Estimating Instantaneous Cache Hit Ratio Using Markov Chain Analysis

Hazem Gomaa, *Student Member, IEEE*, Geoffrey G. Messier, *Member, IEEE*, Carey Williamson, *Member, IEEE*, and Robert Davies, *Member, IEEE*

***Abstract*—This paper introduces a novel analytical model for estimating the cache hit ratio as a function of time. The cache may not reach the steady-state hit ratio when the number of Web objects, object popularity, and/or caching resources themselves are subject to change. Hence, the only way to quantify the hit ratio experienced by Web users is to calculate the instantaneous hit ratio. The proposed analysis considers a single Web cache with infinite or finite capacity. For a cache with finite capacity, two replacement policies are considered: Least Recently Used (LRU) and First-In–First-Out (FIFO). Based on the insights from the proposed analytical model, we propose a new replacement policy, called Frequency-Based-FIFO (FB-FIFO). The results show that FB-FIFO outperforms both LRU and FIFO, assuming that the number of Web objects is fixed. Assuming that new popular objects are generated periodically, the results show that FB-FIFO adapts faster than LRU and FIFO to the changes in the popularity of the cached objects when the cache capacity is large relative to the number of newly generated objects.**

***Index Terms*—Analysis, caching, Markov chain, replacement policy, Web.**

I. Introduction

CACHING Web objects close to end-users allows user requests to be satisfied from a nearby Web cache (cache hit) rather than the origin Web server (cache miss). Cache hits reduce the load on origin Web servers and achieve bandwidth savings over costly Internet links. These bandwidth savings translate into reducing both wide-area network congestion and response time of Web requests. On the other hand, cache misses result in a long response time and extra processing overhead [1], [2]. Caching is critical not only in network applications but also in microprocessor systems. Very fast caches are usually used to hide the speed gap between main memory and the CPU [3]–[5].

The cache performance is commonly evaluated in terms of hit ratio. The hit ratio is the ratio between the number of cache hits and the total requests observed over a period of time. Estimating the hit ratio can be done using analytical models [6]–[16] or simulations [17]–[22]. Analytical models provide more insight into the factors that affect the cache hit ratio. Analytical models, if simple and tractable, allow immediate prediction of the sensitivity of the hit ratio to different factors, such as cache capacity and object expiry rate. Consequently, analytical models, rather than simulations, may allow efficient replacement policies to be developed [14], [23].

The goal of this paper is to develop an analytical model for estimating the hit ratio of a single cache as a function of time. In this paper, this will be referred to as *instantaneous hit ratio*. Estimating the instantaneous hit ratio is important for any application where the number of objects, object popularity, and/or the caching resources themselves are subject to change. Consider a scenario, such as a group of users watching sports highlights or accessing vehicular traffic information, where objects experience only short-term popularity. Here, it is important for a cache to reach maximum performance quickly before the popularity of the cached objects starts to drop. Similarly, a vehicular or mobile network may store cached objects on mobile devices [24]–[27]. In this case, it is also important for the cache to reach high hit ratio quickly before the mobile device that is hosting the cache disconnects from the network.

In this paper, our concern is to evaluate how the cache hit ratio evolves as a function of time in two cases. The first case assumes that the number of Web objects and the object popularity are fixed throughout the evaluation interval. In this case, we evaluate the instantaneous hit ratio in the transient period starting from an empty cache. The second case adopts more realistic assumptions by assuming that the number of Web objects increases periodically. It is assumed that the most recently generated objects become the most popular ones (news headlines for example). Hence, the popularity of the formerly cached objects decreases every time new objects are generated. In this case, the cache may not reach the steady-state hit ratio, and the only way to quantify the hit ratio experienced by Web users is to calculate the instantaneous hit ratio.

The main contribution of this paper is a novel analytical model for estimating the instantaneous hit ratio of a single cache. It is assumed that each object brought into the cache has a finite lifetime beyond which it becomes obsolete and is ejected [1], [28]. The proposed analytical model accommodates either infinite or finite cache capacity. If the cache has infinite capacity (Infinite Cache), then it stores all the requested objects. For the finite case, the cache applies a replacement policy to decide which objects to keep. This paper introduces analytical solutions for two replacement policies: Least Recently Used (LRU), and First-In–First-Out (FIFO). LRU ejects the least recently

Manuscript received September 20, 2011; revised March 13, 2012; August 17, 2012; and October 23, 2012; accepted October 25, 2012; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor T. Bonald. Date of publication December 11, 2012; date of current version October 11, 2013. This work was supported by TRLabs, Alberta, Canada.

H. Gomaa, G. G. Messier, and R. Davies are with the Department of Electrical and Computer Engineering, Schulich School of Engineering, University of Calgary, Calgary, AB T2N 1N4, Canada (e-mail: hagomaa@ucalgary.ca; gmessier@ucalgary.ca; davies@ucalgary.ca).

C. Williamson is with the Department of Computer Science, University of Calgary, Calgary, AB T2N 1N4, Canada (e-mail: carey@cpsc.ucalgary.ca).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNET.2012.2227338

requested object, while FIFO ejects the object that was brought into the cache earliest [3], [14], [29]. The proposed analytical model can also be extended for other replacement policies such as Most Recently Used (MRU) and Last-In–First-Out (LIFO). Furthermore, based on the insights from the proposed analytical model, we propose a new replacement policy, called Frequency Based-FIFO (FB-FIFO). FB-FIFO improves the instantaneous hit ratio of FIFO by creating a variable-size protected cache segment for objects that are requested more than once within a short time span.

In this paper, the performance of LRU, FIFO, and FB-FIFO is compared to Perfect-LFU using the independent reference model (IRM) [12], [29], [30]. Perfect-LFU ejects the least frequently used object from the cache and keeps a record of the number of requests for each object even after the object is ejected. Thus, under the IRM, Perfect-LFU achieves better steady-state hit ratio than LRU and FIFO.

The results show that different replacement policies have very different transient performance, whether the popularity of cached objects is fixed or not. This makes our analysis valuable when considering scenarios like changing popularity or mobile caching. The following list summarizes the major findings in this paper.

- When the number of Web objects and the object popularity are fixed during the transient period:
  - Unlike the other replacement policies studied in this paper, FIFO reaches its maximum hit ratio during the transient fluctuations in hit ratio that occur before reaching steady state. These fluctuations diminish as the cache capacity decreases, or as the object expiry rate increases with respect to the request rate.
  - FB-FIFO outperforms both LRU and FIFO, especially when cache capacity is small. Moreover, FB-FIFO matches Perfect-LFU for a longer time as the cache capacity increases.
  - As the object expiry rate increases with respect to the request rate, the instantaneous hit ratio for Infinite Cache decreases more rapidly than Perfect-LFU, FB-FIFO, LRU, and FIFO, which decreases the least.
- When new popular objects are generated periodically:
  - FB-FIFO outperforms LRU and FIFO and adapts faster than the other replacement policies to the changes in the popularity of the cached objects (i.e., FB-FIFO is more robust) when the cache capacity is large relative to the number of the newly generated objects.
  - For small caches, the robustness of FB-FIFO improves as the object expiry rate increases.

Note that these findings are valid under the IRM, which is widely used to develop tractable analytical caching models. However, the IRM might not accurately model empirical traces that have varying degrees of temporal correlation between object requests [9], [31]. Thus, an interesting extension of the proposed model would be to incorporate temporal locality based on the LRU stack model (e.g., [18]).

