Q-caching: an integrated reinforcement-learning approach for caching and routing in information-centric networks

Wouter Caarls[*], **Eduardo Hargreaves, Daniel S. Menasché**

[1] Universidade Federal do Rio de Janeiro (UFRJ)
Av. Athos da Silveira Ramos, 149 - Blc C - 21.941-909
Rio de Janeiro, Brasil

wouter@caarls.org,eduardo.hargreaves@ppgi.ufrj.br,sadoc@dcc.ufrj.br

Abstract. *Content delivery, such as video streaming, is one of the most prevalent Internet applications. Although very popular, the continuous growth of such applications poses novel performance and scalability challenges. Information-centric networks put content at the center, and propose novel solutions to such challenges but also pose new questions on the interface between caching and routing. In this paper, building on top of Q-routing we propose a caching strategy, namely Q-caching, which leverages information that is already collected by the routing algorithm. Q-caching promotes content diversity in the network, reducing the load at custodians and average download times for clients. In stylized topologies, we show that the gains of Q-caching against state-of-the-art algorithms are significant. We then consider the RNP topology, and show that Q-caching performance is more flexible while competitive when compared against existing algorithms.*

1. Introduction

Information-centric networking (ICN) is a new networking paradigm that shifts the basic network service semantics from *delivering the packet to a given destination address* to *fetching data identified by a given name* [Afanasyev et al. 2014, Jacobson et al. 2009]. In the ICN paradigm, routers are equipped with caches, with the aim of increasing network performance and scalability. As the network service is content-aware, users can opportunistically download replicas of the content which are closer to the requesters.

In traditional IP networks when there is a change in the network, routing protocols need to exchange routing updates in order to maintain the topology consistent. Thus, IP routing protocols need to converge fast in order to keep packet delivery after network changes. Fortunately, unless a failure occurs, IP networks are typically static in short timescales.

On the other hand, ICN is extremely dynamic because it is a highly distributed caching network, with content location and availability changing over time. In this case, even a static network will have dynamic properties, as the location of content may change due to caching. These differences made [Yi et al. 2014] rethink the role of routing in ICN.

[*]Now at Pontifícia Universidade Católica do Rio de Janeiro (PUC-RIO), Rua Marquês de São Vicente, 225, Gávea - Rio de Janeiro, RJ - Brasil - 22451-900

Q-routing [Boyan and Littman 1994] was proposed to address the problem of packet routing in dynamically changing networks. It uses single-hop delays to continually estimate the cost to retrieve a packet, called cost-to-go, essentially implementing an asynchronous version of the Bellman-Ford shortest paths algorithm using only local information. It is asynchronous because each router uses only local informational without any route update message.

In recent work, [Chiocchetti et al. 2013] propose a mechanism called *Inform* which consists in extend Q-routing to work in information-centric networks, by estimating not the best routing decision for a particular destination, but for a particular content.

In this paper, we propose a reinforcement learning based solution to the joint problem of content placement and routing, which naturally embraces the notion of utility for the two decisions. This is motivated by the fact that leveraging advantages of ubiquitous caching requires coordination between content placement (insertion and eviction decisions) and content search (routing decisions) [Rossini and Rossi 2014]. In particular, we exploit the fact that Q-routing computes the *cost-to-go* and use it to make not only routing but also caching decisions in a weighted least-frequently used (WLFU) manner, evicting the item with the *minimum expected cost* (MEC) to retrieve. Then, Q-routing and MEC are combined in order to efficiently route requests and store content with the goal of minimizing the download time experienced by users. The resulting combination is referred to as Q-caching.

With this kind of approach, the routing and caching decisions therefore influence each other, as caching an item will cause the routes to change, and changing a route changes the requests seen by other nodes in the network, changing their caches. It is therefore important for the two to be well-coordinated, which we realize by basing them on the same information.

There have been separate efforts to incorporate the concept of utility for placement (e.g., web proxy caching [Wooster and Abrams 1997] and TTL caches [Dehghan et al. 2016]) and search decisions (e.g., Q-routing [Boyan and Littman 1994]). Nonetheless, to the best of our knowledge none of the prior works accounted for the notion of utility in an integrated manner for placement and routing decisions.

The paper is organized as follows. First, we introduce Q-routing for information-centric networking in Section 2. Then, we propose our cost-to-go based caching strategy in Section 3. In Sections 4 and 5 we introduce the local cost minimization problem and some of the key features of Q-caching. Section 6 describes the workload used in our trace-driven experiments. We will continue by presenting our preliminary results, related work and discussions in Sections 7, 8 and 9, and drawing conclusions in Section 10.

2. Q-routing

In Q-routing [Boyan and Littman 1994], the cost-to-go is estimated locally and asynchronously using reinforcement learning [Sutton and Barto 1998]. It uses a tabular *value function* $Q_x(d, y)$ which stores, for every node x, the cost to reach destination d when routing through immediate hop y. In the original paper d is a destination *node*, but in ICNs it is a content.

