This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2018.2855160, IEEE Transactions on Cloud Computing

Risk-Averse Caching Policies for YouTube Content in Femtocell Networks using Density Forecasting

William Hoiles, S M Shahrear Tanzil, and Vikram Krishnamurthy, *Fellow, IEEE*

Abstract—The paper presents risk-neutral and risk-averse caching policies that can be deployed in a femtocell network with limited storage capacity to reduce the time delay of servicing content requests. The caching policies use a forecasting algorithm to estimate the cumulative distribution function of content requests based on the content features. Given the cumulative distribution function, a mixed-integer linear program is used to compute where to cache content in the femtocell network. The caching policies account for the uncertainty associated with estimating the content requests using the coherent Conditional Value-at-Risk (CVaR) measure. For a large number of content, a risk-neutral caching policy is constructed that accounts for both the content features and routing protocol that only requires the evaluation of a unimodular linear program. Using data from YouTube (comprising 25,000 videos) and the NS-3 simulator, the caching policies reduce the delay of retrieving content in femtocell networks compared with industry standard caching policies. Specifically, a 6% reduction in delay is achieved by accounting for the uncertainty, and a 60% reduction in delay is achieved if both the uncertainty and femtocell routing protocol are accounted for compared to the risk-neutral caching policy that neglects the routing protocol.

Index Terms—Conditional Value-at-Risk, risk-averse caching, Femtocell network, unimodular linear program, conformal prediction, machine learning.

1 Introduction

Caching content in femtocell networks is an attractive method for alleviating network congestion resulting from increased video on demand streaming. In future 5G mobile network protocols, proactive caching will be an integral component to reduce the network congestion and the delay of delivering content to users. Proactive caching is integral as a substantial portion of the network congestion results from transferring popular video content to users. If the video content is cached locally, then both network congestion and the delay of delivering content to users is reduced. To determine which content to cache requires a caching policy that accounts for both future content requests, load balancing, and the routing protocol to deliver content to users that is used in the femtocell network. This paper presents two risk-neutral and two risk-averse caching policies that account for the uncertainty of predicting the future popularity of content, the content routing protocol, and load balancing in the network.

Given the main source of traffic in femtocell networks is video content, several methods have been proposed for estimating video content requests. These methods fall into two categories, namely, time-series or content feature based. Time-series methods use the historical content requests to predict the future content requests. Multivariate linear models [1], pure birth stochastic process [2], and ordinary differential equations [3] have all been used for constructing time-series methods for estimating content requests. A limitation with time-series methods is that they can only be used to predict the content requests of posted content. Content feature based methods use the uploader, textual, and image features of the content to predict the content requests. These methods include multivariate linear regression [4], Markov clustering [5], and extreme learning machines [6]. All these methods provide point forecasts (expected value) for the content requests. No estimate of the confidence interval, prediction interval, or measure of the uncertainty associated with the predicted content requests is provided. In this paper we construct a conformal prediction algorithm for estimating the cumulative distribution function of the content requests. The key idea of the conformal prediction algorithm is to first group content, then for each group assume that the resulting error between the point forecasts and the actual content requests are generated from the same cumulative distribution function. The estimated cumulative distribution function provides a complete description of the uncertainty associated with the estimated content requests. Both the confidence interval and prediction interval of the content requests can be constructed from the estimated probability distribution. Note that density forecasting [7], [8] in the time-domain is typically referred to as prequential forecasting [9] in the economics literature.

Having estimated the future content requests, the aim is to optimally cache the content throughout the femtocell network to minimize the delay to transfer the requested content to the users. This requires that content is cached where it is likely to be requested, and to optimally transfer

- *W. Hoiles and S. Tanzil are with the Department of Electrical and Computer Engineering, University of British Columbia, Vancouver V6T 1Z4, Canada (e-mail: whoiles@ece.ubc.ca;stanzil@ece.ubc.ca).*
- *V. Krishnamurthy is with the Department of Electrical and Computer Engineering, Cornell Tech, Cornell University, New York 10044, USA (e-mail: vikramk@cornell.edu).*
This research was supported by a Schmidt Sciences grant.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2018.2855160, IEEE Transactions on Cloud Computing

content throughout the network to serve user requests. The methods in [10], [11], [12] account for user specific downloading delay and wireless channel fading gain to determine where to cache content and which user to connect to which femtocell access point. These caching methods can be considered as dynamic cache replacement methods as they adapt the cache based on the users ability to connect to different femtocell access points. However, a limitation with the caching methods [10], [11], [12] is that they do not account for the routing protocol used in the backhaul network to retrieve content that is not cached in the femtocell access points. In [13] cooperative caching is used to account for the backhaul link bandwidth in the network to cache content throughout the network. Once the content is cached in the femtocell network, then content and load aware routing methods are used to transfer the requested content to users [14], [15], [16]. The aim of these routing methods to ensure load balancing occurs throughout the network to minimize the delay of transferring content to users.

A common theme with the caching methods [10], [11], [12], [13] is that they contain three distinct steps. First, a point estimate of the content popularity is performed. Second, the content is cached in the network to minimize the delay of transferring content to users. Third, given the cached content, a routing method is used to deliver the content to the users. Notice that the routing method does not affect how the content is cached in the network. Additionally, the above methods are not risk-averse–that is, the point estimate of the content requests is equivalent to computing the expectation of the requests using the forecasted content request density. An issue with using the point estimate is that it does not provide a measure of the uncertainty associated with estimating the future content requests. As such, no probabilistic guarantees can be made on the network operating characteristics (e.g. downloading delay, bit-error-rate, energy consumption, cache miss ratio) for a selected caching decision.

In this paper, instead of minimizing the expected downloading delay, we replace the additive expectation operator with a more general subadditive risk operator to make caching decisions to optimize network performance. Specifically, we use the Conditional Value-at-Risk (CVaR) risk operator to construct the caching policies in the femtocell network[1]. There are two important properties of CVaR that make it useful for performing caching decisions to reduce delay. First, CVaR accounts for the minimal probability of a substantial network delay for a caching decision where the total delay probability distribution is asymmetric. Second, CVaR is a coherent risk measure–that is, CVaR is monotonic, subadditive, positive homogeneous, and translation invariant. The monotonic property states that if the delay of a caching decision D_1 is always less than another caching decision D_2 almost surely, then the risk of selecting D_1 is always less than D_2. Additionally, the subadditive property guarantees that the risk of using two caching decisions D_1 and D_2 is always less than or equal to the risk associated with using D_1 and D_2 separately. Since CVaR is a coherent risk measure, the optimization of CVaR results in a convex optimization problem. As we show, the optimization of CVaR to confidently reduce the network delay while accounting for the routing protocol results in a convex mixed-integer linear program.

1. CVaR is one of the "big" developments for risk-averse decision making in mathematical finance; see [17], [18], [19].

The paper is organized as follows. The system model of the femtocell network is provided in Sec.2 where Table 1 provides a summary of the parameters used throughout the paper. In Sec.3 dynamic cache replacement policies are discussed. In Sec.4 we discuss four static caching policies, two which are risk-neutral and two which are risk-averse. Specifically, in Sec.4.1 and 4.2 we construct risk-neutral static caching policies for the femtocell network. The term risk-neutral is used to indicate that these methods use point estimates of the content requests–that is, they do not account for the uncertainty associated with estimating the future content requests. An useful outcome of Sec.4.2 is that the static caching policy, which accounts for content requests, cache size, bandwidth, load, and content routing, only requires the solution to a unimodular linear program. In Sec.4.4 and 4.5, risk-averse caching policies are constructed based on the risk-neutral caching policies in Sec.4.1 and 4.2. The main idea is that the uncertainty associated with estimating content requests is accounted for using the Conditional Value-at-Risk (CVaR) measure. In Sec.5 a novel content request conformal prediction algorithm is constructed based on the extreme learning machine and CVaR optimization. The performance of the risk-neutral and risk-averse caching policies are evaluated using real-world data from the YouTube social network in Sec.6. The results show that a 6% reduction in the average delay can be achieved if the uncertainty of the content requests is accounted for, and a 60% reduction in average delay is achieved if both the uncertainty and femtocell routing protocol are accounted for compared to the risk-neutral caching policy that neglects the routing protocol. Therefore, it is essential to account for both the uncertainty of predicting the content requests and routing when performing caching decisions.

2 System Model

In this section we introduce the system model of the femtocell network and users, and introduce the mathematical notation that will be used throughout the paper to formulate the risk-neutral and risk-averse caching policies.

We consider a heterogenous Long-Term Evolution (LTE) wireless femtocell network [20], [21]. The network contains wireless nodes (e.g. base stations, femtocell access points, femtocell gateway) and a core network as illustrated in Fig. 1. The nodes in the network are defined by the set $\mathcal{V} = \{1, \ldots, V\}$. Each wireless node $v \in \mathcal{V}$ contains a physical cache of size S_v which stores the cached content. Additionally, the bandwidth between nodes is given by the parameter $b_{ij} \in \mathbb{R}_+$ for $i, j \in \mathcal{V}$. If the nodes i and j have no direct communication link then their bandwidth $b_{ij} = 0$. In the LTE network, the bandwidth between nodes are typically heterogeneous and they are composed of wired, fiber-optic, and wireless links [22]. For example, the bandwidth between base stations and femtocell gateway nodes to the core network are on the order of several GB/s (fiber-optic). The link capacity of base stations and femtocell

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2018.2855160, IEEE Transactions on Cloud Computing

TABLE 1
Notation for Risk-Averse Caching

Parameters	Definition
$F(\cdot)$	cumulative distribution function
$\hat{F}(\cdot)$	empirical cumulative distribution function
$p(\cdot)$	probability mass function
α	confidence level
t	time
LTE Network Parameters	
$\mathcal{V}$	set of nodes $\{1, \ldots, V\}$
$\mathcal{V}_d$	destination nodes with $\mathcal{V}_d \subseteq \mathcal{V}$
S_v	cache size of node $v \in \mathcal{V}$
$C(t)$	cached content indicator matrix
$c_v(t)$	cached content indicator vector for node $v \in \mathcal{V}$
$n_v(t)$	load at node $v \in \mathcal{V}$
q_v	the request-queue-time at node $v \in \mathcal{V}$
A_{ijf}	weight between nodes $i, j \in \mathcal{V}$ for $f \in \mathcal{F}$
b_{ij}	bandwidth between nodes $i, j \in \mathcal{V}$
l_{ij}	latency between nodes $i, j \in \mathcal{V}$
δ_{ijdf}	shortest-path indicator for $i, j, d \in \mathcal{V}$ and $f \in \mathcal{F}$
$d_{vf}(t)$	content retrieval delay at $v \in \mathcal{V}_d$ for $f \in \mathcal{F}$
Content Parameters	
$\mathcal{F}$	set of content $\{1, \ldots, F\}$
$\mathcal{D}(t)$	dataset of content features and requests at time t
$\mathcal{G}$	content groups $\{1, \ldots, G\}$
f	content index $f \in \mathcal{F}$
s_f	size of content $f \in \mathcal{F}$
$g_{vf}(t)$	group association of content $f \in \mathcal{F}$ at $v \in \mathcal{V}_d$
$y_{vf}(t)$	request count for content $f \in \mathcal{F}$ at $v \in \mathcal{V}_d$
x_f	feature vector of content $f \in \mathcal{F}$
$\hat{g}_{vf}(t)$	estimated group association of content $f \in \mathcal{F}$ at $v \in \mathcal{V}_d$
$\hat{y}_{vf}(t)$	estimated request count for content $f \in \mathcal{F}$ at $v \in \mathcal{V}_d$
μ_g	mean vector of group $g \in \mathcal{G}$
Σ_g	covariance matrix of group $g \in \mathcal{G}$
$\boldsymbol{\beta}$	neuron weights
$\boldsymbol{\theta}$	neuron transfer function parameters
L	number of neurons

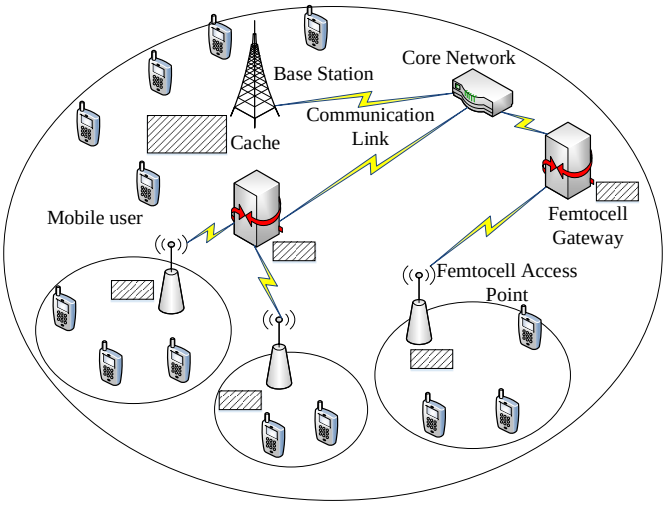


Fig. 1. Schematic of a Long-Term Evolution (LTE) wireless network. The LTE wireless network is composed of femtocell access points, base stations, femtocell gateways, and a core network. The femtocell access point, femtocell gateway, and base stations all contain physical caches that can store content. Mobile users are connected to either the base station or the femtocell access point through a low-bandwidth connection. Femtocell access points are connected to the femtocell gateway which is then connected to the core network. Note that the base stations and femtocell gateway nodes do not communicate directly with each other. The core network is connected to the content server over the wide area network.

access points to mobile users are typically on the order of 1-100 MB/s (wireless). And the link capacity of femtocell access points to the femtocell gateways are on the order of 100 MB/s (wired connection) [23]. The content server stores all the content that can be requested by users. The core network communicates with the content server over the wide area network. Note that the content server is typically comprised of a commercial content distribution network (CDN) such as Akamai, Amazon CloudFront, Azure CDN or dedicated telco CDN that is maintained by a wireless network operator [24].