The remainder of the paper is organized as follows. First, the related work is summarized in Section II. In Section III, the analysis assumptions are discussed. In Section IV, the analysis for Infinite Cache is presented. In Section V, the analysis for LRU and FIFO is presented. In Section VI, the proposed replacement policy FB-FIFO is described. Section VII presents the analytical and simulation results. Finally, Section VIII concludes the paper.

II. Related Work

Analytical models for estimating the steady-state hit ratio for a single cache with infinite capacity were proposed in [10]–[12].

The analytical model proposed by Wolman *et al.* [10] considered object lifetime, which is exponentially distributed and correlated with the object popularity. The authors showed that the steady-state hit ratio is very sensitive to the object expiry rate. They also showed that the increase in the request rate, relative to the expiry rate, enhances the steady-state hit ratio. In our paper, we investigate how the ratio between the object expiry rate and request rate impacts the transient behavior of LRU and FIFO.

Breslau *et al.* [12] proposed an analytical model to estimate the steady-state hit ratio of Infinite Cache and Perfect-LFU, assuming objects that do not expire. Breslau *et al.* [12] showed that the steady-state hit ratio for Perfect-LFU increases logarithmically or as a small power as a function of cache size. On the other hand, our analytical model assumes cached objects with limited lifetime and provides more insight into the evolution of Infinite Cache with time.

The analytical model proposed by Rodriguez *et al.* [11] focused on the performance of cooperative caching schemes. The proposed model in [11] aims at studying the performance of hierarchical and distributed caching in terms of the hit ratio, the client's perceived latency, the bandwidth usage, the load in the caches, and the disk space usage.

The main difference between our study and the study in [11] is that our study focuses on providing an analytical solution for a single cache with finite capacity.

Analytical models for estimating steady-state hit ratio for LRU and FIFO were also introduced in [6]–[9] and [13]–[15].

Gelenbe [6] extended the analysis provided for FIFO in [30] in order to show that FIFO and Random replacement policies reach the same steady-state hit ratio under the IRM. Dan *et al.* [14] developed approximate analytical models for predicting the steady-state hit ratio of LRU and FIFO under the IRM. The study in [14] showed that LRU always outperforms FIFO in steady state. This method was then used by Laoutaris *et al.* [16] for studying distributed caching in the context of LRU.

Jelenkovic [7] showed that computing the LRU fault probability is the same as computing the Move-To-Front search cost distribution, assuming the IRM. Also, the study in [15] proposed an analytical model for estimating the steady-state hit ratio of a single cache running LRU under the IRM. Moreover, Jelenkovic *et al.* [8] showed that the caching performance does not depend on the correlation in the request traffic for large cache sizes [32]. Furthermore, Jelenkovic *et al.* [9] determined the smallest cache size above which the cache performance does not depend on the temporal correlation.

The analytical models introduced in [6]–[9], [14], and [15], however, assumed a fixed number of objects that do not expire. The study by Mookerjee *et al.* [13] provided an analytical model for estimating the browser cache hit ratio and access

delay assuming local caches running LRU replacement policy, and a fixed number of objects that expire periodically.

While the previous studies focus on estimating the steady-state hit ratio of a single Web cache running LRU or FIFO, to the best of our knowledge, our work is the first attempt to develop an analytical model to study the evolution of LRU and FIFO with time, assuming cached objects that have limited lifetimes and popularity that decreases periodically.

III. Analysis Assumptions

The proposed analysis models the caching activity of a single Web cache during a fixed evaluation interval $[0, T]$. At the beginning of the evaluation interval (i.e., time $t = 0$), the Web server generates M_0 new Web objects, and the Web cache is assumed to be empty. At time $t > 0$, the Web server may generate new Web objects. Thus, the total number of Web objects stored at the Web server increases to $M(t)$.

Consider an interval where the number of objects is fixed, such that $M(t) \simeq M$. The requests for those M objects arrive at the server according to a Poisson process with rate β. The probability of each request being for one of the M objects is determined by a Zipf-like distribution [12]. In this distribution, the probability of the ith most popular object being selected is $1/\sigma i^{\alpha}$, where $\sigma = \sum_{i=1}^{M} 1/i^{\alpha}$ and α is the Zipf slope such that $0 < \alpha \leq 1$.

The overall Poisson request process at the server can be modeled as a sum of M independent Poisson processes that each represent the requests for one of the M objects [33]. The average request rate of the Poisson process for object i at time t, where $1 \leq i \leq M$, is

$$\lambda_i = \frac{\beta}{\sigma i^{\alpha}}.$$

The user requests are directed to the Web cache. If the cache has the requested object, the user downloads the object from the cache. Otherwise, the cache downloads the requested object from the origin Web server and stores a copy of the object to satisfy future requests. While downloading the object, the cache forwards the object to the requesting user.

The analysis also assumes that the cached object i has limited lifetime that is exponentially distributed [10] with mean $1/\mu_i$. The object lifetime is independent from both the object size and the object popularity.

It is assumed that α, β, and μ_i are not subject to change within the evaluation interval $[0, T]$. However, the proposed analysis can be extended to cover these cases, which are excluded from this paper merely for conciseness.

The proposed analysis considers both infinite and finite cache capacity. If infinite cache capacity (Infinite Cache) is assumed, then the cache stores all requested objects. Otherwise, for a cache with finite capacity, a replacement policy is applied to make room for the requested object if the cache is full. We define the cache capacity C as the average number of objects that can be cached simultaneously, which approximately equals the cache size divided by the average object size. It is assumed that the object size is independent of object popularity [12], [13].

In the following sections, continuous-time Markov chain analysis [34] is used to calculate the instantaneous hit ratio for M objects, $H(t)$. Since the Poisson request processes for each of the M objects are independent, it is possible to start by analyzing the hit ratio of object i at time t, $H_i(t)$. Then, $H(t)$ will be calculated as the sum of individual object hit ratios weighted by their probability of request as follows:

$$H(t) = \sum_{i=1}^{M} \frac{H_i(t)}{\sigma i^{\alpha}}. \tag{1}$$

An analytical solution for finding $H(t)$ for Infinite Cache, LRU, and FIFO is developed in the following sections. The solution for Infinite Cache is discussed in Section IV. The solution for LRU and FIFO is described in Section V.