The Q table is estimated on-line via the Q-learning update rule:

$$Q_x(d,y) \leftarrow Q_x(d,y) + \alpha \left(t + \left(\min_{y' \in \mathcal{N}(y)} Q_y(d,y') \right) - Q_x(d,y) \right) \tag{1}$$

where α is the parameter of an exponential moving average filter, $\mathcal{N}(y)$ is the set of immediate neighbors of node y and t is the cost (e.g., delay) between x and y. The minimum operation in Equation 1 is performed by node y when it receives the request from node x, and returned as part of the acknowledgement.

When routing a request, the Q table is consulted to find the neighbor y with the lowest cost-to-go for a particular content d. Although in general some form of *exploration* is necessary to find optimal routes (by randomly selecting interfaces which are not currently thought to provide the lowest cost), in practice the algorithm has shown to perform well [Boyan and Littman 1994]. Alternatively, a separate *exploration phase* [Chiocchetti et al. 2013] may be used to first estimate the Q values.

Note that Equation 1 estimates the expected cost incurred by node x to fetch content d through its neighbor y. Let $u(x,d,y)$ be the instantaneous local utility associated to sending a request for content d from node x to its neighbor y. In general, any utility function may be used in our estimates, by letting $t = -u(x,d,y)$. For example, some content may be treated preferentially, or some nodes may be avoided.

3. Q-caching

In previous work, Q-routing was used to find content in an information centric network using least-recently used (LRU) caching [Chiocchetti et al. 2013]. LRU is a ubiquitous caching strategy due to its fast and easy implementation, combined with good performance. However, the availability of the cost-to-go allows us to make better caching decisions than this simple heuristic. In particular, LRU does not take into account the downstream availability of the content.

If we wish to minimize the path delay for a given request, it makes sense to cache the items that are most difficult to obtain, i.e. have the highest cost-to-go $Q_x(d)$. Given a request distribution $r_x(d)$ that specifies the probability of a request for information d arriving at node x, we wish to minimize the expected cost:

$$\sum_d r_x(d) Q_x(d) = \sum_d r_x(d) \min_{y \in \mathcal{N}(x)} Q_x(d,y) \tag{2}$$

This is achieved by sorting the content according to this expected cost and caching the items with the highest values. If $r_x(d)$ is estimated by request counting, this amounts to a weighted least-frequently used (WLFU) caching policy [Wooster and Abrams 1997], where the weight is the minimum expected cost (MEC) to obtain the content.

Algorithm 1 describes the behavior of a single node. The request distribution is approximated by request counting, on line 4. Then, the caching decision is made. On the backward path, the requested item is added to the cache (line 7). If that causes the cache to overflow, the item with the minimum expected cost to obtain is evicted (line 9). The routing decision is taken in line 10, and the minimum in Equation (1) is returned on acknowledgement (line 12). Finally, the Q table is adjusted according to Equation (1) in line 13.

Algorithm 1 Q-caching

```
1: $Q \leftarrow 0, \hat{r} \leftarrow 0$
2: $C \leftarrow \emptyset$
3: **procedure** Q-CACHING(x,d)
4:     $\hat{r}_x(d) \leftarrow \hat{r}_x(d) + 1$                                ▷ Count requests
5:     **if** $d \in C(x)$ **then**
6:         **return** content to client, starting backward path
7:     Add $d$ to $C(x)$                                                        ▷ On backward path
8:     **if** $|C(x)| > B$ **then**
9:         Evict $\arg\min_{d' \in C(x)} \left(\hat{r}_x(d') \min_{y' \in \mathcal{N}(x)} Q_x(d', y')\right)$
10:    $y \leftarrow \min_{y' \in \mathcal{N}(x)} Q_x(d, y')$                   ▷ Routing decision
11:    Route request for $d$ to $y$, measuring $t$
12:    Receive $q = \min_{y' \in \mathcal{N}(y)} Q_y(d, y')$
13:    $Q_x(d, y) \leftarrow Q_x(d, y) + \alpha\,(t + q - Q_x(d, y))$
```

4. Cost Minimization

In this section we relate Q-caching to the cost minimization problem. We introduce the local optimization problem in Section 4.1, and then use it to relate Q-caching to other policies in Section 4.2.

4.1. Local Optimization Problem

In this section, we consider the local optimization problem faced by each cache-router.

variable	description
$h_x(d)$	hit probability
$r_x(d)$	probability that request is for content d
λ_x	total request arrival rate (requests/s)
$\lambda_x(d)$	request arrival rate for content d, $\lambda_x(d) = \lambda_x r_x(d)$ (requests/s)
$E[D_x]$	expected delay experienced by node x
$Q_x(d, y)$	cost to go (computed using Q-routing)
$Q_x(d)$	minimum cost to go (computed using Q-routing)
B	buffer size
$w_x(d)$	weight equal to $r_x(d)Q_x(d)$

Table 1. Notation

All quantities in this section account for a single tagged cache x. Let λ_x be the request arrival rate. Let $\lambda_x(d)$ be the request rate for content d. Then, $\lambda_x(d) = \lambda_x r_x(d)$, where $r_x(d)$ is the fraction of requests for content d. Let $h_x(d)$ be the hit probability of content d at cache x, i.e., $h_x(d)$ is the probability that content d is stored at cache x. The average delay to download a typical content at cache x is $E[D_x] = \sum_d r_x(d)(1 - h_x(d))Q_x(d)$. A summary of the notation used throughout this paper is presented in Table 1.