When a mobile user connects to the LTE network, the LTE network protocol established the communication link between the mobile user and either a femtocell access point or base station based on the minimal signal-to-noise ratio of the wireless channel [25], [26]. The set of content that can be requested by mobile users is denoted by $\mathcal{F} = \{1, 2, \cdots, F\}$. The size of each content is denoted by $s_f \in \mathbb{R}_+$ for $f \in \mathcal{F}$. When a femtocell access point or base station receives a user request for content $f \in \mathcal{F}$, the node that receives the request will retrieves the content from the LTE network. The set of nodes (femtocell access points and base stations) that receive users' requests is denoted by $\mathcal{V}_d$ where $\mathcal{V}_d \subset \mathcal{V}$. The delay between when the user request was received and when the content is delivered to the user is known as the content retrieval delay. The content retrieval delay for content $f \in \mathcal{F}$ requested at node $v \in \mathcal{V}_d$ at time t is denoted by $d_{vf}(t)$. The content retrieval delay $d_{vf}(t)$ depends on where the content is cached in the network, the load of each node, bandwidth between nodes, network-layer protocol, and link-layer protocol of the LTE network. The load $n_v(t)$ of each wireless node $v \in \mathcal{V}$ is the total number of content requests the wireless node is currently processing. If a user requests content $f \in \mathcal{F}$ from the wireless node $v \in \mathcal{V}_d$ and node v has the content cached, then the content retrieval delay $d_{vf}(t) = s_f q_v n_v(t)$ where q_v is the request-queue-time of node v and s_f is size of the content f. The request-queue-time q_v of node $v \in \mathcal{V}$ provides the average time to process a single packet/byte request. Note that if node v does not contain the requested content, then it must be retrieved by another node in the network or from the content server.

The goal of the LTE network is to minimize the total content retrieval delay to serve all user requests in the network. To achieve this objective, each node in Fig.1 contains a physical cache and a cache manager. The cache manager of each node controls the content that is cached at the node [27]. The currently cached content at node $v \in \mathcal{V}$ is given by the cached content indicator vector $c_v(t) \in [0, 1]^F$ where F is the total number of contents that users can request in the LTE network. The cache of each node is composed of a static segment and a dynamic segment. The content in the static cache does not change for a time interval ΔT and is associated with the slow-time scale caching decisions. The content in the dynamic cache changes as a function of the user requests and is associated with the fast-time scale. The cache manager controls the cache replacement policy for the static and dynamic caches to minimize content retrieval delay $d_{vf}(t)$. Specifically, the cache manager at each node:

1) runs the cache replacement policy to minimize request delays.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2018.2855160, IEEE Transactions on Cloud Computing

2) forwards content requests to neighbouring nodes or the content server if the content is not cached locally.
3) records the number of requests $y_{vf}(t)$ and feature vector x_f for content $f \in \mathcal{F}$ at $v \in \mathcal{V}$.

It is assumed that each node in the LTE network has computational resources equivalent to a standard desktop computer to allow the operation of the cache manager. Given that the cache size S_v, only a small fraction of all the content $\mathcal{F}$ can be cached locally (except for the content server which contains all contents). If the content is not cached locally, the content is retrieved from another node that minimizes the content retrieval delay $d_{vf}(t)$.

To perform cache replacement of the static cache, the cache manager records the request statistics $y_{vf}(t)$ and feature vector x_f for content $f \in \mathcal{F}$ at node $v \in \mathcal{V}$. For video content, the feature vector comprises information related to the thumbnail and title of the video, as well as information regarding the user that uploaded the video such as number of subscribers. The complete set of content features and requests at each node is contained in the dataset

$$\mathcal{D}(T) = \{\{x_f, y_{vf}(t), g_{vf}(t)\} : v \in \mathcal{V}, f \in \mathcal{F}, t \in [0, T]\} \tag{1}$$

where T is the total time the content $\mathcal{F}$ has been available to users. The parameter $g_{vf}(t)$ in (1) is the group association of content $f \in \mathcal{F}$ at node $v \in \mathcal{V}$. The possible groups that content can be associated with is denoted by $\mathcal{G} = \{1, 2, \ldots, G\}$. The cache manager uses the information in $\mathcal{D}(T)$ to estimate the future number of requests of content for both new content and previously cached content that users have requested.

Given the LTE network parameters (cache size, bandwidth between nodes, network-layer protocol, and link-layer protocol) and the content parameters (feature vector, request count, group association), the aim is to design caching policies to minimize the cumulative content retrieval delay

$$d(T) = \sum_{k=1}^{K_t} \sum_{v=1}^{V} \sum_{f=1}^{F} d_{vf}(t_k) \tag{2}$$

where K_t is the total number of content requests in the time interval $[0, T]$, and $t_k \in [0, T]$ denotes the time of each of the $k \in \{0, 1, \ldots, K_t\}$ content requests. To minimize the delay requires a method to estimate the future content requests, and a method to cache popular content based on the request estimates and routing protocol to deliver content to users in the LTE network. In this paper we construct a content request density forecasting method, and both risk-neutral and risk-averse cache replacement policies to minimize the LTE network delay.

3 Dynamic Caching Policies

The cache of each node in the LTE network, illustrated in Fig. 1, is composed of a dynamic segment and a static segment. This section discusses dynamic caching policies, while Sec.4 discusses static caching policies. To give more perspective, recall from Sec.2 that the content in the dynamic segment changes as a function of the user requests, while the content in the static segment changes on a time interval ΔT that is significantly larger then the time-scale of individual content requests. The content in the dynamic and static segments of the cache are controlled by the dynamic caching policy and static caching policy respectively. Here we briefly discuss dynamic caching policies that can be used in the LTE network. In Sec.4 we present static caching policies that are used in combination with the dynamic caching policies presented in this section.

A schematic of the interaction of the dynamic and static cache is illustrated in Fig. 2. There are three possible scenarios that can occur depending on where the requested content is cached in the network. In the first scenario Fig. 2, the content is transferred from the static cache to the user and no change in the dynamic cache occurs. In the second scenario in Fig. 2, the content is transferred from the dynamic cache to the user. Additionally, the content in the dynamic cache will be adjusted according to the dynamic cache replacement policy. In the third scenario in Fig. 2, the content must first be transferred to the dynamic cache from another node in the network, and then transferred to the user. Additionally, the content in the dynamic cache will be adjusted and evicted according to the dynamic cache replacement policy. In all of the three scenarios, the content in the static cache remains unchanged, and identical content is not simultaneously available in both the static and dynamic caching segments. The content in the static cache is only updated at a time interval ΔT after it was first initialized. The aim of the static caching segment is to store content that is expected to have a large number of user requests in the duration ΔT, while the dynamic cache stores other content requested by users.

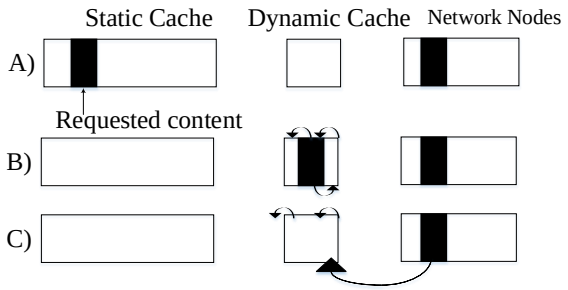


Fig. 2. Schematic of the interaction between the static and dynamic caching policies. The left box and the middle box refer to the static and dynamic cache of a single node while the right most box (network nodes) refers to the storage of all other nodes in the network. Three possible scenarios can be result from a content request at a node. Scenario (A) refers to the situation when the requested content is available in the static cache of the node. In this case, content in the static and dynamic cache remain unchanged. B) refers to the case when the requested content is available in the dynamic cache of the node. In this case, content in the dynamic cache are adjusted according to the dynamic cache replacement policy and content in the static cache remains unchanged. C) represents the situation when the requested content is unavailable in the static or dynamic cache of the node. In this case, the content will be retrieved from another network node and stored in the dynamic cache of the node where the request was received. This will cause the least frequently requested content in the node's dynamic cache to be evicted while the static cache remains unchanged.

Several dynamic caching policies exist which are based on users' real-time content requests including: Least-Recently-Used (LRU), Segmented Least-Recently-Used (SLRU) [28], Least-Frequently-Used, and Least-Frequently-Used with Dynamic Ageing and Adaptive Re-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2018.2855160, IEEE Transactions on Cloud Computing

placement Cache. The LRU cache replacement policy operates by maintaining an ordered cache where recently requested content will reside at the beginning of the cache (known as most recently used position). As user requests are processed by the node, the content in the dynamic cache that are not requested by users are shifted towards the end of the cache (least recently used position). If the requested content is not currently available in the cache, it will be retrieved from other nodes in the LTE network and stored in the most recently used position. All other content in the cache will be shifted towards the end of the cache with the content in the least recently used position being evicted from the cache. Variants of the LRU policy commonly used include the SLRU. In the SLRU cache replacement policy, the entire cache is divided into different segments with each segment associated with a priority or popularity level. When requested content is unavailable in the cache, the content will be retrieved and stored in the most recently used position of the lowest priority segment of the dynamic cache. Simultaneously, the content that is in the least recently used position of the lowest priority segment will be evicted from the cache. The main advantage of the LRU and SLRU dynamic cache replacement policies is that they do not require knowledge of the LTE network architecture or where the content is stored throughout the network. As such, these caching policies are straightforward to implement and are widely used in commercial distribution networks such as Facebook [28].

4 Risk-Neutral and Risk-Averse Static Caching Policies

This section presents four static caching policies (two risk-neutral and two risk-averse) and constitutes the main contribution of the paper.

The static cache replacement policy controls the content stored in the static cache of each node in the LTE network illustrated in Fig.1. The content in the static cache remains the same for a time interval ΔT that is significantly longer than the characteristic time-scale of individual content requests from users. The goal of the static caching policy is to cache content that is predicted to have a large number of requests in order to minimize the total content retrieval delay in the time interval ΔT. Given the parameters of the LTE network and the content dataset $\mathcal{D}$, this section presents two risk-neutral and two risk-averse static caching policies. The term risk-neutral is used for any policy that uses point estimates of the content requests and does not account for the uncertainty associated with predicting the content requests. These include the risk-neutral (RN) and risk-neutral and network-aware (RNNA) caching policiespresented in Sec.4.1 and 4.2. Risk-averse caching policies in contrast to the risk-neutral policies, account for the uncertainty associated with estimating the content requests y_{vf} for content f at node v. Here, the uncertainty associated with the content requests is accounted for using the coherent CVaR risk measure with a confidence level $\alpha \in [0,1]$ (as discussed in Sec.4.3). These include the risk-averse (RA) and risk-averse and network-aware (RANA) caching policiespresented in Sec.4.4 and 4.5. The RA and RANA policies can be viewed as generalizations of the RN and RNNA policies. Note that if we do not consider risk (e.g. risk-neutral) then the confidence level $\alpha = 0$.

4.1 Risk-Neutral (RN) Static Caching Policy

Let us assume we have the predicted number of requests for each content $f \in \mathcal{F}$ at node $v \in \mathcal{V}$, which we denote by $\hat{y}_{vf}$. Then, the content to be cached at each node can be selected by solving the following binary integer program

$$C^* \in \arg\min_{c_{vf}} \left\{ \sum_{f=1}^{F} \sum_{v=1}^{V} \hat{y}_{vf}(1 - c_{vf}) \right\}$$
$$\text{s.t.} \sum_{f=1}^{F} s_f c_{vf} \leq S_v \quad \text{for } v \in \mathcal{V}, \tag{3}$$

where $C^* \in [0,1]^{V \times F}$ and $c_{vf} \in [0,1]$ indicates if the content $f \in \mathcal{F}$ is cached at node $v \in \mathcal{V}$. The inequality constraint in (3) ensures that each node can cache files upto its associated cache size. Although (3) is a binary integer program which has complexity NP-complete, (3) can be solved with complexity $O(F\log(F))$ as each node merely caches the maximum number of content that are predicted to have the highest number of requests.