IV. Analysis of Infinite Cache

In the Infinite Cache, an object that enters the cache is never evicted by a replacement policy. They are merely ejected when they expire. The state of the cache with respect to an object i can be modeled using a two-state Markov chain. State 0 corresponds to when object i is not in the cache. State 1 corresponds to when object i is in the cache. Note that the rate at which the object moves from state 0 to 1 is equal to the object request rate, and the rate the process returns to state 0 is equal to the object expiry rate. The Markov chain flow matrix of object i, $\mathbf{Q}_i$, is given by

$$\mathbf{Q}_i = \begin{bmatrix} -\lambda_i & \lambda_i \\ \mu_i & -\mu_i \end{bmatrix}. \tag{2}$$

Assume that the number of objects increases at time $t_0 + \delta t$ to $M(t_0 + \delta t)$. If $M(t_0 + \delta t)$ remains fixed to time t, then the probability matrix of object i at time t, $\mathbf{P}_i(t)$, is calculated using

$$\mathbf{P}_i(t) = \exp(\mathbf{Q}_i \quad \Delta t) \tag{3}$$

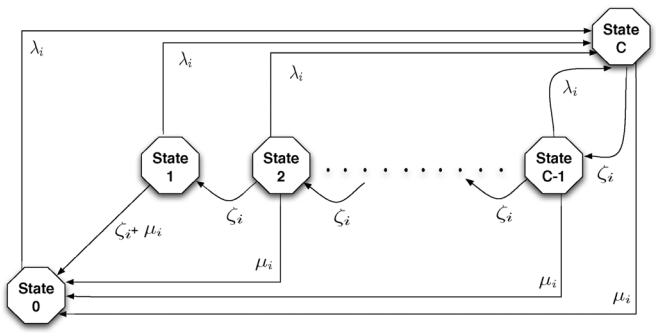
where $\Delta t = t - t_0$. Note that $\exp(.)$ in (3) is the matrix exponential operator [34]. Thus

$$\begin{aligned} [\mathbf{P}_i(t)]_{1,2} &= \frac{\lambda_i}{\lambda_i + \mu_i} - \frac{\lambda_i}{\lambda_i + \mu_i} e^{-(\lambda_i + \mu_i)\Delta t} \\ [\mathbf{P}_i(t)]_{2,2} &= \frac{\lambda_i}{\lambda_i + \mu_i} + \frac{\mu_i}{\lambda_i + \mu_i} e^{-(\lambda_i + \mu_i)\Delta t} \end{aligned} \tag{4}$$

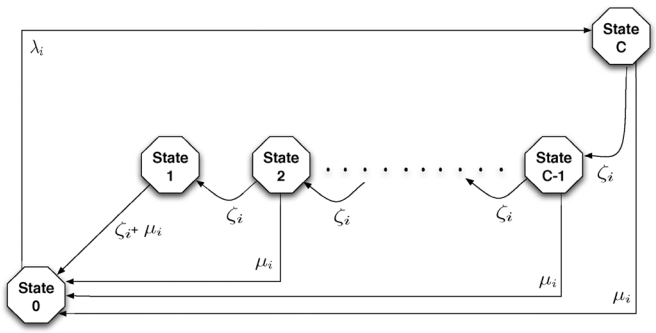
where the notation $[.]_{r,c}$ denotes the element in row r and column c of a matrix [33]. The instantaneous hit ratio of object i at time t, $H_i(t)$, is calculated using (5), where $P_{i,j}(t)$ denotes the probability that object i is in state j at time t, such that $\sum_{j=0}^{1} P_{i,j}(t) = 1$

$$\begin{aligned} H_i(t) &= P_{i,1}(t) \\ &= \sum_{r=1}^{2} [\mathbf{P}_i(t)]_{r,2} \, P_{i,r-1}(t_0). \end{aligned} \tag{5}$$

The instantaneous hit ratio is then calculated using (1). Note that calculating $H(t)$ for $M(t)$ objects requires $M(t)$ Markov chains, each with two states.



(a)



(b)

Fig. 1. (a) LRU and (b) FIFO: Markov chains for estimating the lower-bound instantaneous hit ratio of object i.

V. Analysis of LRU and FIFO

This section describes an iterative algorithm for estimating the instantaneous hit ratio $H(t)$ for both LRU and FIFO, assuming a finite cache capacity $C < M(t)$. The first iteration of the algorithm determines a lower-bound hit ratio, and then the estimate of the hit ratio is improved in the following iterations until no more improvement can be obtained. In the proposed iterative algorithm, a Markov chain, which has $C+1$ states, is used to model each object in the cache. In this Markov chain, state 0 corresponds to when object i is not in the cache (non-caching state), and states from 1 to C correspond to when object i is in cache (caching states). The hit ratio of object i at time t is simply the probability that object i occupies one of those caching states at time t.

The first iteration of the proposed algorithm is described in Section V-A, while Section V-B discusses how to obtain an exact estimate for $H(t)$.

A. Estimating Lower-Bound Hit Ratio

Fig. 1(a) shows the LRU Markov chain for estimating the lower-bound hit ratio of object i, $H_i^1(t)$. The caching states from 1 to C represent how recently an object has been requested. For example, the Markov chain for an object is in state C if it is the most recently requested object, state $C-1$ if it is the second most recently requested object, and so on.

In the LRU Markov chain proposed in Fig. 1(a), the lower-bound hit ratio of object i is calculated under two assumptions that cause the analysis to underestimate the exact hit ratio $H_i(t)$ for both LRU and FIFO.

First, we assume that a request for any object $k \neq i$ results in moving the cached object i from a current state j to the state $j-1$. Therefore, the transition rate for object i from any state j to the state $j-1$, $R_{j,j-1}$, equals $\zeta_i = \beta - \lambda_i$ except for $j=1$, where $R_{1,0} = \mu_i + \zeta_i$. In LRU, this assumption is only valid when the object k is in a state below the current state of object i. In FIFO, this assumption is only valid when the object k is not cached. The result is an underestimation of hit ratio since the transition rate $R_{j,j-1}$ is overestimated.

Second, we ignore the case when an object $k \neq i$ expires while objects k and i are in cache. Therefore, the transition rate for object i from a current caching state j to the caching state $j+1$ due to the expiry of object k equals zero. In both FIFO and LRU, this assumption is not valid if the object k is in a state above the current state of object i. In this case, the transition rate of object i between caching state j and caching state $j+1$ due to the expiry of object k exceeds zero. Therefore, the second assumption leads to underestimating the hit ratio since the transition rate, $R_{j,j+1}$, between caching state j and caching state $j+1$ is underestimated.

The LRU Markov chain flow matrix for estimating the lower-bound hit ratio of object i, $\mathbf{Q}_i^1$, is given by

$$\mathbf{Q}_i^1 = \begin{bmatrix} -\lambda_i & 0 & \dots & 0 & 0 & \lambda_i \\ \zeta_i+\mu_i & -\beta-\mu_i & \dots & 0 & 0 & \lambda_i \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \mu_i & 0 & \dots & \zeta_i & -\beta-\mu_i & \lambda_i \\ \mu_i & 0 & \dots & 0 & \zeta_i & -\zeta_i-\mu_i \end{bmatrix}. \tag{6}$$

This flow matrix is used to calculate the lower-bound probability matrix $\mathbf{P}_i^1(t)$ using

$$\begin{aligned} \mathbf{P}_i^1(t) &= \exp\left(\mathbf{Q}_i^1 \Delta t\right) \\ &\approx \left(\mathbf{I} + \mathbf{Q}_i^1 \frac{\Delta t}{n}\right)^n \qquad \text{for large } n \end{aligned} \tag{7}$$

where $\mathbf{I}$ is the identity matrix [33]. The lower-bound hit ratio, $H_i^1(t)$, is calculated using

$$\begin{aligned} H_i^1(t) &= 1 - P_{i,0}^1(t) \\ &= 1 - \sum_{r=1}^{C+1} \left[\mathbf{P}_i^1(t)\right]_{r,1} P_{i,r-1}^1(t_0). \end{aligned} \tag{8}$$

Note that the superscript in $\mathbf{Q}_i^1$, $P_{i,0}^1(t)$, $\mathbf{P}_i^1(t)$, and $H_i^1(t)$ refers to the iteration number in the iterative algorithm. Note that it is assumed that the cache is empty at time $t=0$ (i.e., $P_{i,0}^1(t=0) = 1 \forall i \in [1, M]$).