The optimization problem faced at cache-router x is,

$$\text{maximize} \quad \sum_d r_x(d) h_x(d) Q_x(d) \tag{3}$$

$$\text{subject to} \quad \sum_d h_x(d) = B \tag{4}$$

Note that, as in [Melazzi et al. 2014], we consider an expected buffer size constraint, i.e., the number of expected items in the cache cannot exceed the buffer size B.

Let $w_x(d) = r_x(d)Q_x(d)$. The weight $w_x(d)$ is approximated online as the product of the number of request counts to a content multiplied by the cost to go. The optimal solution to the problem above consists of storing the contents that have larger values of $w_x(d)$, i.e., setting $h_x(d) = 1$ for the contents with larger values of $w_x(d)$ and $h_x(d) = 0$ otherwise. This motivates the caching policy proposed in this paper, which consists of storing the items with largest weights (i.e., largest costs $w_x(d)$) and evicting those with *minimum expected cost* (MEC).

4.2. Contrasting MEC Against Other Policies

The optimal solution of (3)-(4) provides insight about how MEC compares against pre-existing policies. If $Q_x(d) = 1$, for all d, we note that MEC degenerates to storing only the most popular items. This intuitive policy has formally shown to be optimal in [Liu et al. 1997].

Consider now the case where the popularity of the items is not known beforehand. In that setup, a cache which does not take into account the topology and costs of the remainder of the network when making its eviction decision can use LRU or LFU as proxies to the strategy of storing only the most popular items.

MEC, in contrast, accounts for the topology of the remainder of the network when making its eviction decisions. While doing so, each cache 1) assumes that the network is optimized and 2) does not account for the impact of local decisions on the state of the other nodes in the network. Together, properties 1) and 2) provide a simple and distributed strategy, which however is suboptimal in general. The optimal routing and placement problem, accounting for all cache interdependencies, can be shown to be NP-hard [Neves et al. 2010].

5. Q-caching Features

Next, we discuss some of the features derived by combining Q-routing with MEC.

5.1. Utility-Driven Caching

Q-routing naturally allows for utility-driven routing. By leveraging the cost-to-go computed by Q-routing to drive caching decisions, we also allow for utility-driven decisions for content storage. In particular, this provides additional flexibility with respect to existing policies such as LFU or LRU, whose utility is essentially coupled to the content hit rate.

Note that LRU automatically stores all content that passes through a given cache. Under Q-caching, in contrast, the controller might decide not to store a given content which passes through the cache. This, in turn, may lead to less churn.

5.2. Enabling Content Diversity

Q-caching promotes content diversity in the system, by favoring the storage of different contents at different caches. Under LRU, the most frequently requested contents are always locally stored, irrespectively of the state of their neighbors. The state of the neighbors affects a cache only through their miss streams. Under Q-caching, in contrast, the state of the neighbors directly impacts the caching decisions at a given tagged cache. This is because the cost-to-go is distributedly computed taking into account the distribution of the content in the network. One of the consequences of increased content diversity is reduced load at the custodians, in addition to reduced expected download times.

5.3. Handling Loops

One of the critical aspects of any routing algorithm relates to the handling of loops. Initial work on ICN suggested that Pending Interest Tables (PITs) could prevent loops [Dai et al. 2012, Jacobson et al. 2009]. This is because PITs associate to each pending request a random *nonce*, which can be used to discard duplicate requests. However, recent work [G-L-Aceves 2015] indicates that ICNs are subject to routing loop problems even when cache-routers are equipped with PITs.

Routing in ICN can rely on Distance-Vector protocols, like Bellman-Ford, or Link-State Protocols, like Dijkstra. While the Dijkstra algorithm gracefully handles positive loops, Bellman-Ford based algorithms may lead to infinite relaying under adverse conditions. As Q-routing is based on Bellman-Ford, it is subject to the *count-to-infinity* problem, i.e., the convergence might take significant time in the worst case.

The *count-to-infinity* problem is exacerbated by the continuing variation of content availability due to caching. In this paper, we deal with this by not making the Q updates immediately influence the routing decisions. Instead, the Q-table used for routing is updated only periodically, allowing the Q values to settle in between routing updates. Using such a separate "target" Q-table has proven to be a stabilizing factor in recent work on large-scale reinforcement learning [Mnih 2015]. We leave a more theoretical study of the convergence to future work.

5.4. Complexity

The majority of the computational effort is expended on the caching decision, as it requires evaluating the utility of all cached objects $(O(B|\mathcal{N}(x)|))$. However, as the utility only changes when the Q value or popularity changes, the caching decision could use a priority queue, leading to a scalable $O(log(B) + |\mathcal{N}(x)|)$ implementation. The periodic update only involves memory copy, and is therefore not very expensive.

6. Workload

Next, we present the workload used in our trace-driven numerical evaluations. The physical topology is obtained from the Brazilian National Research Network (RNP). [1] The number of clients at each point of presence (PoP) is assumed to be proportional to the number of students of that region given by the 2013 university students census [INEP 2013].