The RN caching policy (3) is used extensively in the literature [27], [29], [30]. The key feature of the risk-neutral caching policy is that it requires an accurate estimation of y_{vf}, namely, the number of requests for the content. The RN policy does not account for any aspects of the LTE network other than the cache size of each node. Additionally, (3) does not account for the uncertainty associated with estimating the number of content requests y_{vf}. Therefore, although (3) can be evaluated with low complexity $O(F\log(F))$, the total content retrieval delay is expected to be higher compared with static caching policies that account for the LTE network parameters, routing protocol, and the uncertainty associated with predicted content requests.

4.2 Risk-Neutral and Network-Aware (RNNA) Static Caching Policy

Here we construct RNNA static caching policy to optimally cache content given the predicted content requests $\hat{y}_{vf}$, the LTE network parameters (bandwidth, load at the nodes, request-queue-time, and cache size of each node), the network-layer protocol, and the link-layer protocol. The RNNA caching policy accounts for both the LTE network parameters and routing protocol, however neglects the uncertainty associated with predicted content requests $\hat{y}_{vf}$.

The (RNNA) static caching policy is given by the following binary integer program

$$C^* \in \arg\min_{C,k,\delta,r} \left\{ \sum_{f=1}^{F} \sum_{d \in \mathcal{V}_d} \sum_{i,j \in \mathcal{V}} \hat{y}_{df} A_{ijf} \delta_{ijdf} \right\}$$
$$\text{s.t. } c_{sf} \in [0,1], \quad k_{sdf} \in [0,1], \quad \delta_{ijdf} \in [0,1],$$
$$r_{sdf} \in [0,1], \quad T_{ij} \in \mathbb{Z}_+$$
$$\sum_{i \in \mathcal{V}} \delta_{sidf} - \delta_{isdf} = k_{sdf}, \quad \sum_{i \in \mathcal{V}} \delta_{didf} - \delta_{iddf} = -1,$$
$$\mathbf{1}\{b_{ij} = 0\} + \delta_{ijdf} \leq 1 \tag{4a}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2018.2855160, IEEE Transactions on Cloud Computing

$$\sum_{f=1}^{F} s_f c_{sf} \leq S_s, \quad \sum_{s=1}^{V} c_{sf} \geq 1 \tag{4b}$$
$$\sum_{s=1}^{V} k_{sdf} = 1, \quad \sum_{s=1}^{V} r_{sdf} = 1,$$
$$\sum_{f=1}^{F} \sum_{d \in \mathcal{V}_d} \delta_{ijdf} \leq T_{ij} \tag{4c}$$
$$r_{sdf} \leq k_{sd}^{f}, \quad r_{sdf} \leq c_{sf}, \quad r_{sdf} \geq k_{sdf} + c_{sf} - 1, \tag{4d}$$
$$\forall s \in \mathcal{V}, \quad \forall d \in \mathcal{V}_d, \quad \forall f \in \mathcal{F}.$$

In (4), $C^* \in [0,1]^{V \times F}$ indicates the content cached in the static cache of all nodes, $\mathcal{V}_d \subset \mathcal{V}$ are the destination nodes in the LTE network, and T_{ij} is a positive integer that indicates the maximum number of content transfer paths allowed between nodes i and j (e.g. congestion threshold). The destination nodes $\mathcal{V}_d$ communicate directly with the users and are comprised of the base stations and femtocell access points illustrated in Fig. 1. Given the predicted content requests $\hat{y}_{df}$, the objective function in (4) represents the total content retrieval delay over the time-interval $[t, t + \Delta T]$. Note that we have dropped the time-dependence from the parameters in (4) to improve readability. The parameter A_{ijf} in (4) is the edge weight of the LTE network for content $f \in \mathcal{F}$ and is equal to

$$A_{ijf} = \begin{cases} s_f(l_{ij} + q_j n_j) \text{ if } i \neq j \\ s_f n_j \quad \text{otherwise} \end{cases} \tag{5}$$

where s_f is the size of content f, l_{ij} is the latency between nodes $i \in \mathcal{V}$ and $j \in \mathcal{V}$, q_j is the request-queue-time of node $j \in \mathcal{V}$. The latency l_{ij} (second per byte) between nodes is a function of the bandwidth b_{ij}, the network topology, the network-layer protocol, and the link-layer protocol used in the LTE network. The parameter δ_{ijdf} indicates if nodes $i, j \in \mathcal{V}$ are used to transfer the content $f \in \mathcal{F}$ to the destination node $d \in \mathcal{V}_d$. k_{sdf} indicates if source node $s \in \mathcal{V}$ is used to retrieve content $f \in \mathcal{F}$ for the destination node $d \in \mathcal{V}_d$. The parameter $r_{sdf} = k_{sdf} c_{sf}$ indicates if the source node $s \in \mathcal{V}$, used by destination node $d \in \mathcal{V}_d$ to retrieve content f, currently has the requested content cached.

The RNNA caching policy (4) optimally selects the cache C^* to minimize the total content retrieval delay while accounting for the LTE network parameters and predicted content requests. Additionally, RNNA accounts for the shortest-path routing used to transfer content throughout the network to serve user requests. The path constraints (4a) ensures that the δ_{ijdf} defines the shortest-path from source node $s \in \mathcal{V}$ to destination node $d \in \mathcal{V}_d$ for transferring content $f \in \mathcal{F}$. The caching constraints (4b) ensure that the files cached at each node do not exceed the nodes cache size, and that atleast one instance of each content $f \in \mathcal{F}$ is cached in the LTE network. The link congestion constraint (4c) ensures that the number of content transferred over the link between node $i \in \mathcal{V}$ and node $j \in \mathcal{V}$ satisfies the congestion threshold T_{ij}. The source constraints (4d) ensures that only one source node $s \in \mathcal{V}$ is used to transfer content f optimally to the destination node $d \in \mathcal{V}_d$. Additionally, the source constraints ensure that the source node s has the content f to be transferred to the destination node d.

The binary integer program (4) to be solved for the RNNA caching policy contains a total of $(V + 2V^2 + V^3)F$ binary variables, $(3V + V^2)F$ equality constraints, and $\Big((1 + V + 2V^2 + V^3)F + V^2 + V\Big)$ inequality constraints. If each content has approximately equal size, then the constraint matrix in (4) is a network matrix. As such, (4) is a unimodular linear program and can be solved with polynomial time complexity using interior-point numerical methods [31]. Although YouTube content is not of equal size, it can be broken into equal sized blocks. The reason is that typically YouTube content is of duration between 30 seconds to 5 minutes, with a user attention span of approximately 90 seconds [32]. Therefore, if each YouTube video is broken into 30 second intervals, then the decision of where to cache these video blocks throughout the network can be solved in polynomial time using the optimization problem (4).

Though the RNNA caching policy (4) uses the LTE network parameters, popularity of content, and content transfer protocols, it does not account for the uncertainty associated with estimating the request count y_{vf} for content f at node v. Therefore, we can not control how to account for the risk associated with any given caching decision. Here the risk can be viewed as a measure of the worst case content retrieval delay that results from a given caching decision.

4.3 Conditional Value-at-Risk (CVaR) and Content Retrieval Delay Minimization

The two risk-averse caching policies RA and RANA both use the coherent CVaR risk measure to account for the uncertainty associated with predicting the content requests. Here, we precisely define the CVaR risk measure and how it accounts for the uncertainty associated with predicting the content requests.

Let D be a random variable that denotes the total content retrieval delay in a time-interval $[t, t + \Delta T]$. Realizations of the total content retrieval delay are defined by d (2). The ability to compare random outcomes of D based on the confidence $\alpha \in [0,1]$ is crucial for accounting for the risk associated with the uncertainty of estimating y_{vf} when performing caching decisions.

How can we estimate the total content retrieval delay D for a given confidence level $\alpha \in [0,1]$ (where $\alpha = 1$ is completely risk-averse, and $\alpha = 0$ is risk-neutral)? One possibility is to use the Value-at-Risk (VaR) risk measure

$$\mathrm{VaR}_\alpha(D) = \min\{d \in \mathbb{R} : F_D(d) \geq \alpha\} \tag{6}$$

where $F_D(d)$ is the cumulative distribution function of D. Classically, $\mathrm{VaR}_\alpha(D)$ was a popular method to estimate risk, however, it has several limitations [17], [18]. First, $\mathrm{VaR}_\alpha(D)$ is difficult to optimize as it is non-convex and is a non-coherent measure of risk as it fails the sub-additive condition. Second, $\mathrm{VaR}_\alpha(D)$ does not account for the properties of the distribution $F_D(d)$ beyond the threshold $\mathrm{VaR}_\alpha(D)$.

To account for the uncertainty of estimating y_{vf} for computing D, we use the CVaR measure [17], [18] which is a coherent risk measure. That is, CVaR satisfies the following properties: it is positive homogeneous, sub-additive, monotonic, translation invariant with respect to first order stochastic dominance, and monotonic with respect to second order stochastic dominance. CVaR is the expected total

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2018.2855160, IEEE Transactions on Cloud Computing

delay given that we are in the $\alpha \in [0,1]$ confidence interval of the cumulative distribution $F_D(d)$. Formally, CVaR is given by

$$\begin{aligned}\mathrm{CVaR}_\alpha(D) &= E_D[D|D \geq \mathrm{VaR}_\alpha(D)] \\ &= \frac{1}{1-\alpha}\int_\alpha^1 \mathrm{VaR}_\beta(D)\, d\beta. \end{aligned} \tag{7}$$

where E_D denotes the expectation with respect to the random variable D that represents the content retrieval delay. As seen from (7), $\mathrm{CVaR}_\alpha(D)$ can be interpreted as the conditional expectation of D where the expectation is evaluated on the α-confidence portion of the tail distribution $F_D(d)$. If $\alpha = 0$, then $\mathrm{CVaR}_\alpha(D) = E[D]$ is the expected value of D, and if $\alpha \to 1$, then $\mathrm{CVaR}_\alpha(D) = \max\{D\}$ gives the maximum value of D.

Given the CVaR risk measure (7), to minimize the total content retrieval delay in a time-interval $[t, t+\Delta T]$ for a confidence level α requires solving the following optimization problem

$$C^* \in \arg\min_{z\in\mathcal{Z}} \{\mathrm{CVaR}_\alpha(D)\}. \tag{8}$$

where z are the decision variables that affect the total content retrieval delay D, and $\mathcal{Z}$ are the associated constraints on the decision variables. C^* in (8) is the associated caching decision that minimizes the total content retrieval delay with a confidence of α. If $\alpha = 0$ in (8), then (8) is equivalent to the risk-neutral caching policies discussed in Sec.4.1 and Sec.4.2. Here, we are interested in risk-averse caching policies for evaluating (8) where $\alpha \in (0,1]$.

The evaluation of (8) for general $\alpha \in (0,1]$ is non-trivial as it requires an analytical expression for the cumulative distribution function $F_D(d)$ which is unknown. Recall that the random variable D representing the total content retrieval delay in the time-interval $[t, t+\Delta T]$ which depends on the LTE network parameters, popularity of content, content transfer protocols, and the uncertainty associated with estimating the request count y_{vf} for content f at node v. Using Theorem 2 in [17], the optimization problem (8) can be represented by

$$C^* \in \arg\min_{z\in\mathcal{Z}, c\in\mathbb{R}} \{c + \frac{1}{1-\alpha} E_D[\max\{0, D-c\}]\} \tag{9}$$

where the expectation is taken with respect to the random variable D. Though the distribution $F_D(d)$ is unknown, if the distribution of the content requests $F_{Y_{vf}}(y_{vf})$ at node $v \in \mathcal{V}$ is known, then the objective function (9) can be approximated using Monte-Carlo integration techniques. That is, the expectation $E_D[\cdot]$ in (9) is estimated using K independent and identically distributed samples of the content requests y_{vf} generated from the distribution $F_{Y_{vf}}(y_{vf})$. Additionally, the non-smooth operator $\max\{\cdot\}$ in the objective function (9) can be removed by introduced auxiliary parameters ξ_k. The approximate solution to (9) can be computed by solving:

$$\begin{aligned} C^* \in \arg\min_{z\in\mathcal{Z}, c\in\mathbb{R}} &\{c + \frac{1}{K(1-\alpha)}\sum_{\xi=1}^{K}\xi_k\} \\ \text{s.t.}\quad & \xi_k \geq D(z, \hat{y}_{vfk}) - c \\ & \hat{y}_{vfk} \sim F_{Y_{vf}}(y_{vf}) \text{ for } v \in \mathcal{V}, f \in \mathcal{F} \\ & \xi_k \geq 0, \quad \text{for } k \in \{1,\ldots,K\}. \end{aligned} \tag{10}$$

where $\hat{y}_{vfk}$ represents the generated sample of the content requests for content f at node v for sample k.