The lower-bound hit ratio for FIFO is calculated using the same steps as LRU. The only difference between the Markov chains of FIFO and LRU is that LRU has additional paths to caching state C from the caching states 1 through $C-1$, as shown in Fig. 1. The FIFO Markov chain for estimating the lower-bound hit ratio of object i is shown in Fig. 1(b). The caching states from 1 to C represent the relative time at which object i was brought into the cache. State C corresponds to the most recently cached object. The recency of a cached object decreases as the object goes from caching state j to the state $j-1$.

State 1 corresponds to the oldest cached object. The flow matrix for this Markov chain is

$$\mathbf{Q}_i^1 = \begin{bmatrix} -\lambda_i & 0 & \dots & 0 & 0 & \lambda_i \\ \zeta_i + \mu_i & -\zeta_i - \mu_i & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \mu_i & 0 & \dots & \zeta_i & -\zeta_i - \mu_i & 0 \\ \mu_i & 0 & \dots & 0 & \zeta_i & -\zeta_i - \mu_i \end{bmatrix}. \tag{9}$$

Like LRU, the lower-bound probability matrix $\mathbf{P}_i^1(t)$ and the lower-bound hit ratio $H_i^1(t)$ are calculated for FIFO using (7) and (8), respectively.

The overall lower-bound hit ratio for $M(t)$ objects, $H^1(t)$, can then be calculated using (1). As stated above, the assumptions made in this section underestimate the hit ratio for LRU and FIFO. In Section V-B, we introduce the modifications required to obtain an exact estimate for the hit ratio $H(t)$

B. Estimating the Exact Hit Ratio

In this section, the lower-bound hit ratio $H^1(t)$ is refined to be an exact estimate of instantaneous hit ratio. First, we describe the second iteration that is used to calculate an enhanced estimate for the hit ratio, $H^2(t)$, using the lower-bound probability matrices obtained in the first iteration (i.e., $\mathbf{P}_i^1(t) \forall i \in [1, M]$). Then, we illustrate how this procedure is repeated in order to obtain an exact estimate for $H(t)$.

An enhanced estimate of the hit ratio of object i, $H_i^2(t)$, is calculated based on two main modifications in the LRU and FIFO Markov chains. These modifications alter the transition rates between caching state j and the states $j-1$ and $j+1$ (i.e., $R_{j,j-1}$ and $R_{j,j+1}$). As discussed in Section V-A, calculating $H_i^1(t)$ overestimates $R_{j,j-1}$ in LRU and FIFO. To overcome this problem in LRU, we exclude from ζ_i the requests for those objects that currently occupy caching states $w > j$. Thus, ζ_i in Fig. 2(a) is replaced by $\epsilon_{i,j}^2(t)$ in the LRU Markov chain such that

$$\epsilon_{i,j}^2(t) = \zeta_i - \sum_{k=1,k\neq i}^{M} \lambda_k \sum_{w=j+1}^{C} P_{k,w}^1(t). \tag{10}$$

Similarly, in FIFO, we exclude from ζ_i the requests for objects that are currently cached. Thus, ζ_i in Fig. 2(b) is replaced by $\epsilon_i^2(t)$ in the FIFO Markov chain such that

$$\epsilon_i^2(t) = \zeta_i - \sum_{k=1,k\neq i}^{M} \lambda_k \sum_{w=1}^{C} P_{k,w}^1(t). \tag{11}$$

The problem of underestimating $R_{j,j+1}$ in LRU and FIFO can be solved by incorporating the expiry rate of those objects that currently occupy caching states $w > j$. Thus, a quantity $\gamma_{i,j}^2(t)$, where $j \neq 0$, is added to the LRU and FIFO Markov chains as shown in Fig. 2(a) and 2(b) such that

$$\gamma_{i,j}^2(t) = \sum_{k=1,k\neq i}^{M} \mu_k \sum_{w=j+1}^{C} P_{k,w}^1(t). \tag{12}$$

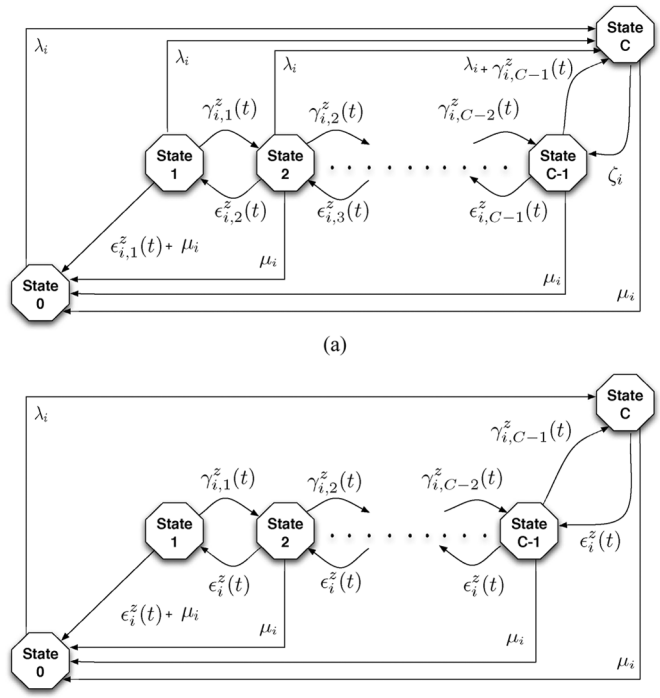


Fig. 2. (a) LRU and (b) FIFO: Markov chains for enhanced estimate for instantaneous hit ratio $H_i^z(t)$.

TABLE I
ITERATIVE ALGORITHM FOR ESTIMATING THE EXACT INSTANTANEOUS HIT RATIO USING LRU OR FIFO

- *Estimating lower-bound hit ratio*
  - Calculate: $\mathbf{P}_i^1(t)\ \forall\ i \in [1, M]$
  - Calculate: $H_i^1(t) = 1 - P_{i,0}^1(t)\ \forall\ i \in [1, M]$
  - Calculate: $H^1(t)$ using (1)
- $z = 1$
- Start Loop:
  - $z = z + 1$
  - *Enhanced estimates of the hit ratio*
    - Calculate: $\mathbf{P}_i^z(t)$ using $\mathbf{P}_i^{z-1}(t)\ \forall\ i \in [1, M]$
    - Calculate: $H_i^z(t) = 1 - P_{i,0}^z(t)\ \forall\ i \in [1, M]$
    - Calculate: $H^z(t)$ using (1)
  - If: $(H^z(t) \neq H^{z-1}(t))$ Go To: Start Loop
- End Loop:
- Output $H(t) = H^z(t)$

The final Markov chains for LRU and FIFO after applying the two proposed modifications are shown in Fig. 2(a) and (b). Similar to the process in Section V-A, a flow matrix $\mathbf{Q}_i^2$ can be defined for LRU or FIFO using the Markov chains illustrated in Fig. 2. Afterwards, enhanced estimates for the probability matrix $\mathbf{P}_i^2(t)$ and the hit ratio, $H_i^2(t)$ are calculated according to (7) and (8), respectively. In order to obtain an exact estimate for the hit ratio $H(t)$, the procedure used to obtain $H^2(t)$ is repeated to obtain further enhanced estimates of the hit ratio $H^3(t), H^4(t) \dots$ until no more enhancement is observed as shown in Table I.