[1]The topology is obtained from `http://web.archive.org/web/20140905111428/http://www.rnp.br/servicos/conectividade/rede-ipe`

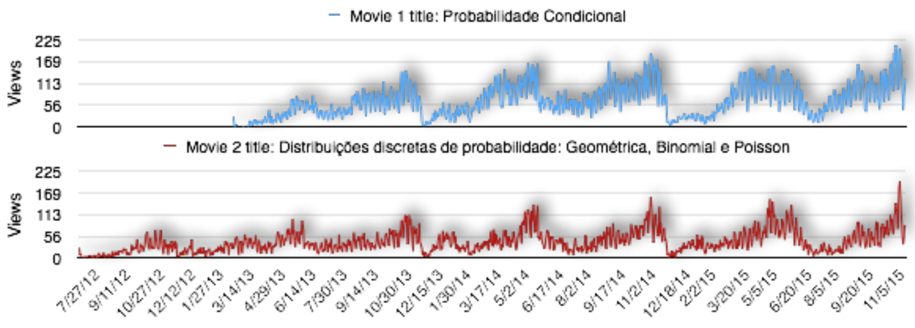


Figure 1. Time series associated to the top two most popular movies

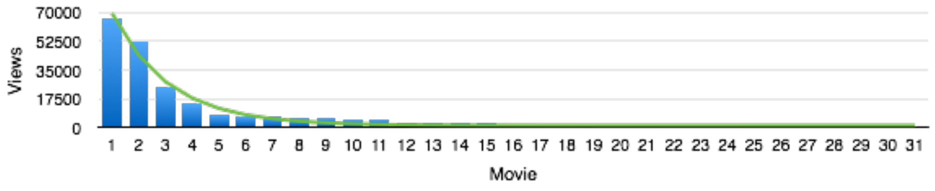


Figure 2. Distribution of the number of views per movie

We consider a content catalog comprising 31 Youtube videos from the UFRJ Performance Evaluation channel. The videos were published between 2012 and 2015, and count with up to 200,000 views and up to 700 followers. Figure 1 illustrates the time series of the two most popular movies. Each series start at the movie publishing date. As a side contribution of this work, we make available the collected traces [UFRJ 2015]. Among the interesting properties extracted from the data, we point out the following (Figure 1):

Periodicity: We note the periodic behavior of the number of views. During the semester, the number of views grows. During vacations, it decreases. During working days, the number of views is usually smaller than during the weekends. The trend is consistent among the movies.

Non-stationarity: As a general trend, the number of views increases from an year to the other. In our trace-driven simulations we consider non-stationary workloads. The simulation is divided into epochs, where each epoch corresponds to a day in the trace. The arrival rate for each content at each day is assumed to be proportional to the number of requests issued for that content, which is obtained from the trace.

Figure 2 shows the distribution of the accumulated number of views by November of 2015. The bars correspond to the collected data, and the line is result of curve fitting. The distribution popularity is well approximated by an exponential distribution, with $y = 1473 + 108947 \exp(-0.4707x)$. The root mean square of the residuals equals 2393. As a comparison, the minimum root mean square of the residuals obtained with a power law distribution equals 4024.

7. Experiments

Next, we report the numerical results of our experiments. The major strategy combinations considered are shown in Table 2. We consider four algorithms (Q-LRU, Q-routing, Q-caching and SPF + LRU), three caching strategies (LRU, LFU and MEC) and two routing strategies (shortest path first routing (SPF) and Q-routing). Q-LRU is a simplified version of *Inform*. Under SPF, requests take the path with minimum cost towards the custodian, assumed to be fixed and given. In all cases, if a replica is opportunistically found, the request is immediately served. We first consider simple topologies in Section 7.1. Then, we contrast different solutions under the RNP topology in Section 7.2.

Algorithm	Routing	Caching	Notes
Q-LRU	Q-routing	LRU	Simplified Inform. LRU is too volatile to work with Q-routing
Q-caching	Q-routing	MEC	less volatile than Inform, comparable to Q-LFU
Q-LFU	Q-routing	LFU	comparable to Q-caching
SPF+LRU	shortest path first	LRU	see Section 7.2

Table 2. Some of the strategy combinations considered in this paper

In our simulations, run in Matlab, we assume stationary latency between hops, with zero download delays (ZDD assumption) on the backward path, which is assumed to be identical to the forward path. As such, when a piece of content is found, all caches on its request path are immediately updated in accordance to the ZDD assumption. All nodes run in lock-step, and process their entire request queue on every step. The ability of Q-routing to dynamically route around congestion is therefore not used, but will be the subject of future work with more detailed simulations.

7.1. Simple Topologies

Figure 3 presents the three simple topologies considered in our experiments. The layered topology (Figure 3(a)) is inspired by the Akamai network [Sitaraman et al. 2014]. The tree topology (Figure 3(b)) is used to clarify the role of caching as opposed to routing in network performance.

The reference setup considered in our experiments is shown in Table 2. The request arrival rate for the k-th content is given by $1/k^{\alpha}$, for $k = 1, \ldots, C$, where α is referred to as the *Zipf parameter*. Parameters are varied according to our experimental goals. For each parameter setting we simulated 10 runs to obtain the mean and the 95% confidence interval of the metrics of interest, represented by a line and a shaded region in the plots that follow, respectively.