The optimization problem (10) provides the basis for constructing the risk-averse static caching policies in this paper.

4.4 Risk-Averse (RA) Static Caching Policy

Here we construct a risk-averse (RA) static caching policy that accounts for the uncertainty associated with predicting the content requests y_{vf} for content f at node v. The caching method RA is a generalization of the caching method RN in Sec.4.1 that does not account for the uncertainty associated with estimating the content requests.

Given the conditional density function $F_{Y_{vf}}(y_{vf})$ of content requests, the CVaR optimization problem (10), and using the constraints in the popularity based caching method (3), the RA caching policy is defined by

$$\begin{aligned} c_v^* \in \arg\min_{c_{vf}, c} &\left\{ c + \frac{1}{K(1-\alpha)}\sum_{\xi=1}^{K}\xi_k \right\} \\ \text{s.t.}\quad & c_{vf} \in [0,1], c \in \mathbb{R}, \\ & \xi_k \geq \sum_{f=1}^{F}\sum_{v=1}^{V} \hat{y}_{vfk}(1-c_{vf}) - c, \qquad (11a) \\ & \hat{y}_{vfk} \sim F_{Y_{vf}}(y_{vf}), \qquad \eta_k \geq 0, \\ & \sum_{f=1}^{F} s_f c_{vf} \leq S_v, \qquad (11b) \\ & v \in \mathcal{V}, f \in \mathcal{F}, k \in \{1,\ldots,K\}, \end{aligned}$$

where $C^* \in [0,1]^{V\times F}$. The parameter K in (11) is the total number of the content requests generated from the distributions $F_{Y_{vf}}(y_{vf})$ for $v \in \mathcal{V}$ and $f \in \mathcal{F}$. Additionally, the parameter c in (11) represents the estimated Value-at-Risk (VaR) of the total content delay for a confidence α, and $\alpha \in (0,1]$ is the confidence level.

The RA caching policy (11) is a mixed integer linear program that contains $2K+V$ inequality constraints, FV binary variables, $K+1$ real variables, and requires the generation of VFK samples from the distributions $F_{Y_{vf}}(y_{vf})$. The complexity of (11) is NP-hard, however several numerical methods exist which can be used to evaluate (11) including: Branch-and-bound, cutting planes, branch-and-cut, and branch-and-price [33], [34]. The selection of the number of samples K to use is still an open research problem. However, we found that a reasonable performance is achieved using $K = 10,000$ samples.

Though RA caching policy (11) accounts for the uncertainty associated with predicting the number of requests, it neglects the LTE network parameters (network bandwidth, load at the nodes, request-queue-time, content cached at other nodes), the network-layer protocol, and the link-layer protocol of the LTE network.

4.5 Risk-Averse and Network-Aware (RANA) Caching Policy

Here we construct RANA caching policy that accounts for the uncertainty associated with predicting the number of

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2018.2855160, IEEE Transactions on Cloud Computing

requests, LTE network parameters (network bandwidth, load at the nodes, request-queue-time, content cached at other nodes), the network-layer protocol, and the link-layer protocol of LTE network.

Given the conditional density function $F_{Y_{vf}}(y_{vf})$ of content requests, the CVaR optimization problem (10), and using the constraints in (4), the RANA caching policy is given by

$$
\begin{aligned}
C^* \in \arg\min_{C,k,\delta,r,c} & \left\{ c + \frac{1}{K(1-\alpha)} \sum_{\xi=1}^{K} \xi_k \right\} \\
\text{s.t.} \quad & c_{sf} \in [0,1], \quad k_{sdf} \in [0,1], \quad \delta_{ijdf} \in [0,1], \\
& r_{sdf} \in [0,1], \quad T_{ij} \in \mathbb{Z}_+, \quad c \in \mathbb{R}, \\
& \xi_k \geq \sum_{f=1}^{F} \sum_{d \in \mathcal{V}_d} \sum_{i,j \in \mathcal{V}} \hat{y}_{df} A_{ijf} \delta_{ijdf} - c, \\
& \sum_{i \in \mathcal{V}} \delta_{sidf} - \delta_{isdf} = k_{sdf}, \quad \sum_{i \in \mathcal{V}} \delta_{didf} - \delta_{iddf} = -1, \\
& \mathbf{1}\{b_{ij} = 0\} + \delta_{ijdf} \leq 1 \\
& \sum_{f=1}^{F} s_f c_{sf} \leq S_s, \quad \sum_{s=1}^{V} c_{sf} \geq 1 \\
& \sum_{s=1}^{V} k_{sdf} = 1, \quad \sum_{s=1}^{V} r_{sdf} = 1, \\
& \sum_{f=1}^{F} \sum_{d \in \mathcal{V}_d} \delta_{ijdf} \leq T_{ij} \qquad \text{(12a)} \\
& r_{sdf} \leq k_{sd}^f, \quad r_{sdf} \leq c_{sf}, \quad r_{sdf} \geq k_{sdf} + c_{sf} - 1, \\
& \forall s \in \mathcal{V}, \quad \forall d \in \mathcal{V}_d, \quad \forall f \in \mathcal{F}, \quad k \in \{1, \ldots, K\}.
\end{aligned}
$$

A description of each of the constraints in (12) is provided below (4). Note that the RANA caching policy (12) is equivalent to the RNNA caching policy (4) if we do not account for the uncertainty associated with predicting the number of requests for content $f \in \mathcal{F}$ at node $v \in \mathcal{V}$–that is, we set $\alpha = 0$ in (12).

The mixed integer linear program (12) contains a total of $(V + 2V^2 + V^3)F$ binary variables, $(2K + 1)$ real variables, $(3V+V^2)F$ equality constraints, and $\Big((1+V+2V^2+V^3)F+ V^2 + V + 2K + 1\Big)$ inequality constraints. As discussed in Sec.4.4, several numerical methods can be used to evaluate C^* in (12) for typical femtocell networks that contain up to $V = 10$ wireless nodes and $F = 1000$ content.

Summary: In this section we constructed four static caching policies, namely, RN in (3), RNNA in (4), RA in (11), and RANA in (12). The RNNA caching policy accounts for content requests, cache size, bandwidth, load, and content routing, only requires the solution to a unimodular linear program. The unique feature of the risk-averse caching policies RA and RANA compared to RN and RNNA is that they use a coherent risk measure to account for the uncertainty associated with predicting the content requests. The network operator can use confidence level $\alpha \in (0, 1]$ in the RA and RANA methods to select the level of risk when performing a caching decision. For example, setting $\alpha = 1$ leads to minimizing the maximum possible content retrieval delay in the network for RA and RANA methods. To evaluate the caching decision using RA or RANA requires a conformal prediction of the content requests–that is, an estimate of the cumulative distribution function of the content requests. In Sec.5 we provide a conformal prediction algorithm for constructing the cumulative distribution function of the requests.

5 Content Request Cumulative Distribution Function Forecasting

The risk-neutral and risk-averse static caching policies constructed in Sec.4 require a point estimate $\hat{y}_{vf}$ or cumulative distribution function estimate $F_{Y_{vf}}(y_{vf})$ of the request count y_{vf} for content $f \in \mathcal{F}$ at node $v \in \mathcal{V}$. In this section we construct a conformal prediction algorithm to estimate $F_{Y_{vf}}(y_{vf})$ given the dataset $\mathcal{D}(T)$ in (1). The key idea of the conformal prediction algorithm is to use discriminant analysis to perform coarse-grained prediction of the content requests. Then a feed-forward neural network is used to perform fine-grained request estimation. For both the coarse-grained and fine-grained request estimates, the prediction interval of each is denoted by $P(g|x_f)$ and $F_{Y_f}(y_f|g, x_f)$ respectively where $g \in \mathcal{G}$ represents the group association, and Y_f is the random variable representing the number of requests for content $f \in \mathcal{F}$. Using the total-law of probability the cumulative distribution function of the content requests is

$$
F_{Y_f}(y_f|x_f) = \sum_{g=1}^{G} F_{Y_f}(y_f|g, x_f) P(g|x_f). \qquad (13)
$$

for content $f \in \mathcal{F}$. Below we present the coarse-grained and fine-grained request prediction methods, and the density forecasting algorithm to evaluate (13).

5.1 Content Group Association Classifier

Given the dataset $\mathcal{D}(T)$ in (1), the goal is to construct a classifier that can assign content $f \in \mathcal{F}$ to a particular group $g \in \mathcal{G}$ and provide a confidence estimate of the group association. That is, we desire a classifier which learns the conditional probability mass function $P(g|x)$ of group association.

The elements of the content features x_f are commonly constrained to intervals on the real line. For example, for YouTube videos, the number of subscribers to the user that uploaded the video must be a positive number and the minimum length of a video is 1 second (15 seconds if a ads are present). Let us assume that the feature vector x is a random variable which has a conditional probability density function given by a doubly truncated multivariate normal distribution

$$
\begin{aligned}
p(x|g) &= \mathcal{N}(\mu_g, \mu_g^-, \mu_g^+, \Sigma_g) \qquad (14) \\
&\propto \exp\left(-\frac{1}{2}(x - \mu_g)' \Sigma_g^{-1} (x - \mu_g) \right) \mathbf{1}\{\mu_g^- \leq x \leq \mu_g^+\}.
\end{aligned}
$$

In (14), μ_g is the mean vector of features in group $g \in \mathcal{G}$, μ_g^- is the minimum value of the content features in group g, μ_g^+ is the maximum value of the content features in group g, and Σ_g is the covariance matrix of the features in group g. If the mean μ_g, lower and upper limits (μ_g^- and μ_g^+) on the feature

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2018.2855160, IEEE Transactions on Cloud Computing

vector, and covariance matrix Σ_g are known for each group $g \in \mathcal{G}$, and the prior probability of group association is $P(g)$, then the probability of content $f \in \mathcal{F}$ being associated with group $g \in \mathcal{G}$ is

$$P(g|x_f) = \frac{P(x_f|g)P(g)}{\sum_{r=1}^{G} P(x_f|r)P(r)}. \tag{15}$$

The prior probability $P(g)$ of group association can either be set to an uninformative prior (i.e. $P(g) = 1/G$), or to the population average with $P(g) = n_g/F$ where n_g is the number of content in $\mathcal{F}$ that are associated with group $g \in \mathcal{G}$.

Given the dataset $\mathcal{D}(T)$ in (1), how can the parameters $\mu_g, \mu_g^+, \mu_g^-, \Sigma_g$ in (14) be estimated? If $\mu_g^- = -\infty$ and $\mu_g^+ = +\infty$ (the feature vector x is unconstrained), then classical discriminant analysis methods can be used to estimate μ_g and Σ_g. These include Linear Discriminant Analysis, Factor-Based Linear Discriminant Analysis, Maximum Uncertainty Linear Discriminant Analysis, and Regularized Discriminant Analysis [35]. Typically, the content features x are contained on an interval of the real line such that $-\infty < \mu_g^-$ and $\mu_g^+ < +\infty$. Given μ_g^- and μ_g^+, the estimate of the mean μ_g and covariance Σ_g can be computed using Gibbs sampling, maximum likelihood estimation, and generalized method of moments [36]. Here use the maximum likelihood estimation method to estimate μ_g and Σ_g using the Limited-memory Broyden-Fletcher-Goldfarb-Shanno algorithm with box constraints to account for the limits of the feature vector x.

5.2 Risk-Averse Feed-foward Neural Network for Predicting Content Requests

In this section we construct the risk sensitive extreme learning machine (RSELM) that combines the benefits of the extreme learning machine with a risk-aware training method to predict content requests. The RSELM is comprised of a single-layer feed-foward neural network trained using a stochastic optimization algorithm that dynamically adjusts the number of neurons and weights to minimize the risk of over fitting with a confidence level $\alpha \in (0, 1)$.

