and the Fundamental Research Funds for the Central Universities (2015RC22).

Appendixes

Appendix A Transition probabilities

i) For $1 \leqslant i \leqslant S_1, 1 \leqslant j \leqslant S_0$,

$$\begin{cases} P_0(i,j) = \sum_{t=1}^{K} \beta_t^1 \cdot [b_t^1(i-1) + I_{i=1}], \\ P_1(i,j) = 1 - \sum_{t=1}^{K} \beta_t^1 \cdot [D_t^{0,1}(j-1) + b_t^1(S_1) - p_t^1(i) + I_{t=k}], \\ P_3(i,j) = \sum_{t=1}^{K} \beta_t^1 \cdot [D_t^{0,1}(j-1) + b_t^1(S_1) - b_t^1(i)], \\ P_6(i,j) = \beta_t^1, \\ P_2(i,j) = P_4(i,j) = P_5(i,j) = 0. \end{cases}$$

ii) For $i = M, 1 \leqslant j \leqslant S_0$,

$$\begin{cases} P_0(M,j) = \sum_{t=1}^{K} \beta_t^1 \cdot [b_t^1(S_1) + D_t^{0,1}(j-1)], \\ P_4(M,j) = 1 - \sum_{k=1}^{K} \beta_t^1 \cdot [b_t^1(S_1) + D_t^{0,1}(j-1) + I_{t=k}]. \end{cases}$$

iii) For $j = M$,

$$P_3(i,M) = 1 - \sum_{t=1}^{K} \beta_t^1 \cdot [b_t^1(i-1) + I_{t=k}].$$

iv) For $i = M, j = M$,

$$P_0(M,M) = 1 - \beta_k^1.$$

Appendix B Balance equations

$$\begin{cases} \pi(1,1) = \sum_{u=2}^{S_1} P_6(u,1)\pi(u,1) + \sum_{u=1}^{M} P_6(M,u)\pi(M,u) \\ \qquad + P_0(1,1)\pi(1,1); \\ \pi(i,1) = P_3(i-1,1)\pi(i-1,1) + P_0(i,1)\pi(i,1), i \in [2, S_1]; \\ \pi(M,1) = P_0(M,1)\pi(M,1); \\ \pi(1,j) = \sum_{u=2}^{M} P_6(u,j)\pi(u,j) + P_0(1,j)\pi(1,j), \quad j \in [2,M]; \\ \pi(i,j) = P_3(i-1,j)\pi(i-1,j) + P_0(i,j)\pi(i,j) \\ \qquad + P_1(i-1,j-1)\pi(i-1,j-1), \quad i,j \in [2,S_0]; \\ \pi(M,j) = P_1(S_1,j-1)\pi(S_1,j-1) + P_4(M,j-1)\pi(M,j-1) \\ \qquad + P_3(S_1,j)\pi(S_1,j) + P_0(M,j)\pi(M,j), j \in [2,M]; \\ \sum_{i=1}^{M}\sum_{j=1}^{M} \pi(i,j) = 1. \end{cases}$$

Appendix C Transformed balance equations

With $\pi(i, j)$ changed to π_j^i,

$$\begin{cases} (1-P_0(1,1))\pi_1^1 - \sum_{u=2}^{S_1} P_6(u,1)\pi_1^u = \sum_{u=1}^{M} P_6(M,u)\pi_u^M; \\ (1-P_0(i,1))\pi_1^i - P_3(i-1,1)\pi_1^{i-1} = 0, \qquad i \in [2, S_1]; \\ (1-P_0(M,1))\pi_1^M = 0. \end{cases}$$

$$\begin{cases} (1-P_0(1,j))\pi_j^1 - \sum_{u=2}^{M} P_6(u,j)\pi_j^u = 0, \qquad j \in [2, M]; \\ (1-P_0(i,j))\pi_j^i - P_3(i-1,j)\pi_j^{i-1} \\ \quad = P_1(i-1,j-1)\pi_{j-1}^{i-1}, \qquad i, j \in [2, S_0]; \\ (1-P_0(M,j))\pi_j^M - P_3(S_1,j)\pi_j^{S_1} \\ \quad = P_1(S_1,j-1)\pi_{j-1}^{S_1} + P_4(M,j-1)\pi_{j-1}^M, \qquad j \in [2, M]. \end{cases}$$