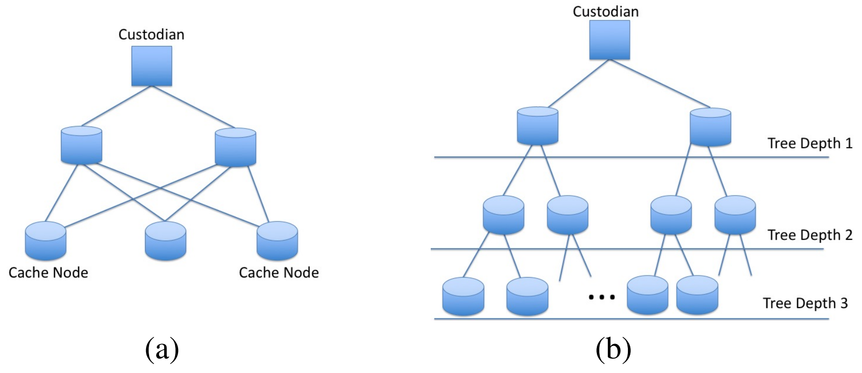


Figure 3. Hierarchical topologies: (a) layered; (b) tree.

Number of nodes, N	Cache size, B	Zipf parameter, α	Topology	Number of files, C	Custodian cost	Exploration rate	Link cost
10	10	0.8	layered	100	100	0.05	1

Table 3. Reference setup for experiments

7.1.1. Optimal Exploration Rate

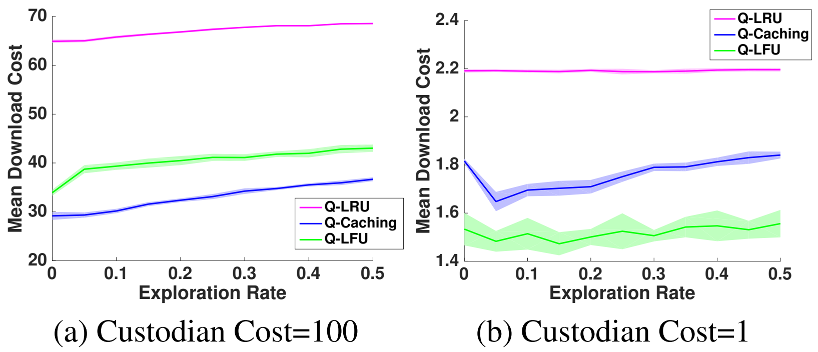
Next, we study the impact of the exploration rate on the metrics of interest. To this aim, we consider a network of 15 nodes, and vary the exploration rate. The other parameters are shown in Table 3.

We consider a layered topology with exogenous arrivals occuring at every node. Recall that exploration is necessary under Q-routing in order to learn the distribution of cost-to-go. This cost-to-go is used for routing decisions by Q-routing and for caching decisions by Q-caching, i.e., when caching is performed using LFU or LRU, the cost-to-go is used only for routing decisions.

According to Figure 4(a), when the custodian cost is 100, the optimal exploration rate is 0 and 0.05 for Q-caching and 0 for Q-LFU. When the custodian cost is 1, the optimal exploration rate is roughly 0.05 for Q-caching and 0.15 for Q-LFU as shown in Figure 4(b). These results shows that is worth to do a little *exploration* when all links has similar costs. When there are different costs, is better *exploit* the optimal routes. This happens because different costs implies in more *stable* topologies.

Figure 4(a) also shows that Q-caching outperforms Q-LFU when the custodian cost is 100, while Figure 4(b) shows the opposite when the custodian cost is one. This means that the Q-caching is *cost-aware* as shown in Section 7.1.3.

Note that Q-LRU is not sensitive to the exploration rate. We believe that the estimates of the cost-to-go are not very helpful under LRU due to the high content churn associated to such a policy, which causes high variability in the distribution of items in the caches over time. This variability precludes the quick convergence and gains of Q-routing.



(a) Custodian Cost=100 (b) Custodian Cost=1

Figure 4. Impact of exploration rate on the Mean Download Cost for two differents custodian costs

7.1.2. Q-caching Increases Space Diversity

Next, we indicate that Q-caching increases space diversity, and as a consequence, decreases mean download cost. To this aim, we consider a tree topology with l levels and $2^{l+1} - 1$ nodes. The remaining parameters are shown in Table 3. Figure 5 shows how the mean download cost varies as a function of the tree depth. As the tree depth increases, the mean download cost decreases more significantly under Q-caching as opposed to other strategies. Because Q-caching considers cost as well as popularity, the state of the neighbors impacts caching decisions. Thus Q-caching might not store a popular content if it is easy to retrieve from a neighbor. This increases space diversity because different replicas tends to be distributed all over the network, allowing requests to opportunistically download content from within the network and avoiding requests to the custodian, assumed to be associated to the most costly link in the system.