5.2.1 Classic Extreme Learning Machine

Given the dataset $\mathcal{D}(T)$ in (1), the goal is to construct a method to estimate the functional relationship between the content features x_f and the associated request count $y_f(t)$. A single-layer feed-foward neural network can be used to relate the content features x_f to the requests $y_f(t)$. Denoting $\hat{y}_f(t)$ as the estimated request count given x_f, the feed-forward neural network is given by

$$y_f(t) = \sum_{i=1}^{L} \beta_i h_i(x_f; \theta_i) = \boldsymbol{\beta}' \boldsymbol{h}(x_f) \tag{16}$$

with $\boldsymbol{\beta} = [\beta_1, \beta_2, \ldots, \beta_L]'$ are the weights of each neuron, and $\boldsymbol{h}(x_f) = [h_1(x_f; \theta_1), \ldots, h_L(x_f; \theta_L)]'$ the associated transfer function of each neuron. Popular transfer functions include the sigmoid, hyperbolic tangent, and Gaussian however any non-linear piecewise continuous function can be utilized. The neuron weights β_i and θ_i in (16) are computed from the solution to

$$\boldsymbol{\theta}^*, \boldsymbol{\beta}^* \in \arg\min \left\{ \sum_{f=1}^{F} (y_f(t) - \hat{y}_f(t))^2 \right\}. \tag{17}$$

For general transfer functions $\boldsymbol{h}(x_f)$, (17) is a non-convex and non-linear optimization problem that is commonly solved using stochastic gradient decent, ant-colony optimization, and simulated anealling methods [37], [38], [39]. A draw-back with these numerical methods for solving (17) is that they require a large number of computations to converge to the global optimal solution of (17). However, using random matrix theory, it has been shown that the parameter values at the local minima of the objective function (17) yield similar mean-square error compared to the global optimal solution of (17) [40]. Therefore, the mean-square error of the feed-forward neural network is not sensitive to the set of local minima $\{\boldsymbol{\theta}^*, \boldsymbol{\beta}^*\}$ is used.

Could selecting the neuron weights $\boldsymbol{\theta}$ randomly, and then fitting the weights $\boldsymbol{\beta}$ via least-squares minimization provide a reasonable approximation to the solution of (17)? Since the mean-squared error of the feed-forward neural network is not sensitive to which local minima of (17) are used, it is reasonable to postulate that randomly selecting $\boldsymbol{\theta}$ and then fitting $\boldsymbol{\beta}$ will produce a reasonable solution. This is the main idea behind the extreme learning machine (ELM) proposed in [41]. For fixed number of neurons L, the ELM selects the parameters $\boldsymbol{\theta}$ and $\boldsymbol{\beta}$ in two steps. First, the hidden layer parameters $\boldsymbol{\theta}$ are randomly initialized. Any continuous probability distribution can be used to initialize the parameters $\boldsymbol{\theta}$. Second, $\boldsymbol{\beta}$ is selected to minimize the mean-square error between the model output and the measured output from the dataset $\mathcal{D}(T)$ in (1). Formally,

$$\begin{aligned} \boldsymbol{\theta}^* &\sim \mathcal{N}(0, 1) \\ \boldsymbol{\beta}^* &= H(X; \boldsymbol{\theta}^*)^+ Y \end{aligned} \tag{18}$$

where $\mathcal{N}(0, 1)$ is the multivariate normal distribution with unit variance, $H(X; \boldsymbol{\theta}^*)$ is the hidden-layer output matrix with entries $H_{if}(X; \boldsymbol{\theta}) = h_i(x_f; \theta_i)$ for $i = 1, \ldots, L$ and $f \in \mathcal{F}$, H^+ denotes the Moore-Penrose generalized inverse of H, and $Y = [y_1, \ldots, y_F]'$. Several efficient methods can be used to compute $\boldsymbol{\beta}^*$, for example Gaussian elimination. The major benefit of using the ELM, (16) and (18), is that the training only requires the random generation of the parameters $\boldsymbol{\theta}^*$, and $\boldsymbol{\beta}^*$ is computed as the solution of a set of linear equations.

The ELM satisfies the universal approximation condition [42], can be implemented in parallel [43] which is important for online implementation [44]. In addition ELM can be trained sequentially for large datasets or as new training data becomes available [45], [46], and can be efficiently implemented on field-programmable gate array devices as well as complex programmable logic devices [47]. A limitation with the ELM is that it can not be used to select the number of neurons L while minimizing the risk of overfitting the dataset $\mathcal{D}(T)$ in (1).

5.2.2 Regularized Extreme Learning Machine

Here we construct the regularized extreme learning machine (RELM) which is a generalization of the ELM that minimizes

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2018.2855160, IEEE Transactions on Cloud Computing

the risk of overfitting with a confidence level $\alpha \in (0, 1]$. For example, given the dataset $\mathcal{D}(T)$ in (1) and setting $\alpha = 1$, the RELM selects the parameters $\boldsymbol{\theta}, \boldsymbol{\beta}, L$ in (16) to minimize the mean-square generalization error between the actual and predicted content requests. The RELM is equivalent to the ELM if we do not account for risk–that is, we set $\alpha = 0$.

The RELM is trained by solving the following optimization problem (we discuss the motivation below):

$$L^* \in \underset{L\in\mathbb{Z}_+, c\in\mathbb{R}}{\arg\min} \{c + \frac{1}{K(1-\alpha)} \sum_{k=1}^{K} z_k\}$$

$$\text{s.t.} \quad z_k \geq \frac{1}{(1-\gamma)F} ||\eta(L)H(\bar{X}_k; \boldsymbol{\theta}_k)\boldsymbol{\beta}_k - \bar{Y}_k||_2^2 - c \tag{19a}$$

$$X_k, \bar{X}_k, \bar{Y}_k, Y_k \sim \text{FY}(\mathcal{D}(T), \gamma) \tag{19b}$$

$$\boldsymbol{\theta}_k \sim \mathcal{N}(0, 1), \tag{19c}$$

$$\boldsymbol{\beta}_k = [\eta(L)H(X_k; \boldsymbol{\theta}_k)]^+ Y_k \tag{19d}$$

$$z_k \geq 0, \quad L \leq L_{\max}, \quad \text{for } k \in \{1, \ldots, K\}. \tag{19e}$$

In (19), $\mathcal{N}(0, 1)$ is the multivariate normal distribution, $\text{FY}(\mathcal{D}, \gamma)$ is the Fisher-Yates random permutation which is used to partition the data $\mathcal{D}(T)$ into complementary subsets of training data (X_k, Y_k) and validation data $(\bar{X}_k, \bar{Y}_k)$ where $\gamma \in (0, 1)$ denotes the percentage of data used for training, and $\eta(L)$ is a diagonal matrix with elements

$$\eta_{ii}(L) = \begin{cases} 1 \text{ if } i \leq L \\ 0 \text{ otherwise.} \end{cases} \tag{20}$$

The parameter $L_{\max}$ is the maximum number of neurons in the RELM, and K is the number of samples used to estimate the cumulative distribution function of the mean-square error of the feed-forward neural network.

The objective function in (19) represents the CVaR of the mean-square generalization error of the ELMs with L neurons for a confidence level α. The mean-square generalization error of each ELM is evaluated using

$$\frac{1}{(1-\gamma)F} ||\eta(L)H(\bar{X}_k; \boldsymbol{\theta}_k)\boldsymbol{\beta}_k - \bar{Y}_k||_2^2$$

defined in constraint (19a). The constraint (19b) is used to generate the training (X_k, Y_k) and testing $(\bar{X}_k, \bar{Y}_k)$ data for each of the $k \in \{1, \ldots, K\}$ ELMs. The constraint (19c) is used to generate the neuron transfer function weights $\boldsymbol{\theta}_k$. Given the transfer function weights $\boldsymbol{\theta}_k$ and the training data (X_k, Y_k), constraint (19d) is used to construct the neuron weights $\boldsymbol{\beta}_k$ of each ELM. The final constraint (19e) defines the maximum number of possible neurons $L_{\max}$ and the number of ELMs K generated to evaluate the CVaR risk measure. The selection of $L_{\max}$ and K are important to restrict the computational resources used to train and evaluate the RELM. The final result of the RELM (19) is the optimal number of neurons L^* of use for predicting the content requests while minimizing the risk of overfitting the training data with a confidence level α. Given L^*, the ELM weights $\boldsymbol{\theta}, \boldsymbol{\beta}$ can be constructed using equation (18).

Discussion of (19): Solving the mixed integer nonlinear program (19) is equivalent to optimally selecting the number of neurons L to minimize the risk of over fitting the dataset $\mathcal{D}(T)$. The training method in (19) dynamically adjusts the number of neurons L based on the available training data in $\mathcal{D}(T)$. Typically, as the number of observations in $\mathcal{D}(T)$ increases, the number of neurons in the RSELM will also increase. The solution to (19) can be computed in polynomial time by solving (19) for $L = 1, \ldots, L_{\max}$ independently, and then selecting the solution that minimizes the objective function in (19).

5.3 Conformal Prediction Algorithm for Content Requests

In this section we construct the conformal prediction algorithm to estimate the cumulative distribution function $F_{Y_f}(y_f|x_f)$ of the number of requests for content $f \in \mathcal{F}$ given the content features x_f.

A schematic of the conformal prediction algorithm is provided in Fig.3. The algorithm is comprised of an offline training stage, and an online stage to evaluate the requests for new content. In the offline stage the dataset $\mathcal{D}(T)$ in (1) is used to train the group association classifier defined in Sec.5.1, and the RELM defined in Sec.5.2. Then, using the dataset

$$\hat{D}(T) = \{\{x_f, g_f(t), y_f(t), \hat{y}_f(t)\} : v \in \mathcal{V}, f \in \mathcal{F}, t \in [0, T]\}, \tag{21}$$

the set of prediction errors $\{\varepsilon_i(g)\}$, where $\varepsilon_i(g) = \hat{y}_i(g, x_f) - y_f$ for each content $i \in \mathcal{F}$ associated with group $g \in \mathcal{G}$ is constructed. The trained parameters from the offline stage are then used in the online stage to estimate the request count of the content. In the online stage, when a new content with features x_o is received, the group association classifier is used to compute the group association probabilities $P(g|x_o)$. Then the RELM is used to estimate the number of requests $\hat{y}_f(x_o, g)$ for the content f. The group association probabilities and predicted number of requests from the RELM are then sent to the conformal prediction block which outputs the conformal prediction $F_Y(y_o|x_o)$.

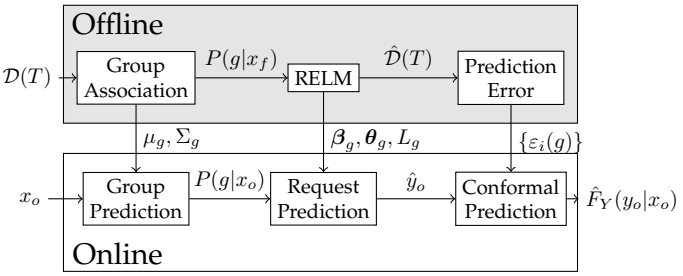


Fig. 3. Schematic of the conformal prediction algorithm. $\mathcal{D}(T)$ in (1) is the training dataset, $P(g|x_f)$ is the probability of group association, $\hat{D}(T)$ is the training dataset with the predicted content requests from the RELM, $\{\varepsilon_i(g)\}$ are the errors between the predicted and actual requests for content in group $g \in \mathcal{G}$, x_o is a new content feature, and $\hat{F}_Y(y_o|x_o)$ is the empirical cdf function of the content given the content features x_o.

Insight into the design of the conformal prediction algorithm in Fig. 3 is gained by considering the content requests y_f for content f as a random variable. We assume that the random variable y_f satisfies

$$y_f = \hat{y}_f(g, x_f) + \varepsilon_f(g) \tag{22}$$

where $\hat{y}_f(g, x_f)$ is the estimated number of requests from the RELM trained using data from group $g \in \mathcal{G}$, and $\varepsilon_f(g)$ is the random variable that accounts for the error

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2018.2855160, IEEE Transactions on Cloud Computing

in the predicted and actual number of requests. The error $\varepsilon_f(g)$ accounts for the errors associated with the parametric structure of the RELM, the parameters of the RELM, and the errors that may be contained in the feature vector x_f. The random variables $\varepsilon_f(g)$ are assumed independent and identically distributed. Then, given the dataset $\hat{\mathcal{D}}$, the empirical cdf of the random variable y_f is

$$\hat{F}_Y(y_f|g,x_f) = \frac{1}{n_g}\sum_{i=1}^{n_g} \mathbf{1}\left\{\varepsilon_i(g) \leq y_f - \hat{y}_f(g,x_f)\right\} \quad (23)$$

where n_g is the total number of contents in group $g \in \mathcal{G}$ and $\mathbf{1}\{\cdot\}$ is the indicator function.

Substituting (23) into (24) gives the conformal prediction of the content $f \in \mathcal{F}$. Formally, the empirical conditional distribution function of the number of requests for content $f \in \mathcal{F}$ is

$$\hat{F}_Y(y_f|x_f) = \sum_{g=1}^{G} \hat{F}_Y(y_f|g,x_f)P(g|x_f) \quad (24)$$

where $P(g|x_f)$ is the conditional probability of content f belonging to group $g \in \mathcal{G}$ from the discriminant analysis classifier.