Note that calculating an enhanced estimate z of the hit ratio, $H^z(t)$, requires using the previous estimate for the probability matrices (i.e., $\mathbf{P}_i^{z-1}(t) \forall i \in [1, M]$). Thus, the exact estimate of the hit ratio, $H(t)$, requires the exact probability matrices.

On the other hand, since the exact probability matrices are initially unknown, we started by finding the lower-bound probability matrices, which basically produce the lower-bound hit ratio. Afterwards, the probability matrices are enhanced iteratively until the exact probability matrices and exact hit ratio are obtained.

This proposed method of calculating $H(t)$ can be considered a contraction mapping [35], [36]. The Appendix contains a discussion of the convergence of the algorithm.

C. Discussion

The proposed Markov chain model provides a useful insight into the operation of LRU and FIFO. As shown in Fig. 2, the transition paths between caching states of FIFO and LRU are similar except that in LRU there are additional paths, with equal rates λ_i, to caching state C from caching states 1 through $C-1$. These additional paths in LRU exist since any object i becomes the most recently requested object as soon as it is requested, regardless of its current state.

The additional paths in LRU allow the current popular objects, which are recently having high request rates, to be organized in the higher caching states, while the current unpopular objects are more likely to occupy the noncaching state. Therefore, it is expected that LRU achieves better instantaneous hit ratio than FIFO, especially as the Zipf slope increases. On the other hand, the main drawback of LRU and FIFO is that one-time requested objects are immediately cached at the highest caching state C. One-time requested objects may further degrade the performance of FIFO since the cached object cannot change its position relative to other cached objects. Therefore, the key to enhance FIFO is to prevent unpopular objects from immediately occupying caching state C, as discussed in Section VI.

VI. FB-FIFO Replacement Policy

Unlike Perfect-LFU, LRU and FIFO do not store any information about the popularity of objects (i.e., number of requests), whether they are currently cached or not. Hence, LRU and FIFO can be easily implemented with a linked list, with length C. However, LRU requires more overhead on every cache hit to an object in order to change its state to C (i.e., move it to the front of the list as the most recently requested object) [37]. Many replacement policies are introduced in order to enhance the steady-state hit ratio of LRU, such as LRU-k and ARC, and LRFU [38]–[40]. These policies are even more complex than LRU, and they require additional resources.

In Section V, the Markov chain representation of FIFO showed that the hit ratio degrades because FIFO allows all new objects, unpopular or not, to occupy state C. State C is the position in the chain that allows the object to remain in the cache for the maximum amount of time. In this section, we take advantage of this insight to develop an improved version of FIFO algorithm, called FB-FIFO.

The main idea behind FB-FIFO is to create a variable-size protected segment in the cache (S_{p}) for objects that are requested more than once within a short time span. The remainder of the cache is considered an unprotected segment (S_{u}). In FB-FIFO, it is assumed that both cache segments are managed separately with the FIFO algorithm. When an uncached object is requested, the object is moved to S_{u} as the newest object. If S_{u} is full, the object that was brought into S_{u} earliest will be ejected from the cache. If an object in S_{u} experiences a cache hit, the object is moved to S_{p} as the newest object. If S_{p} is full, the object that was brought into S_{p} earliest will move back to S_{u} as the newest object. A counter, n, determines the capacity of S_{p}, while the capacity of S_{u} is $C-n$, such that $0 \leq n \leq N_{\max}$, and $N_{\max} < C$. The initial value of n is set to zero at time $t=0$. Every time an object in S_{u} experiences a cache hit, the value of n increments by one if $n < N_{\max}$. If $n = N_{\max}$, n cannot increment further. Note that if an object in S_{p} expires, it will be ejected from the cache and the value of n decrements by one. Note that objects in S_{p} do not move back to S_{u} if $n < N_{\max}$.

As n increases, the probability that a new cached object lingers in S_{u} decreases. When a new object is cached in S_{u}, it will be ejected if the next $C-n$ requests are all cache misses. Therefore, as n increases, the probability that the objects that are moved to S_{p} (as the newest objects) are popular increases. Furthermore, the one-time requested objects will not affect the S_{p} queue. Hence, FB-FIFO is expected to outperform both LRU and FIFO, which can easily be polluted by a sequence of requests for one-time requested objects.

FB-FIFO allows the current popular objects to be cached in S_{p}. In FB-FIFO, any object moves from S_{u} to S_{p} when it is requested within the next $C-n$ requests (excluding cache hits), regardless of the past average request rate of this object. Therefore, FB-FIFO is expected to adapt faster than Perfect-LFU to the changes in the popularity of the cached objects (i.e., FB-FIFO is more robust than Perfect-LFU), especially when the cached objects accumulate many requests over time. In this case, Perfect-LFU does not eject these cached objects even if they are never requested again. LFU-Aging [18] overcomes this problem by periodically reducing the counter values associated with cached objects.

The robustness of FB-FIFO may also decrease for small cache capacities, when $n = N_{\max}$, and objects do not expire (i.e., n cannot decrease). Assuming $n = N_{\max} = C-1$, only very popular objects can replace the past popular objects in S_{p} since the single object in S_{u} is ejected on the next cache miss. To overcome this problem, more complicated versions of FB-FIFO may be implemented to reset n to zero after a certain number of cache misses. In this paper, we merely evaluate the simplest version of the proposed FB-FIFO, where n decreases if and only if a cached object in S_{p} expires. Note that as the object expiry rate increases, the robustness of FB-FIFO improves for small caches since FB-FIFO allows the expired past popular objects in S_{p} to be ejected from the cache.

Note that determining a static value for $n = N_{\mathrm{static}}$ (i.e., fixed capacities for S_{u} and S_{p} throughout the evaluation interval) reduces the instantaneous hit ratio for FB-FIFO. If N_{static} is increased, the speed at which S_{p} fills up decreases. Consequently, the hit ratio increases very slowly in the transient period starting from an empty cache. Second, if N_{static} is decreased, the number of popular objects that can be stored in S_{p} decreases. Thus, the instantaneous hit ratio for FB-FIFO decreases until FB-FIFO matches FIFO when $N_{\mathrm{static}} = 0$.

TABLE II
OBJECT SIZE MODEL IN SIMULATIONS

Object Size Distribution	Lognormal
Smallest Object Size	1.00 MB
Median Object Size	7.06 MB
Average Object Size	8.02 MB
Largest Object Size	84.30 MB

TABLE III
EVALUATION FACTORS AND LEVELS

Factor	Symbol	Value
Number of Objects	M	5000 - 10,000
Time Duration	T	3 - 20 hours
Time Resolution	δt	0.25 hour
Zipf Slope	α	0.8 [17], [18]
Cache Capacity	C	100, 300, 500, Infinite Cache
Object Expiry Rate	μ	0.5, 1 per hour
Request Arrival Rate	β	200 - 500 per hour

In Section VII, we compare the instantaneous hit ratio of Perfect-LFU, FB-FIFO, LRU, and FIFO. Note that no analytical results are provided for FB-FIFO or Perfect-LFU.