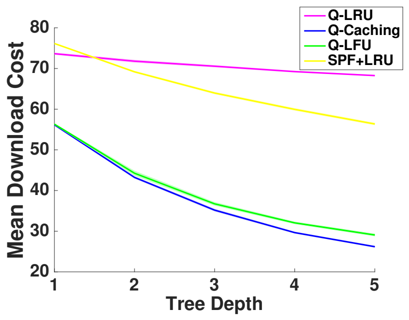
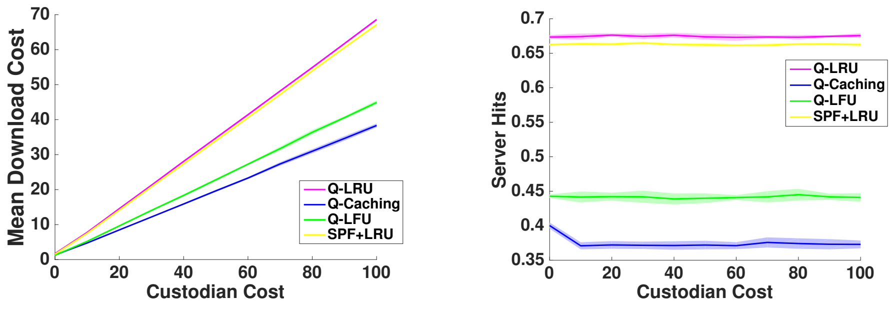


Figure 5. Impact of tree depth on mean download time

7.1.3. MEC is Cost-aware

Figure 6(a) shows the impact of the costs of accesses to the custodian on the mean download cost. We consider the reference setup of Table 3, with custodian cost varying between 0 and 100. Although the mean download cost increases linearly with respect to the cost for all the strategies considered, the slope of Q-caching is smaller than its counterparts. This indicates that Q-caching, by being cost-aware, can gracefully handle changes in custodian costs. Figure 6(b) shows how the number of server hits varies as a function of the custodian cost. While Q-caching adjusts itself to account for cost changes, the other strategies considered are cost-oblivious and maintain the same server hit rate irrespective of server costs.

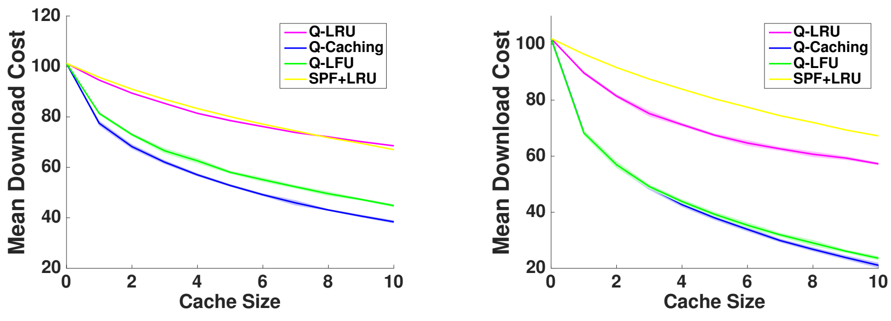


(a) (b)

Figure 6. Impact of custodian cost

7.1.4. Impact of Request Locality

Next, our goal is to study the impact of request locality on mean download cost. We consider the reference setup shown in Table 3. In Figures 7(a) and 7(b) we consider the cases where exogenous requests arrive at every node and only at the leaves, respectively. As Figure 7(a) shows, Q-caching is better than Q-LRU, and slight better than Q-LFU, when there are exogenous arrivals at every node. In contrast, Figure 7(b) shows that Q-LFU is slightly better than Q-caching when there are exogenous arrivals only at the edge. This is because, due to symmetry, the cost-to-go associated to items which are not stored in the leaves is roughly the same for all nodes. Therefore, the gains of Q-caching due to heterogenous cost-to-go values are not leveraged.



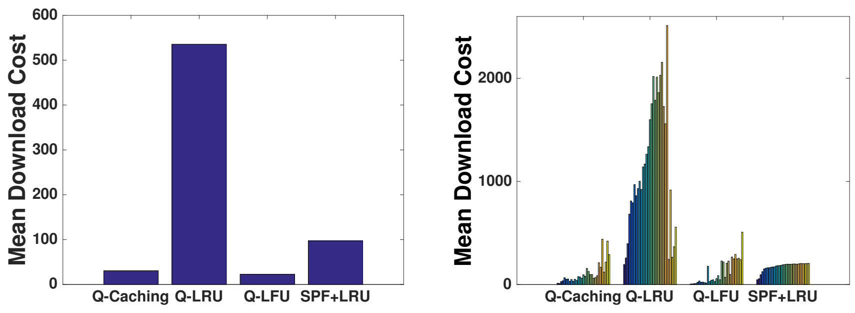
(a) arrivals at every node (b) arrivals only at the edge (leaves)

Figure 7. Impact of request locality

7.2. RNP Topology and Youtube Traces

To validate the strategies proposed, real network topology and video traces are used as described in Section 6. The video custodian is placed at João Pessoa. Every point of presence of the RNP network is associated to a cache-router with capacity to store up to three videos. All videos have unit size. The cost of a link is inversely proportional to its bandwidth. We set 20 Gbps as the reference bandwidth, meaning that its cost is normalized to 1. Henceforth, we consider only normalized costs, referred to simply as *costs*. The custodian has a link with capacity of 100 Mbps, i.e., its cost is 200.