Summary: In this section we constructed a conformal prediction algorithm, illustrated in Fig. 3, for content requests. The conformal prediction algorithm is comprised of an offline learning stage in which content request and features are used to select the doubly-truncated multivariate normal distribution parameters for group association estimation, and train the RELM for constructing point estimates of the content requests. For new content, the output of the trained group classifier and RELM are used to construct the conformal prediction of the content requests. This conformal prediction is used with the risk-averse caching methods in Sec.4.4 and 4.5 to optimally cache content in the LTE network.

6 Numerical Evaluation of the Conformal Prediction Algorithm and Coherent Risk Minimization Caching Policies for YouTube Content

In this section we evaluate the performance of the conformal prediction algorithm and four static caching policies (RN, RNNA, RA, RANA) presented in Sec.4 using real-world YouTube datasets. The results show that a 6% reduction in the average delay can be achieved if the uncertainty of the content requests is accounted for, and a 60% reduction in average delay is achieved if both the uncertainty and femtocell routing protocol are accounted for compared to the risk-neutral caching policy that neglects the routing protocol. These results illustrate the importance of both accounting for the risk associated with estimating content requests and accounting for the routing protocol used to transfer content throughout the network.

6.1 LTE Network Parameters and YouTube Dataset

To evaluate the performance of the static caching policies (RN, RNNA, RA, RANA), and the conformal prediction method illustrated in Fig.3, we construct an LTE heterogeneous network and generate user content requests based on real-world YouTube datasets.

LTE Network Parameters: The LTE network topology used for evaluation is illustrated in Fig.4. The LTE network is composed of a core network that is connected to base station nodes, femtocell gateway nodes, and a server which contains all the content $\mathcal{F}$ that can be requested by users. The content server and the core network communicate via the wide area network. The latency in the wide area network is typically in the range of 10 ms to 100 ms depending on the distance and the number of hops between the server and the core network [48], [49]. The core network, base station, and gateway nodes communicate with one another via an intra-network communication link with link capacity values in the range of 500 Mbps to 1 Gbps. Finally, the femtocell access points are connected to the gateway nodes via heterogeneous backhaul links with link capacity values in the range of 100 Mbps to 500 Mbps.

In the LTE network illustrated in Fig.4, user requests are only received by the femtocell access points and the base station nodes. Each requested video content has a size of $s_f = 200$ MB. The femtocell access points, base station, and gateway have a cache size of 500 GB (approximately 10% of the entire content library), and the core network node has a cache size of 1 TB (approximately 20% of the entire content library). For each node, 90% of the cache storage is used for static caching, and the remaining is used for dynamic caching. The dynamic cache segment of each node is controlled using the Least-Recently-Used (LRU) replacement method discussed in [50]. Given the cache size, link capacity between the nodes, processing delay, propagation delay, number of hops between nodes, packet size, and the topology in the LTE network in Fig.4, the edge weight parameter A_{ijf} (5) between node i and node j is evaluated using the ndnSIM 2.0 NS-3 base simulator [51]. ndnSIM allows a video content f to be addressable and routable inside the network [52]. For the network-layer protocol, ndnSIM's NDN stack is used while *point-to-point* communication is considered as the link layer protocol. In ndnSIM, the femtocell access points and base station are defined as the *Consumer* node. We implement a new consumer application method for the *Consumer* node to generate content requests according to the YouTube datasets. All the nodes having cache storage are considered as the *Producer* node which can satisfy content requests generated by the *Consumer* nodes. The ndnSIM *Best Route* is used to forward content requests to neighbouring nodes until the content is retrieved–this is equivalent to the shortest-path routing algorithm. Finally, the ndnSIM *Application-level trace helper* is used to compute the edge weight A_{ijf} (5) between node i and node j for content f. We set $T_{ij} = 16$ in equation (4c) and in equation (12a).

YouTube Dataset: The real-world YouTube dataset was collected using the YouTube API between the years 2013 to 2015 and consists of $25,000$ YouTube videos. The dataset is comprised of videos from 17 YouTube categories including "Gaming" (44% of videos), "Entertainment" (40% of videos), "Music" (5% of videos), and "Education" (2% of videos). Note that gaming and entertainment are among the most popular video categories on YouTube. The video requests range from 10^2 to above 10^7. Therefore, to prevent the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2018.2855160, IEEE Transactions on Cloud Computing

ELM algorithm from biasing it's prediction to only the videos with the highest requests, we apply a log transform to the requests such that the range is in 2 to 7. All the content features are scaled to satisfy $x(m) \in [0,1]$ for $m \in \{1,\ldots,M\}$. We use $11,747$ video content to construct the training dataset $\mathcal{D} = \{\{x_f, g_f, y_f\} : f \in \mathcal{F}\}$. The remaining videos are used to test the performance of the conformal prediction algorithm. In total there are $G = 3$ groups in the dataset, where group $g = 1$ is associated with videos with less than 100 requests, $g = 2$ with videos with requests in the range of 100 to $30,000$ requests, and $g = 3$ videos that have more then $30,000$ requests. For testing the performance of the discriminant analysis classifier, extreme learning machines, and coherent risk-minimization algorithms, we use the dataset $\hat{\mathcal{D}}$ which is comprised of $13,253$ videos. The number of requests for each video content is identical at each of the femtocell access points and base station.

As a preliminary step, the YouTube videos are clustered based on their associated category. The content requests of cluster $c \in \mathcal{C}$ at node v is:

$$y_c^v = \sum_{f \in F^c} y_f^v, \tag{25}$$

where $F^c \subseteq \mathcal{F}$. Given the content requests per cluster, the caching methods presented in this paper optimally cache each category of content in the network. We use the top 10 most popular YouTube categories to compute the performance metrics of the caching policies. To evaluate the delay of the risk-averse caching policies a method is required to generate user request that is consistent with the YouTube dataset $\bar{\mathcal{D}}$.

Content Request vector: We discretize time into a total of K_t time slots, where for each time-slot a content request is received. We construct content popularity distribution using dataset $\bar{\mathcal{D}}$ where P_f^v denotes the probability of content f being requested at node v and $P_f^v \sim y_f^v$. Let r^v denote the content request vector with elements $r^v(k) \in \mathcal{F}$ that define the content f being requested time k at node v. The content request vector satisfies the condition

$$\sum_{k=1}^{K_t} \mathbf{1}\{r^v(k) = f\} \sim P_f^v \quad \forall f \in \mathcal{F}. \tag{26}$$

To construct the content request vector we randomly generate a total of $K_t = 20,000$ content requests in r^v such that (26) is satisfied. The estimated delay for a given caching decision is computed by evaluating the total delay associated with the content requests from the request vector r^v at each of the v nodes.

6.2 Conformal Prediction Algorithm for YouTube Content

In this section the performance of the conformal prediction algorithm presented in Sec.5, and illustrated in Fig. 3, is evaluated using real-world YouTube datasets. This includes the performance of the discriminant analysis classifier, extreme learning machines, and the conformal prediction for video content requests.

From Fig. 3, the first step in the conformal prediction algorithm is the offline stage in which the parameters of the group association classifier (μ_g and Σ_g) are selected, the extreme learning machines are trained for each group $(\boldsymbol{\beta}_g, \boldsymbol{\theta}_g, L_g)$, and the parameters $\varepsilon_i(g)$ in (23) are estimated using the training dataset $\mathcal{D}$. The performance of the group classifier is evaluated using the true positive rate (TPR) and true negative rate (TNR), and the extreme learning machines using $\mathrm{CVaR}_\alpha(\varepsilon^2)$ for an $\alpha = 0.95$. The results are provided in Table 2 where the parameter n_g is the total number of content in each of the groups $g \in \mathcal{G}$. As seen, the group association classifier has a reasonable TPR and TNR for classifying the group association of each video. Using the RSELM in Sec.5.2, Table 2 illustrates that the selected number of neurons L_g for the ELM associated with group $g \in \mathcal{G}$ varies between the three groups, however the error $\mathrm{CVaR}_\alpha(\varepsilon^2)$ remains approximately equal. This illustrates that the RSELM can dynamically adjust the number of neurons necessary to effectively estimate the content requests $\hat{y}_f$ of new content while minimizing the risk of over-fitting with a confidence level $\alpha = 0.95$.

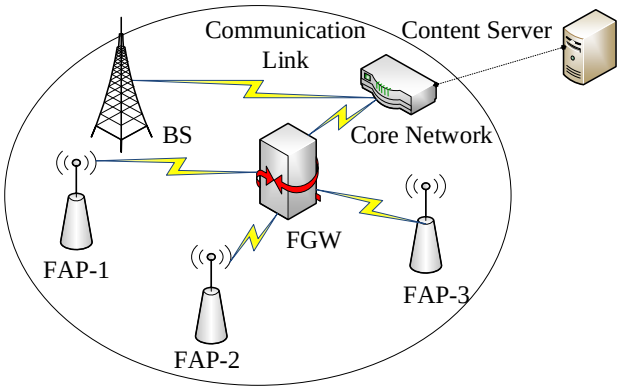


Fig. 4. Schematic of the network. There are three femtocell access points (FAPs) in the network. These FAPs are connected with the femtocell gateway (FGW) via heterogeneous communication links. FGW and base station (BS) are connected with the core network. Each FAPs, BS and SGW has a storage of 500 GB. Storage of the core network is assumed to be 1 TB and the core network is connected to the content server which can cache the entire library via wide area network (WAN).

TABLE 2
Group Classification and Neuron Number Selection

Group	n_g	TPR	TNR	$\mathrm{CVaR}_\alpha(\varepsilon^2)$	L_g
1	2405	0.80	0.96	0.3302	37
2	10314	0.94	0.79	0.3046	59
3	534	0.74	0.99	0.3133	24

To construct the conformal prediction (24) requires an estimate of the prediction errors $\varepsilon_i(g)$ in (23). Alternatively, we can construct an estimate of $\hat{F}_Y(y_f|g, x_f)$ using the empirical cumulative distribution function $\hat{F}_{E|g}(\varepsilon|g)$ of the random variables ε. Fig.5 illustrates the computed group empirical cumulative distribution function $\hat{F}_{E|g}(\varepsilon|g)$ of the error of the ELM associated with each group $g \in \mathcal{G}$. Using maximum likelihood estimation, we find that the empirical cdf $\hat{F}_{E|g}(\varepsilon|g)$ is approximately equal to the generalized ex-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2018.2855160, IEEE Transactions on Cloud Computing

treme value distribution typically used in risk management, finance, and economics. Given $\hat{F}_{E|g}(\varepsilon|g)$, the conditional distribution of content requests can be evaluated using

$$\hat{F}_Y(y_f|g,x_f) = \hat{F}_{E|g}(y_f - \hat{y}_f(x_f,g)|g,x_f) \qquad (27)$$

where $\hat{y}_f(x_f,g)$ is the estimated content requests from the ELM associated with group $g \in \mathcal{G}$ for content f.

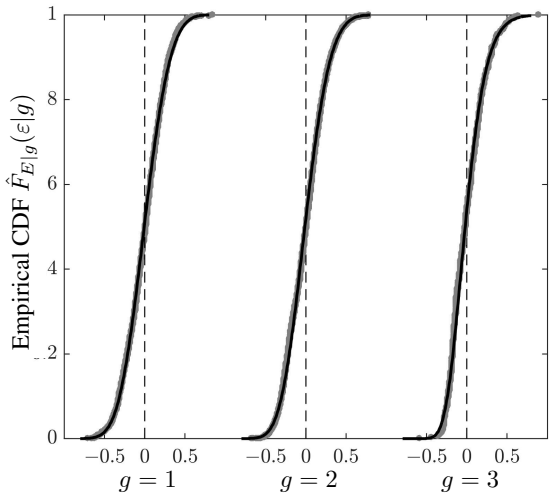
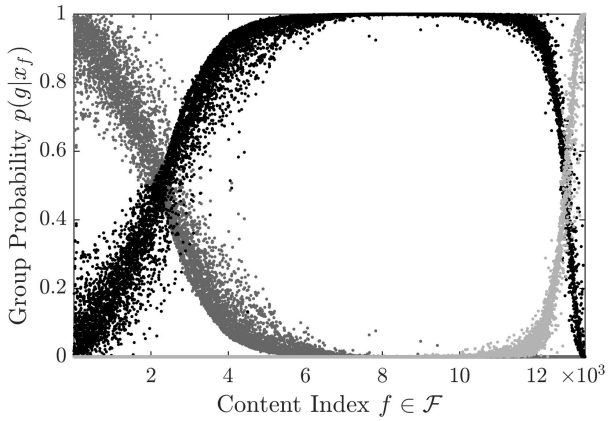


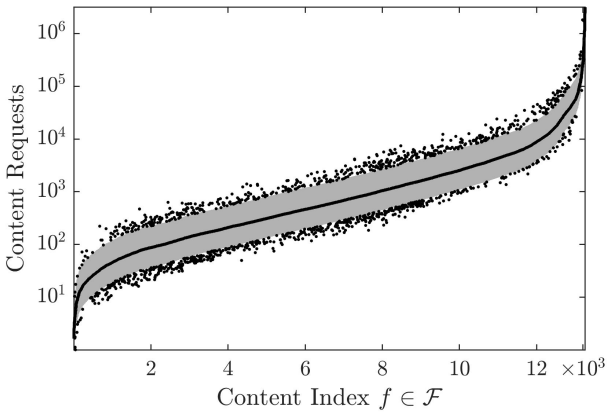
Fig. 5. Empirical cumulative distribution function of the error $\varepsilon(g)$ in (22) for the groups $g \in \mathcal{G}$. The gray dots indicate the empirical cumulative distribution function $\hat{F}_{E|g}(\varepsilon|g)$, and the black line indicates the fitted generalized extreme value distribution.