VII. EVALUATION

An event-driven simulator was developed using C++ to verify the analytical results for Infinite Cache, LRU, and FIFO. Also, FB-FIFO and Perfect-LFU are implemented in the simulations to allow a comparison to LRU and FIFO. The analytical instantaneous hit ratios are compared to the mean of the simulated instantaneous hit ratios generated using 1000 workload profiles that statistically conform with the proposed mathematical models in Section III.

The instantaneous hit ratio $H(t)$ is calculated over a duration of 3–20 h in $\delta t = 0.25$-h intervals as a function of cache capacity, object expiry rate, and request rate. Note that the analysis estimates the instantaneous hit ratio at a specific time t, while the simulator smooths the instantaneous hit ratio over a small time interval δt. Using small time intervals between $H(t)$ calculations eliminates the effect of this difference.

In the simulations, it is assumed that the object size follows a lognormal distribution [18] as listed in Table II. In the analysis and simulations, the ratio between the cache size and the average object size (i.e., the cache capacity) varies from 100 to 500. Though the analysis estimates the instantaneous hit ratio assuming objects with different expiry rates, for simplicity, the instantaneous hit ratio is predicted assuming that all the objects have the same expiry rate, $\mu_i = \mu$. The request rates that vary from 200 to 500 requests/h are chosen such that the evolution of the instantaneous hit ratio can be captured in the transient state. The analysis and simulator adopt the settings shown in Table III. Note that the values in Tables II and III are meant to illustrate the concepts in this paper rather than representing empirical workload values.

In Section VII-A, we evaluate the proposed replacement policies assuming a fixed number of objects. Section VII-B illustrates the evaluation of replacement policies assuming that the Web server generates new popular objects periodically. In the following, all plots show the hit ratio as a function of time. The markers in these plots represent the simulation results, while analysis results that correspond to each simulated scenario are shown with solid lines.

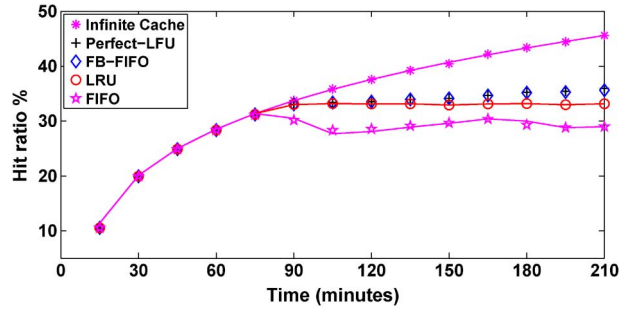


Fig. 3. Instantaneous hit ratio ($C = 500$, $\mu = 0$, $\beta = 500$).

A. Evaluation of Replacement Policies Assuming Fixed Number of Objects

In this section, the instantaneous hit ratio $H(t)$ is plotted within interval $[0, T]$ starting from an empty cache at time $t = 0$, where $T = 210$ min. Within this interval, it is assumed that the number of objects and the Zipf slope are fixed ($M = 10\,000$, $\alpha = 0.8$).

Fig. 3 shows a close match between analysis and simulation results for Infinite Cache, LRU, and FIFO assuming a cache capacity of $C = 500$, cached objects that do not expire (i.e., $\mu = 0$), and request rate of $\beta = 500$. This plot also indicates that the hit ratio of the different replacement policies evolve differently as a function of time. In Fig. 3, Infinite Cache, LRU, FIFO, FB-FIFO, and Perfect-LFU have the same hit ratio until $t_{\text{full}} = 75$ min, where t_{full} denotes the time when the cache becomes full. After t_{full}, the replacement policy starts replacing some cached objects, causing the hit ratio to increase in case of LRU, FB-FIFO, and Perfect-LFU, or decrease in case of FIFO.

Unlike the other replacement policies, Fig. 3 shows that FIFO reaches a maximum hit ratio of 31.4% during the transient period. Afterwards, FIFO decreases again and keeps fluctuating and it reaches the steady-state hit ratio of 29.5% after 210 min. Unlike the other replacement policies, FIFO does not exploit the popularity feature of objects. Thus, it is likely that some of the most popular objects that are cached within the first 30 min will reside at low caching states at t_{full}. Note that the hit ratio increases rapidly within the first 30 min when a big percentage of the most popular objects are cached. Hence, FIFO starts replacing a portion of the most popular objects with less popular ones after t_{full}, which causes the hit ratio to drop. Afterwards, the hit ratio will increase again as FIFO starts caching the most popular objects, which could be replaced after t_{full}, again. An oscillation in hit ratio continues to be observed as the most popular objects work their way through the cache and get ejected. Eventually, steady state is reached once FIFO is able to maintain a fixed percentage of the most popular objects in the cache.

Fig. 3 shows that the proposed FB-FIFO outperforms both LRU and FIFO, while FB-FIFO achieves the same hit ratio as Perfect-LFU during the evaluation interval $[0, T]$. Note that Perfect-LFU and FB-FIFO start outperforming LRU at time $t = 120$ min. Before t_{full}, the three replacement policies maintain the same objects in the cache. However, unlike Perfect-LFU

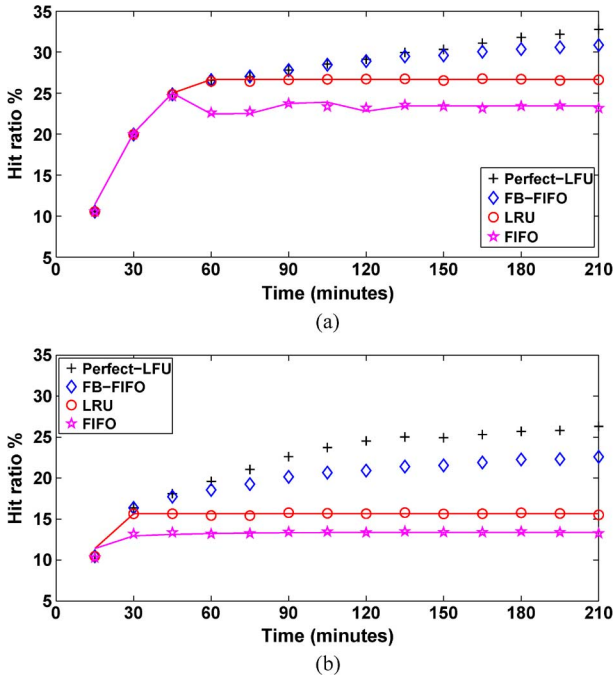


Fig. 4. Instantaneous hit ratio ($\mu = 0, \beta = 500$). (a) $C = 300$. (b) $C = 100$.

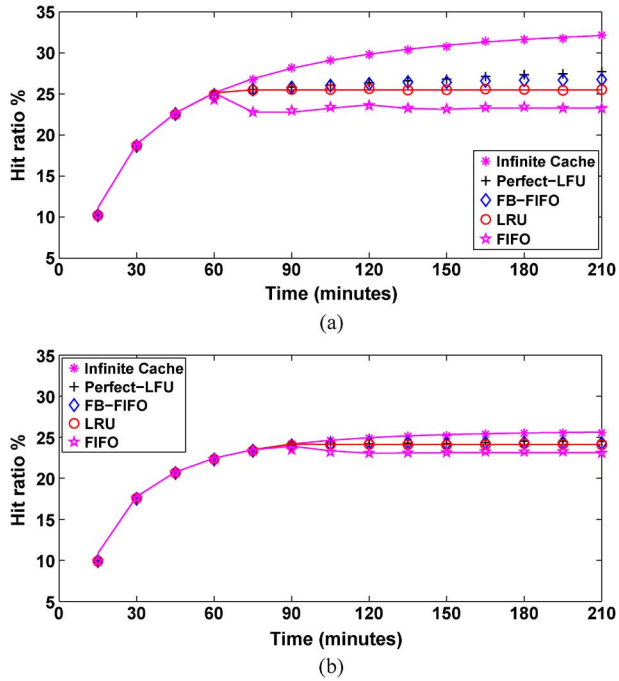


Fig. 5. Instantaneous hit ratio ($C = 300, \beta = 500$). (a) $\mu = 0.5$; (b) $\mu = 1$.