Figures 8(a) and (b) show the results obtained using the RNP topology and the Youtube trace-driven simulations. Figure 8(a) shows the mean download cost to retrieve all files for the four considered strategies. As the figure indicates, Q-LFU is slightly better than Q-caching, and Q-LRU is the worst strategy. Figure 8(b) shows the mean download cost for each video. As in Figure 2, videos are sorted according to the total number of views. Under Q-LRU, the mean download cost is high for all videos. Under Q-caching and Q-LFU, the popular videos have smaller minimum download costs. Under Q-caching, 26 videos have mean download costs lower than 200, in agreement with the hypothesis that Q-caching increases space diversity. Under Q-LFU, in contrast, only 20 videos have mean download cost lower than 200. Under SPF+LRU, the four most popular videos have mean download costs lower than 200. The mean download cost of all other videos is roughly equal to 200.



(a) averaged over all files (b) per file

Figure 8. Mean download costs (RNP topology and Youtube video traces)

Figure 9 shows the evolution of the mean download cost over time for the four considered algorithms. This figure shows although the mean download cost of Q-LRU widely varied during the simulation period, SPF+LRU showed a better and more stable behavior. As previously discussed, this occurs because LRU yields high content volatility in the caches, not allowing Q-routing to properly converge. Under LRU, every new request to a content that is not cached leads to an eviction, naturally producing more churn, which needs to be coped by Q-routing. When Q-routing is combined with MEC (Q-caching) and LFU (Q-LFU), in contrast, a popular content might never show up in the miss stream. This feature promotes stabilization.

Note that the convergence of Q-caching is slower than that of Q-LFU. Nonetheless, after convergence the mean download costs of Q-caching and Q-LFU are 4 and 10, respectively. This means that depending on the dynamics of the content catalog and of the popularity of contents, Q-caching might be preferred over Q-LFU.

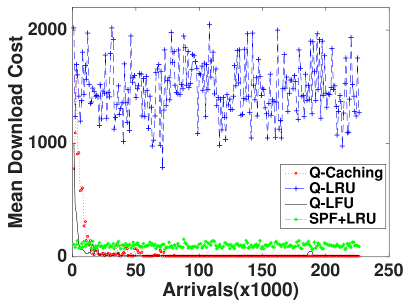


Figure 9. Evolution of the mean download cost over time

8. Related Work

The role of routing and the interplay between routing and caching in ICNs is discussed in [Yi et al. 2014, Chiocchetti et al. 2012, Chiocchetti et al. 2013, Rossini and Rossi 2014]. As such previous work suggests, jointly solving the routing and caching problems might lead to significant performance gains. However, the joint analysis of these two aspects is non-trivial, and taking them into account together in a simple manner was still an open problem. In this paper, we propose to use the cost-to-go provided by Q-routing for caching decisions, integrating routing and caching in a simple way.

When determining the steady state occupancy of the caches, one approach consists of searching for a fixed-point solution which is compatible with the routing and caching decisions [Tortelli et al. 2016]. We believe that the solutions proposed in this paper are also amenable to analysis through adaptations of the models proposed in [Rosensweig et al. 2010].

Efficient content placement in ICN with TTL-caches has been studied in [Domingues et al. 2013]. [Dabirmoghaddam et al. 2014, Fayazbakhsh et al. 2013] argued that *edge-caching* produces most of the gains of *everywhere-caching*. Our results suggest that Q-caching benefits from everywhere-caching because it increases content diversity.

The use of reinforcement learning (RL) for decision making in ICNs has been considered in other works. [Chiocchetti et al. 2012] studied the classical RL tradeoff between exploitation and exploration and then proposed Inform [Chiocchetti et al. 2013], which combines Q-routing with LRU for improved performance. [Bastos et al. 2015] also propose the use of RL for routing decisions. In this work, in contrast, we propose the use of RL for routing and caching decisions, in an integrated fashion. To the best of our knowledge this paper is the first to leverage information provided by reinforcement-learning based routing algorithms for caching decisions.

9. Discussion

Next, we discuss some of the simplifying assumptions considered in this work, together with possible directions for future work.

Forward and Backward Routes: In this paper, we focused on the forward (upstream) routes, namely the routes between requestors and custodians. We leveraged the opportunistic encounters between requests and replicas of the content closer to users. The backward (downstream) routes were assumed to be the same as the forward routes, and the zero download delay (ZDD) assumption was considered. If backward routes may be distinct from forward routes, additional flexibility might be gained to determine how to route the content to users, so as to strategically place new replicas where needed.

Routing and Caching Utilities: We assumed that the utilities associated to routing and caching decisions are the same. However, in general they might be associated to distinct utilities. In such case, additional control messages might be required. The gains, however, might compensate such overhead specially if routing costs (e.g., due to congestion) are very distinct from content costs (e.g., due to the availability of custodians for certain rare or unpopular content).

10. Conclusion

Caching and routing are two of the building blocks of ICNs. In this paper, we showed how to leverage information provided by Q-routing in order to make caching decisions. The proposed solution, Q-caching, is simple and flexible, allowing for different utilities and metrics of interest to be taken into account when coupling caching and routing for increased performance. We numerically investigated the effectiveness of Q-caching, and contrasted it against different schemes, including Q-routing coupled with LRU (the basis for the state-of-the-art Inform). We have found that in the scenarios investigated Q-caching is either better or competitive againts the considered counterparts. Future work consists of establishing the conditions under which Q-caching is optimal, and studying its convergence properties.