References

1. Jacobson V, Smetters D K, Thornton J D, Plass M F, Briggs N H, Braynard R L. Networking named content. In: Proceedings of the 5th ACM International Conference on Emerging Networking Experiments and Technologies. 2009, 1–12
2. Che H, Tung Y, Wang Z. Hierarchical web caching systems: modeling, design and experimental results. IEEE Journal on Selected Areas in Communications, 2002, 20(7): 1305–1314
3. Rosensweig E J, Kurose J, Towsley D. Approximate models for general cache networks. In: Proceedings of IEEE International Conference on INFOCOM. 2010, 1–9
4. Trivedi K S. Probability and Statistics with Reliability, Queuing, and Computer Science Applications. 2nd ed. New York: John Wiley & Sons, 2001
5. Busari M, Williamson C. ProWGen: A synthetic workload generation tool for simulation evaluation of web proxy caches. Computer Networks, 2002, 38(6): 779–794
6. Saleh O, Hefeeda M. Modeling and caching of peer-to-peer traffic. In: Proceedings of the 14th IEEE International Conference on Network Protocols. 2006, 249–258
7. Williamson C. On filter effects in web caching hierarchies. ACM Transactions on Internet Technology, 2002, 2(1): 47–77
8. Jacobson V, Smetters D K, Briggs N H, Plass M F, Stewart P, Thornton J D, Braynard R L. VoCCN: voice-over content-centric networks. In: Proceedings of the Workshop on Re-architecting the Internet. 2009, 1–6
9. Kumar S, Shi L, Ahmed N, Gil S, Katabi D, Rus D. Carspeak: a content-centric network for autonomous driving. ACM SIGCOMM Computer Communication Review. 2012, 42(4): 259–270
10. Oh S Y, Lau D, Gerla M. Content centric networking in tactical and emergency MANETs. In: Proceedings of IEEE International Federation for Information Processing Wireless Days. 2010, 1–5
11. Meisel M, Pappas V, Zhang L. Ad hoc networking via named data. In: Proceedings of the 5th ACM International Workshop on Mobility in the Evolving Internet Architecture. 2010, 3–8
12. Wong W, Nikander P. Secure naming in information-centric networks. In: Proceedings of the Re-Architecting the Internet Workshop. 2010, 1–6
13. Arianfar S, Nikander P, Ott J. On content-centric router design and implications. In: Proceedings of the Re-Architecting the Internet Workshop. 2010, 5
14. Tarkoma S, Kuptsov D, Savolainen P, Sarolahti P. CAT: a last mile protocol for content-centric networks. In: Proceedings of IEEE International Conference on Communications Workshops. 2011, 1–5
15. Dan A, Towsley D. An approximate analysis of the LRU and FIFO buffer replacement schemes. ACM SIGMETRICS Performance Evaluation Review, 1990, 18(1): 143–152
16. Carofiglio G, Gehlen V, Perino D. Experimental evaluation of memory management in content-centric networking. In: Proceedings of IEEE International Conference on Communications. 2011, 1–6
17. Rossi D, Rossini G. Caching performance of content centric networks under multi-path routing. Relatório téconico, Telecom ParisTech, 2011
18. Rossi D, Rossini G. On sizing CCN content stores by exploiting topological information. In: Proceedings of INFCOM Workshops. 2012, 280–285
19. Fricker C, Robert P, Roberts J, Sbihi N. Impact of traffic mix on caching performance in a content-centric network. In: Proceedings of IEEE Conference on Computer Communications Workshops. 2012, 310–315
20. Psaras I, Clegg R G, Landa R, Chai W K, Pavlou G. Modelling and evaluation of CCN-caching trees. In: Proceedings of the 10th International IFIP TC 6 Conference on Networking. 2011, 78–91
21. Carofiglio G, Gallo M, Muscariello L, Perino D. Modeling data transfer in content-centric networking. In: Proceedings of the 23rd International Teletraffic Congress. 2011, 111–118
22. Jia Z, Zhang P, Huang J, Lin C, Lui J C S. Modeling hierarchical caches in content-centric networks. In: Proceedings of the 22nd International Conference on Computer Communications and Networks. 2013, 1–7

Zixiao Jia received his ME from Institute of Computing Technology, Chinese Academy of Sciences, China in 2010, and obtained his PhD in the Department of Computer Science and Technology at Tsinghua University, China. He is currently an engineer of the National Computer Network Emergency Response Technical Team/Coordination Center of China, China. His research interests include performance evaluation and next generation Internet.

Jiwei Huang received his BE and PhD in computer science and technology from Tsinghua University, China in 2009 and 2014, respectively. He is an assistant professor in the State Key Laboratory of Networking and Switching Technology at Beijing University of Posts and Telecommunications, China. He was a visiting scholar at Georgia Institute of Technology, USA. His research interests include services computing and performance evaluation.

Chuang Lin received his PhD degree in computer science from Tsinghua University, China in 1994. He is now a professor of the Department of Computer Science and Technology, Tsinghua University, China, and he is also an honorary visiting professor of University of Bradford, UK. His research interests include computer networks, performance evaluation, network security analysis, and Petri net theory and its applications.