TABLE IV
INSTANTANEOUS HIT RATIO FOR FIFO, WHERE $\mu = 0, \beta = 500$

Time (minutes)	45	75	105	135	165	195
$C = 500$	25	31.4	27.7	28.9	30.4	28.8
$C = 300$	25	22.5	23.9	23.4	23.4	23.4

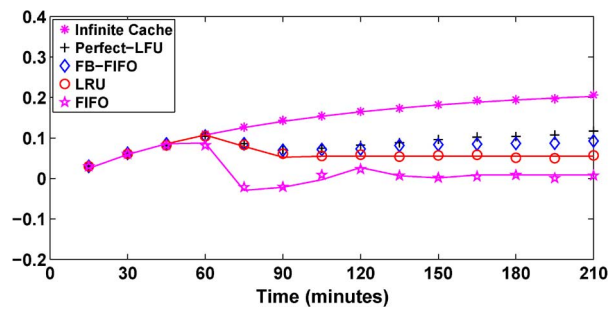


Fig. 6. Relative change in the instantaneous hit ratio when μ increases from 0.5 to 1 ($C = 300, \beta = 500$).

and FB-FIFO, LRU suffers from one-time requested objects, which constrains the increase in the percentage of the cached popular objects after t_{full}.

The remainder of this section discusses the impact of cache capacity, object expiry, and request rate on the considered replacement policies.

1) Impact of Cache Capacity: Fig. 4 shows the impact of cache capacity C on $H(t)$ for the considered replacement policies. Fig. 4 shows that as C decreases, the cache fills up quickly, and thus Perfect-LFU, FB-FIFO, and LRU start outperforming FIFO sooner.

Fig. 4(a) shows that FIFO fluctuates more rapidly and reaches steady state faster when C decreases to 300. Table IV helps illustrating the behavior for FIFO. Fig. 4(b) shows that when C decreases to 100, FIFO fluctuation diminishes. When C decreases, the percentage of the most popular objects that are cached when the cache fills up at t_{full} decreases. Therefore, after t_{full}, the probability that the portion of the popular objects that reside at low caching states will be replaced by objects with similar popularity increases. Consequently, the transient behavior of FIFO is smoothed faster as C decreases.

Fig. 4 shows that as C decreases, FB-FIFO outperforms both LRU and FIFO more rapidly. Moreover, Perfect-LFU outperforms FB-FIFO after a time t_{p} that decreases with C. For example, $t_{\text{p}} = 120$ for $C = 300$ and $t_{\text{p}} = 30$ for $C = 100$. Note that if any replacement policy keeps similar percentage of the popular objects as Perfect-LFU, that replacement policy achieves a hit ratio similar to Perfect-LFU. Therefore, after t_{full}, the difference between Perfect-LFU and FB-FIFO is small. This difference increases over time as Perfect-LFU identifies more of the popular objects. As C increases, it takes longer for Perfect-LFU to distinguish popular objects with request rates that are too low to be identified by FB-FIFO. Thus, FB-FIFO tracks Perfect-LFU for a longer time.

2) Impact of Object Expiry Rate: Fig. 5 shows the impact of the object expiry rate μ on $H(t)$. Fig. 5 shows that as μ increases, $H(t)$ decreases since the rate at which the object is ejected from the cache increases. Moreover, $H(t)$ reaches steady state faster as μ increases. Also, note that the transient behavior of FIFO diminishes as μ increases. As discussed in Section VII-A.1, this is due to the decrease in the percentage of the most popular objects that are cached when the cache fills up.

Let $H(t,1)$ and $H(t,2)$ denote the instantaneous hit ratio in Fig. 5(a) and (b), respectively. Fig. 6 shows the relative change in the instantaneous hit ratio (i.e., $(H(t,1) - H(t,2))/H(t,1)$).

In Fig. 6, the results show that Infinite Cache is the most sensitive to μ, followed by Perfect-LFU, FB-FIFO, LRU, and FIFO. As μ increases, the popular objects are ejected due to object

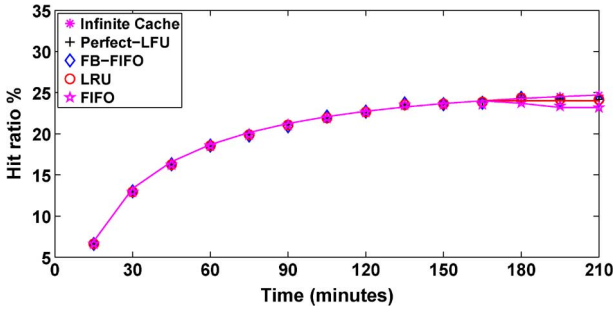


Fig. 7. Instantaneous hit ratio ($C = 300, \mu = 0.5, \beta = 250$).

expiry, rather than requests for uncached objects. Thus, the replacement policies that minimize the ejection of popular objects are the more sensitive to μ.

3) Impact of Request Rate: Figs. 7 and 5(a) show that, in the transient state, the $H(t)$ for all the replacement policies decreases when β decreases to 250 requests/h, while $\mu = 0.5$. This is consistent with the results for the steady-state hit ratio of Infinite Cache generated in [10]. Moreover, comparing Figs. 7 and 5(a) shows that the relative increase in β with respect to μ improves FB-FIFO more rapidly than LRU and FIFO, which improves the least.

Comparing Figs. 7 and 5(b) shows that doubling β and μ results in the same steady-state hit ratio. However, the steady state is reached faster. For example, Perfect-LFU, FB-FIFO, and LRU reach steady state at time $t = 90$ when $\beta = 500$, while the time required to reach steady state is almost doubled when $\beta = 250$.

B. Evaluation of Replacement Policies as the Number of Objects Increases

In real networks, the number of objects stored on the Web servers is growing continuously [41]. New objects with different popularity may be generated causing changes in the hit ratio that the cache can achieve over time.

Fig. 8 shows the instantaneous hit ratio for the replacement policies over 20 hours, assuming that the initial number of objects at time $t = 0$ is $M(t_0) = 5000$. Every $\Delta T = 5$ h, 50 new objects are generated. It is assumed that the 50 new generated objects become the most popular objects within the next ΔT. For example, the most popular object within the interval [0, 5] h becomes the 50th most popular object within the interval (5,10] h, and the 100th most popular object within the interval (10, 15] h. Similarly, the most popular object that is generated at $t = 5$ h becomes the 50th most popular object within the interval (10, 15] h, and so on. Therefore, the initial popularity of the cached objects degrades rapidly with time. Fig. 8 shows that the analysis accurately estimates the instantaneous hit ratio for LRU and FIFO. Also, Fig. 8 illustrates the case when the cache does not reach steady state. Hence, the only way to estimate the hit ratio experienced by Web users is to calculate the instantaneous hit ratio.