Acknowledgment: This work was supported in part by CNPq, PETROBRAS and CAPES. Wouter Caarls was funded by a Science Without Borders scholarship (CSF/Bolsa Jovem Talento/CAPES).

References

Afanasyev, A., Burke, J., Zhang, L., Claffy, K., Wang, L., Jacobson, V., Crowley, P., Papadopoulos, C., and Zhang, B. (2014). Named Data Networking. *ACM SIGCOMM CCR*, 44(3):66–73.

Bastos, I., Sousa, V., and Moraes, I. (2015). Uma estratégia de encaminhamento de pacotes baseada em aprendizado por reforço para redes orientadas a conteúdo. In *SBRC*.

Boyan, J. A. and Littman, M. L. (1994). Packet routing in dynamically changing networks: A reinforcement learning approach. In Cowan, J., Tesauro, G., and Alspector, J., editors, *Advances in Neural Information Processing Systems 6*, pages 671–678. Morgan-Kaufmann.

Chiocchetti, R., Perino, D., Carofiglio, G., Rossi, D., and Rossini, G. (2013). Inform: a dynamic interest forwarding mechanism for information centric networking. In *ICN*, pages 9–14. ACM.

Chiocchetti, R., Rossi, D., Rossini, G., Carofiglio, G., and Perino, D. (2012). Exploit the known or explore the unknown?: Hamlet-like doubts in ICN. In *ICN*, pages 7–12. ACM.

Dabirmoghaddam, A., Barijough, M. M., and G-L-Aceves, J. (2014). Understanding optimal caching and opportunistic caching at the edge of information-centric networks. In *ICN*, pages 47–56. ACM.

Dai, H., Liu, B., Chen, Y., and Wang, Y. (2012). On pending interest table in named data networking. In *Proceedings of the Eighth ACM/IEEE Symposium on Architectures for Networking and Communications Systems*, ANCS '12, pages 211–222, New York, NY, USA. ACM.

Dehghan, M., Massoulie, L., Towsley, D., Menasche, D., and Tay, Y. (2016). A utility optimization approach to network cache design. In *INFOCOM*.

Domingues, G. d. M. B., Leão, R. M. M., Mensché, D. S., et al. (2013). Enabling information centric networks through opportunistic search, routing and caching. In *SBRC*.

Fayazbakhsh, S. K., Lin, Y., Tootoonchian, A., Ghodsi, A., Koponen, T., Maggs, B., Ng, K., Sekar, V., and Shenker, S. (2013). Less pain, most of the gain: Incrementally deployable icn. In *ACM SIGCOMM CCR*, volume 43, pages 147–158. ACM.

G-L-Aceves, J. (2015). A fault-tolerant forwarding strategy for interest-based information centric networks. *IFIP Networking 2015*.

INEP (2013). Sinopses estatísticas da educação superior 2013 - graduação `http://portal.inep.gov.br/superior-censosuperior-sinopse`. Accessed December 25, 2015.

Jacobson, V., Smetters, D. K., Thornton, J. D., Plass, M. F., Briggs, N. H., and Braynard, R. L. (2009). Networking named content. In *CONEXT*, pages 1–12. ACM.

Liu, Z., Nain, P., Niclausse, N., and Towsley, D. (1997). Static caching of web servers. In *Photonics West'98 Electronic Imaging*, pages 179–190. International Society for Optics and Photonics.

Melazzi, N. B., Bianchi, G., Caponi, a., and Detti, a. (2014). A general, tractable and accurate model for a cascade of LRU caches. *IEEE Communications Letters*, 18(5):877–880.

Mnih, V. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

Neves, T., Drummond, L. M., Ochi, L. S., Albuquerque, C., and Uchoa, E. (2010). Solving replica placement and request distribution in content distribution networks. *Electr. Notes in Disc. Math.*, 36:89–96.

Rosensweig, E. J., Kurose, J., and Towsley, D. (2010). Approximate models for general cache networks. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–9. IEEE.

Rossini, G. and Rossi, D. (2014). Coupling caching and forwarding: Benefits, analysis, and implementation. In *ICN*, pages 127–136. ACM.

Sitaraman, R. K., Kasbekar, M., Lichtenstein, W., and Jain, M. (2014). Overlay networks: An akamai perspective. *Advanced Content Delivery, Streaming, and Cloud Services.*

Sutton, R. and Barto, A. (1998). *Reinforcement Learning: An Introduction.* MIT Press, Cambridge, MA.

Tortelli, M., Rossi, D., and Leonardi, E. (2016). Model-graft: Accurate, scalable and flexible analysis of cache networks.

UFRJ (2015). Youtube traces of performance evaluation channel: `https://www.youtube.com/user/ADUFRJ20121`. Traces available here: `https://goo.gl/NwHf0g`.

Wooster, R. P. and Abrams, M. (1997). Proxy caching that estimates page load delays. In *WWW*, pages 977–986, Essex, UK. Elsevier Science Publishers Ltd.

Yi, C., Abraham, J., Afanasyev, A., Wang, L., Zhang, B., and Zhang, L. (2014). On the role of routing in named data networking. In *ICN*, pages 27–36. ACM.