Given the parameters of the group association classifier, ELMs, and $\hat{F}_Y(y_f|g,x_f)$ from the offline stage of the conformal prediction algorithm, we now evaluate the performance of the online portion of the conformal prediction algorithm for new content. The group association probability $P(g|x_f)$ and results of the conformal prediction algorithm for new content are provided in Fig.6. The content index is ordered such that the least requested content is $f = 1$, and the most requested content is $f = 13,253$ based on the results of the conformal prediction. As Fig.6(a) illustrates, the group association probability $P(g|x_f)$ from the discriminant analysis classifier provides a reasonable accuracy for the probability of group association. From Fig.6(b), there are approximately $1,225$ contents (indicated in black) that have predicted number of requests from the ELMs that are outside the 90% confidence interval. Equivalently, approximately 9.2% of the content requests reside outside the 90% confidence interval computed from the conformal prediction algorithm. To determine if the cumulative distribution function $\hat{F}_Y(y_f|x_f)$ is consistent with observed content requests data, we use the quantile-quantile plot. The quantile-quantile plot in Fig. 7 is evaluated using the confidence interval of $\hat{F}_Y(y_f|x_f)$. As seen from Fig. 7, the empirical cumulative distribution $\hat{F}_Y(y_f|x_f)$ is in excellent agreement with the observed data. Therefore, $\hat{F}_Y(y_f|x_f)$ provides a reasonable approximation for the actual content request distribution $F_Y(y_f|x_f)$.

The above results indicate that the conformal prediction algorithm presented in Sec.5, and illustrated in Fig. 3, can be used to estimate the cumulative distribution function $F_Y(y_f|x_f)$ of YouTube content requests. The estimated cumulative distribution function $\hat{F}_Y(y_f|x_f)$ can then be used in the risk-averse caching policies (RA and RANA) to optimally cache content in the femtocell network. Additionally, the trained ELMs can be used for point predictions of the content requests for the risk-neutral caching policies (RN and RNNA).



(a) Group association probability $P(g|x_f)$. Gray dots indicate the probability of association with group $g = 1$, black dogts with $g = 2$, and light-gray dots with $g = 3$.



(b) Conformal prediction of the number of content requests. The expected number of requests is the solid black line, gray region is the 90% interval where the requests are expected to reside, and the black dots are the real number of requests in $\bar{\mathcal{D}}$.

Fig. 6. Performance of the conformal prediction algorithm that was schematically illustrated in Fig. 3, for YouTube content requests. Group $g = 1$ is associated with all videos that are predicted to have less than 100 requests, $g = 2$ with requests in the range of 100 to $30,000$, and group $g = 3$ with more then $30,000$ requests. The 90% confidence interval is evaluated using the empirical cumulative distribution function $\hat{F}_{Y_f}(y_f|x_f)$ of content requests defined in (24). As seen, the computed group association, point content requests, and conformal predictions are in excellent agreement with the YouTube datasets. The training and evaluation datasets are discussed in Sec.6.1.

6.3 Selection of the Confidence Level α for Maximum Content Retrieval Delay Guarantees

The risk-averse caching policies (RA and RANA) account for the uncertainty of the predicted content requests using the CVaR risk measure for a given confidence level α. From (7), for a confidence level α, CVaR is the expected content

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2018.2855160, IEEE Transactions on Cloud Computing

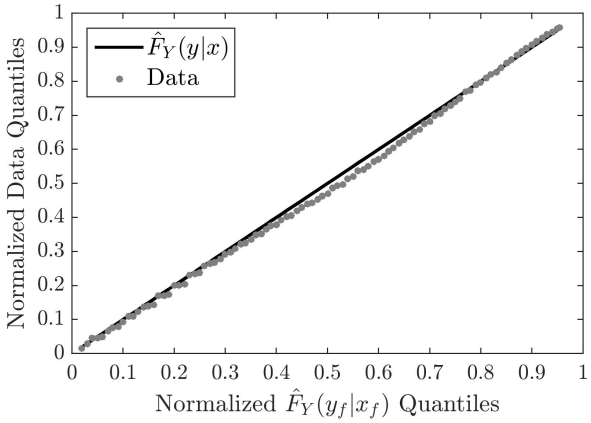


Fig. 7. Quantile-quantile plot for the empirical cumulative distribution function $\hat{F}_Y(y_f|x_f)$ and the YouTube content requests. The linear black line indicated perfect agreement between the data and distribution, and the grey dots indicate the quantiles computed from the YouTube data.

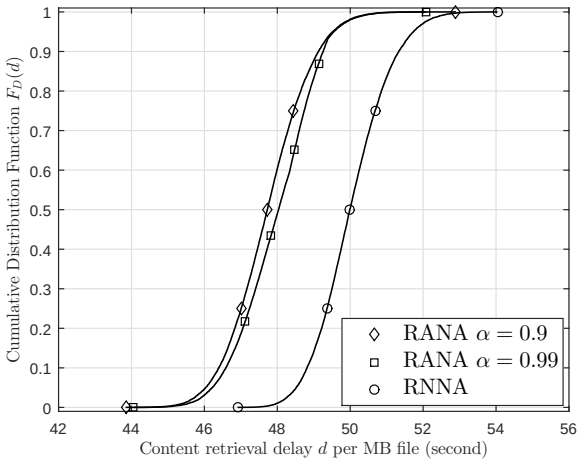


Fig. 8. Cumulative distribution function of the content retrieval delay $F_D(d)$ using the RNNA and RANA caching policies for a confidence level $\alpha = 0.9, 0.99$. The blue circles denote the RNNA caching policy (4), red diamond shape indicate the RANA caching policy (12) with $\alpha = 0.9$, and black squares for the RANA caching policy with $\alpha = 0.99$. As the value of $F_D(d) = P(D \leq d)$ increases, the probability the delay exceeds d decreases.

retrieval delay time given that the delay is greater than or equal to the VaR at α. The selection of the parameter $\alpha \in [0, 1]$ is determined by the network operator. In this section we evaluate the cumulative distribution function $F_D(d)$ to evaluate the probability that the delay D is less than the threshold d using the RNNA and RANA caching policies for a given confidence level α. Given $F_D(d)$, the network operator can confidently select the confidence level α to guarantee that the delay does not exceed the given threshold d_{th}.

Fig. 8 provides the cumulative distribution function $F_D(d)$ for the content retrieval delay for confidence levels $\alpha = 0.9$ and $\alpha = 0.99$ using the RANA caching policy, and the RNNA caching policy. From Fig. 8, the RANA caching policy (12) provides better performance compared to the RNNA caching policy (4) for all delay threshold values d. This results as the RANA caching policy accounts for the uncertainty associated with estimating the content requests. For example, if we are interested in the probability the delay is less than the threshold $d = 50$ seconds, the RNNA policy is approximately 50%, while the RANA policies are approximately 98%. A substantial improvement in performance is obtained by accounting for the prediction error associated with the caching decisions. The associated delay of between the RANA caching policies with $\alpha = 0.99$ and $\alpha = 0.9$ are approximately equal except in the delay range of 46 seconds to 49 seconds. In this region the selection of α is important. For example, if $d = 48$ seconds, then the probability the delay is 50% for RANA with $\alpha = 0.99$, and 60% RANA with $\alpha = 0.9$. Therefore, if the network operator wants to minimize the probability of the delay exceeding the threshold $d_{th} = 48$ seconds, an $\alpha = 0.9$ should be selected. The reason that a larger value of α does not guarantee minimizing $F_D(d)$, for a specific d, is that as $\alpha \to 1$, the maximum delay is being minimized in (12).

6.4 Performance of the Risk-Neutral and Risk-Aware Caching Policies

In this section, we illustrate the performance of the four caching policies (RN and RNNA) in Secs.4.1, 4.2, and (RA and RANA) in Secs.4.4, 4.5. Additionally, we compare the performance of these four caching policies with the caching policy presented in [13]. The caching policy in [13] accounts for the content requests, and LTE network parameters (cache size, bandwidth, latency among nodes), however it does not account for the routing protocol used in the network or the uncertainty associated with the estimated content requests. We refer to the caching policy in [13] as the risk-neutral without routing (RNWR) caching policy.

The metric of performance we use to compare these five caching policies is the cumulative distribution function $F_D(d)$ for the content retrieval delay. To estimate $F_D(d)$, we set $K = 10,000$ and $\alpha = 0.9$, and generate $20,000$ samples of the total content retrieval delay D. Fig. 9 illustrates empirical $F_D(d)$ from the five caching policies. The results in Fig. 9 illustrate that the RA caching policy (11) has a lower delay than the RN caching policy (3) for all possible values of d. That is, the delay that results from the RN caching policy illustrates a first-order stochastic dominance compared with the delay that results from the RA caching policy. Additionally, the delay that results from all the caching policies illustrate a first-order stochastic dominance compared with the RANA caching policy. The delay of the RN and RA policies are approximately twice as large as compared with the RNWR policy which accounts for the LTE network parameters but not the routing protocol used in the network. Therefore, including network parameters can substantially reduce the delay that results when performing a caching decision. Comparing the results for the RNWR, RNNA, and RANA, both the RNNA and RANA caching policies significantly reduce the content retrieval delay in the network compared with the RNWR caching policy by approximately 25%. This results as the RNNA and RANA policies both reduce the congestion of transferring content throughout the network as they account for the network routing protocol used. Finally, the RANA policy pro-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2018.2855160, IEEE Transactions on Cloud Computing

vides the lowest content retrieval delay compared with the other four policies as it accounts for the uncertainty of content requests, LTE network parameters, and the routing protocol used to transfer content throughout the network.

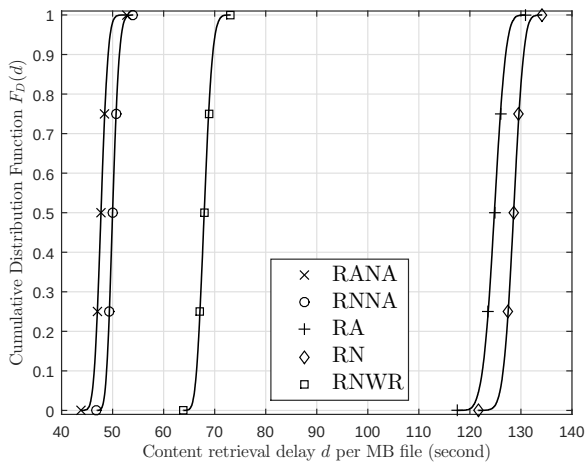


Fig. 9. The cumulative distribution function $F_D(d)$ of the content retrieval delay for the RN, RNNA, RA, RANA, RNWR caching policies. To evaluate $F_D(d)$ for the RA and RANA caching policies, we set $K = 10,000$ and $\alpha = 0.9$ in (11) and (12). A total of $20,000$ samples are generated for the content retrieval delay to construct the empirical cdf $F_D(d)$. The results illustrate that the delay of the RN, RNNA, RA, and RNWR policies all first-order stochastically dominate the delay associated with the RANA caching policy.

7 Conclusion

In this paper we designed risk-neutral and risk-averse caching policies for femtocell networks that contain femtocell access points which have limited storage capacity and low bandwidth backhaul links. The risk-averse caching policies employed the coherent Conditional Value-at-Risk (CVaR) measure to account for the uncertainty of estimating the content requests to perform the caching decisions. The CVaR risk measure is evaluated using information from the conformal prediction algorithm which constructs the cumulative distribution function of the content requests based on the content features. Using real-world datasets from YouTube and the NS-3 simulator, we demonstrate how the caching policies reduce the delay of retrieving content in femtocell networks compared with industry standard caching policies. The results show that a 6% reduction in the average delay can be achieved if the uncertainty of the content requests is accounted for, and a 60% reduction in average delay is achieved if both the uncertainty and femtocell routing protocol are accounted for compared to the risk-neutral caching that neglects the routing protocol.

References

[1] H. Pinto, J. Almeida, and M. Gonçalves, "Using early view patterns to predict the popularity of YouTube videos," *in Proceedings of the sixth ACM international conference on Web search and data mining*, pp. 365–374, 2013.

[2] Z. Tan, Y. Wang, Y. Zhang, and J. Zhou, "A novel time series approach for predicting the long-term popularity of online videos," *IEEE Transactions on Broadcasting*, vol. 62, no. 2, pp. 436–445, 2016.

[3] J. Wu, Y. Zhou, M. Chiu, and Z. Zhu, "Modeling dynamics of online video popularity," *IEEE Transactions on Multimedia*, vol. 18, no. 9, pp. 1882–1895, 2016.

[4] C. Li, J. Liu, and S. Ouyang, "Characterizing and predicting the popularity of online videos," *IEEE Access*, vol. 4, pp. 1630–1641, 2016.

[5] R. Zhou, S. Khemmarat, L. Gao, J. Wan, J. Zhang, Y. Yin, and J. Yu, "Boosting video popularity through keyword suggestion and recommendation systems," *Neurocomputing*, vol. 205, pp. 529–541, 2016.

[6] W. Hoiles, A. Aprem, and V. Krishnamurthy, "Engagement and popularity dynamics of YouTube videos and sensitivity to meta-data," *IEEE Transactions on Knowledge & Data Engineering*, no. 7, pp. 1426–1437, 2017.

[7] A. Tay and K. Wallis, "Density forecasting: a survey," *Journal of forecasting*, vol. 19, no. 4, p. 235, 2000.

[8] D. Hamilton, *Time series analysis*. Princeton university press, 1994, vol. 2.

[9] L. Fang and D. Bessler, "Stock returns and interest rates in china: the prequential approach," *Applied Economics*, pp. 1–14, 2017.

[10] K. Shanmugam, N. Golrezaei, A. Dimakis, A. Molisch, and G. Caire, "Femtocaching: Wireless content delivery through distributed caching helpers," *IEEE Transactions on Information Theory*, vol. 59, no. 12, pp. 8402–8413, 2013.

[11] B. Bharath, K. Nagananda, and H. V. Poor, "A learning-based approach to caching in heterogenous small cell networks," *IEEE Transactions on Communications*, vol. 64, no. 4, pp. 1674–1686, 2016.

[12] J. Song, H. Song, and W. Choi, "Optimal content placement for wireless femto-caching network," *IEEE Transactions on Wireless Communications*, vol. 16, no. 7, pp. 4433–4444, 2017.

[13] S. Tanzil, W. Hoiles, and V. Krishnamurthy, "Adaptive scheme for caching youtube content in a cellular network: Machine learning approach," *IEEE Access*, vol. 5, pp. 5870–5881, 2017.

[14] J. He and W. Song, "Optimizing video request routing in mobile networks with built-in content caching," *IEEE Transactions on Mobile Computing*, vol. 15, no. 7, pp. 1714–1727, 2016.

[15] M. Dehghan, B. Jiang, A. Seetharam, T. He, T. Salonidis, J. Kurose, D. Towsley, and R. Sitaraman, "On the complexity of optimal request routing and content caching in heterogeneous cache networks," *IEEE/ACM Transactions on Networking*, vol. 25, no. 3, pp. 1635–1648, 2016.

[16] K. Poularakis, G. Iosifidis, A. Argyriou, and L. Tassiulas, "Video delivery over heterogeneous cellular networks: Optimizing cost and performance," *"in Proceedings of the IEEE INFOCOM*, pp. 1078–1086, 2014.

[17] T. Rockafellar and S. Uryasev, "Optimization of conditional value-at-risk," *Journal of risk*, vol. 2, pp. 21–42, 2000.

[18] ——, "Conditional value-at-risk for general loss distributions," *Journal of banking & finance*, vol. 26, no. 7, pp. 1443–1471, 2002.

[19] P. Krokhmal, J. Palmquist, and S. Uryasev, "Portfolio optimization with conditional value-at-risk objective and constraints," *Journal of risk*, vol. 4, pp. 43–68, 2002.

[20] M. A. Khan, H. Tembine, and A. V. Vasilakos, "Game dynamics and cost of learning in heterogeneous 4g networks," *IEEE Journal on Selected Areas in Communications*, vol. 30, no. 1, pp. 198–213, 2012.

[21] M. R. Rahimi, J. Ren, C. H. Liu, A. V. Vasilakos, and N. Venkatasubramanian, "Mobile cloud computing: A survey, state of art and future directions," *Mobile Networks and Applications*, vol. 19, no. 2, pp. 133–143, 2014.

[22] G. Zhang, T. Q. Quek, M. Kountouris, A. Huang, and H. Shan, "Fundamentals of heterogeneous backhaul design–analysis and optimization," *IEEE Transactions on Communications*, vol. 64, no. 2, pp. 876–889, 2016.

[23] C.-Y. Wang, C.-H. Ko, H.-Y. Wei, and A. V. Vasilakos, "A voting-based femtocell downlink cell-breathing control mechanism," *IEEE/ACM Transactions on Networking (TON)*, vol. 24, no. 1, pp. 85–98, 2016.

[24] M. R. Rahimi, N. Venkatasubramanian, S. Mehrotra, and A. V. Vasilakos, "On optimal and fair service allocation in mobile cloud computing," *IEEE Transactions on Cloud Computing*, 2015.

[25] D. L. Perez, X. Chu, A. V. Vasilakos, and H. Claussen, "Power minimization based resource allocation for interference mitigation in ofdma femtocell networks," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 2, pp. 333–344, 2014.

[26] D. Lin, Y. Tang, Y. Yao, and A. V. Vasilakos, "User-priority-based power control over the d2d assisted internet of vehicles for mobile

health," *IEEE Internet of Things Journal*, vol. 4, no. 3, pp. 824–831, 2017.
[27] S. Muller, O. Atan, M. van der Schaar, and A. Klein, "Context-aware proactive content caching with service differentiation in wireless networks," *IEEE Transactions on Wireless Communications*, vol. 16, no. 2, pp. 1024–1036, 2017.
[28] Q. Huang, K. Birman, R. van Renesse, W. Lloyd, S. Kumar, and H. C. Li, "An analysis of Facebook photo caching," *in Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, pp. 167–181, 2013.
[29] K. Shanmugam, N. Golrezaei, A. G. Dimakis, A. F. Molisch, and G. Caire, "Femtocaching: Wireless content delivery through distributed caching helpers," *IEEE Transactions on Information Theory*, vol. 59, no. 12, pp. 8402–8413, 2013.
[30] E. Bastug, M. Bennis, and M. Debbah, "Living on the edge: The role of proactive caching in 5g wireless networks," *IEEE Communications Magazine*, vol. 52, no. 8, pp. 82–89, 2014.
[31] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.
[32] M. Zeni, D. Miorandi, and F. De Pellegrini, "YOUStatAnalyzer: a tool for analysing the dynamics of youtube content popularity," *in Proceedings of the 7th International Conference on Performance Evaluation Methodologies and Tools*, pp. 286–289, 2013.
[33] G. L. Nemhauser and L. A. Wolsey, "Integer programming and combinatorial optimization," *Wiley*, 1988.
[34] D. L. Perez, X. Chu, A. V. Vasilakos, and H. Claussen, "On distributed and coordinated resource allocation for interference mitigation in self-organizing lte networks," *IEEE/ACM Transactions on Networking*, vol. 21, no. 4, pp. 1145–1158, 2013.
[35] D. Silva, "Two-group classification with high-dimensional correlated data: A factor model approach," *Computational Statistics & Data Analysis*, vol. 55, no. 11, pp. 2975–2990, 2011.
[36] W. Griffiths, "A Gibbs' sampler for the parameters of a truncated multivariate normal distribution," *Contemporary issues in economics and econometrics: Theory and application*, pp. 75–91, 2004.
[37] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," *in Proceedings of the International Conference on Machine Learning*, pp. 1139–1147, 2013.
[38] R. Rere, M. Fanany, and A. Arymurthy, "Simulated annealing algorithm for deep learning," *Procedia Computer Science*, vol. 72, pp. 137–144, 2015.
[39] M. Mavrovouniotis and S. Yang, "Training neural networks with ant colony optimization algorithms for pattern classification," *Soft Computing*, vol. 19, no. 6, pp. 1511–1522, 2015.
[40] A. Choromanska, M. Henaff, M. Mathieu, G. B. Arous, and Y. LeCun, "The loss surfaces of multilayer networks," *Artificial Intelligence and Statistics*, pp. 192–204, 2015.
[41] G. Huang, Q. Zhu, and C. Siew, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, no. 1, pp. 489–501, 2006.
[42] G. Huang and L. Chen, "Enhanced random search based incremental extreme learning machine," *Neurocomputing*, vol. 71, no. 16, pp. 3460–3468, 2008.
[43] Q. He, T. Shang, F. Zhuang, and Z. Shi, "Parallel extreme learning machine for regression based on mapreduce," *Neurocomputing*, vol. 102, pp. 52–58, 2013.
[44] A. V. Vasilakos, Y. Tang, Y. Yao *et al.*, "Neural networks for computer-aided diagnosis in medicine: A review," *Neurocomputing*, vol. 216, pp. 700–708, 2016.
[45] J. Zhao, Z. Wang, and D. Park, "Online sequential extreme learning machine with forgetting mechanism," *Neurocomputing*, vol. 87, pp. 79–89, 2012.
[46] Y. Ye, S. Squartini, and F. Piazza, "Online sequential extreme learning machine in nonstationary environments," *Neurocomputing*, vol. 116, pp. 94–101, 2013.
[47] A. Basu, S. Shuo, H. Zhou, M. Lim, and G. Huang, "Silicon spiking neurons for hardware implementation of extreme learning machines," *Neurocomputing*, vol. 102, pp. 125–134, 2013.
[48] M. Chen, E. Zadok, A. O. Vasudevan, and K. Wang, "Seminas: A secure middleware for wide-area network-attached storage," *in Proceedings of the 9th ACM International on Systems and Storage Conference*, pp. 1–13, 2016.
[49] S. Abolfazli, Z. Sanaei, M. Alizadeh, A. Gani, and F. Xia, "An experimental analysis on cloud-based mobile augmentation in mobile cloud computing," *IEEE Transactions on Consumer Electronics*, vol. 60, no. 1, pp. 146–154, 2014.
[50] D. Applegate, A. Archer, V. Gopalakrishnan, S. Lee, and K. Ramakrishnan, "Optimal content placement for a large-scale vod system," *IEEE/ACM Transactions on Networking*, vol. 24, no. 4, pp. 2114–2127, 2016.
[51] S. Mastorakis, A. Afanasyev, I. Moiseenko, and L. Zhang, "ndnSIM 2: An updated NDN simulator for NS-3," NDN, Technical Report NDN-0028, Revision 2, November 2016.
[52] A. V. Vasilakos, Z. Li, G. Simon, and W. You, "Information centric network: Research challenges and opportunities," *Journal of Network and Computer Applications*, vol. 52, pp. 1–10, 2015.

William Hoiles received his M.A.Sc. degree in 2012 from the Department of Engineering Science, Simon Fraser University, Burnaby, Canada, and Ph.D. from Electrical and Computer Engineering, University of British Columbia, Vancouver, Canada, in 2015. He was a postdoctoral researcher in Electrical Engineering at the University of California, Los Angeles in 2016, and is currently a postdoctoral lecturer in Electrical and Computer Engineering at the University of British Columbia. His current research interests are social sensors and the bioelectronic interface.

S M Shahrear Tanzil received the B.Sc. degree in electrical and electronics engineering from Bangladesh University of Engineering and Technology (BUET), Bangladesh, in 2011 and the M.A.Sc. degree from the University of British Columbia (UBC), Canada, in 2013. He is currently working towards the Ph.D. degree at UBC and is a member of the Statistical Signal Processing Laboratory. His research interests include wireless networks, mobile cloud computing, and game theory.

Vikram Krishnamurthy (S'90–M'91–SM'99–F'05) received the Ph.D. degree from the Australian National University, Canberra, Australia, in 1992. He is currently a professor at the Department of Electrical and Computer Engineering, Cornell University. From 2002-2016 he was a Professor and Canada Research Chair at the University of British Columbia, Canada. His research interests include statistical signal processing, computational game theory, and stochastic control in social networks. He served as Distinguished Lecturer for the IEEE Signal Processing Society and Editor-in-Chief of the IEEE Journal on Selected Topics in Signal Processing. In 2013, he was awarded an Honorary Doctorate from KTH (Royal Institute of Technology), Sweden. He is author of the book *Partially Observed Markov Decision Processes* published by Cambridge University Press in 2016.