Fig. 8(a) shows that for $C = 100$, FB-FIFO adapts faster than the other replacement policies when new objects are generated at time $t = 5$ h. Assume that the robustness of the replacement policy is denoted by the instantaneous hit ratio achieved shortly after $M(t)$ changes (for example, $H(t = 5.5)$ after

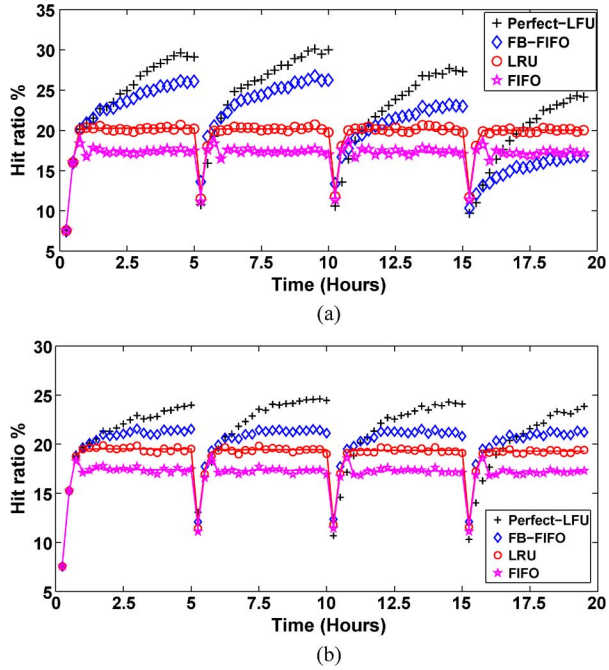


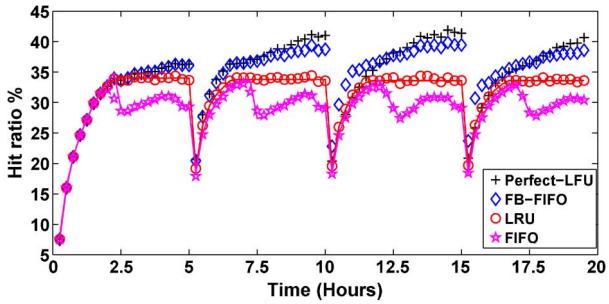
Fig. 8. Instantaneous hit ratio ($C = 100, \beta = 200$). (a) $\mu = 0$. (b) $\mu = 0.5$.

$M(t)$ changes at $t = 5$). Fig. 8(a) shows that FB-FIFO is the most robust replacement policy after $t = 5$ h. After $t = 10, 15$ h, LRU becomes the most robust replacement policy followed by FIFO, FB-FIFO, and then Perfect-LFU. At $C = 100$, the popularity of the cached objects decreases greatly for Perfect-LFU and FB-FIFO every time 50 new popular objects are generated. Hence, Perfect-LFU and FB-FIFO degrade greatly over time and they take longer to eject past popular objects. After $t = 15$ h, it might be useful for FB-FIFO to reset the counter to zero, or decrease its counter value on every cache miss, which may help remove old popular objects faster, as discussed in Section VI. Also, as discussed in Section VI, as μ increases FB-FIFO becomes more robust as shown in Fig. 8(b).

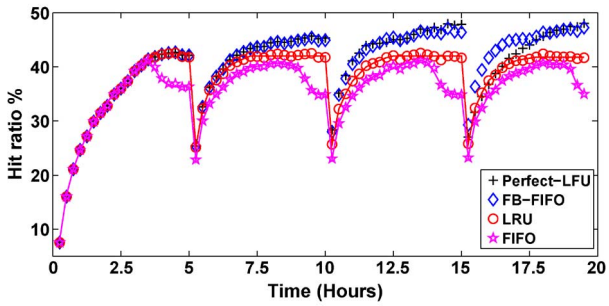
Note that the ratio between the number of new popular objects and the cache capacity plays a key role in the relative robustness of the replacement policies. Fig. 9(a) shows that as C increases to 300, after $t = 10, 15$ h, FB-FIFO becomes the most robust replacement policy followed by LRU, Perfect-LFU, and FIFO. The relative robustness of FB-FIFO increases each time new objects are generated. As discussed in Section VI, FB-FIFO caches the current popular objects in S_p faster as C increases, regardless of the past request rate of these objects. Fig. 9(b) shows that as C increases to 500, FB-FIFO matches Perfect-LFU for a longer time, as discussed in Section VII-A.1. Furthermore, Fig. 9(a) and (b) shows that after $t = 15$ h, FB-FIFO outperforms Perfect-LFU for a longer duration when C increases from 300 to 500. As discussed in Section VI, Perfect-LFU is affected more than FB-FIFO when more cached objects accumulate many requests over time. In this case, Perfect-LFU takes longer than FB-FIFO to identify the new popular objects and get rid of the old ones.

VIII. Conclusion

In this paper, a novel Markov chain analysis for estimating the instantaneous hit ratio of Infinite Cache, LRU, and FIFO was in-



(a)



(b)

Fig. 9. Instantaneous hit ratio ($\mu = 0, \beta = 200$). (a) $C = 300$. (b) $C = 500$.

troduced. Moreover, a new replacement policy (FB-FIFO) was developed based on the insights gained from the proposed analytical model. The results showed how the instantaneous hit ratio of the considered replacement policies evolves with time starting from an empty cache.

Assuming a fixed number of objects stored at the Web server, the results show that the hit ratio reaches steady state within a period that depends on cache capacity, object expiry rate, and request rate. In this case, results suggested that FB-FIFO outperforms LRU and FIFO in the steady state, especially for small cache capacities. On the other hand, these results change when the Web server generates new popular objects periodically. Therefore, evaluating the hit ratio of the replacement policies in the steady state assuming a fixed access pattern (i.e., fixed number of objects, objects popularity, request rate, etc.) can be misleading. Furthermore, the results show that when the Web server generates new popular objects periodically, the Web users may experience different hit ratios when they initiate requests at different times. In this case, the hit ratio experienced by these Web users can only be quantified by calculating the instantaneous hit ratio.

APPENDIX

This appendix discusses the convergence of the proposed iterative method for calculating the Markov chain probability vector $\mathbf{V}^z(t) = [P^z_{1,0}(t), P^z_{2,0}(t), \ldots, P^z_{M,0}(t)]$ from the perspective of contraction mapping theorem (CMT). According to the CMT, the sequence $\mathbf{V}^z(t)$, where $z = 0, 1, 2, \ldots$, converges to a unique fixed point $\mathbf{V}^*(t) = \lim_{z\to\infty} \mathbf{f}(\mathbf{V}^z(t))$, where $\mathbf{V}^z(t) = \mathbf{f}(\mathbf{V}^{z-1}(t)) = \mathbf{f}(\mathbf{f}(\mathbf{V}^{z-2}(t))) = \ldots$, if the Jacobian norm of $\mathbf{f}(\mathbf{V}(t))$ is less than one (i.e., $\|\mathbf{f}'(\mathbf{V}(t))\| < 1$) [35]. Markov chain balance equations can be used to show convergence for a scalar function under the following assumptions.

(A.1) The cache is in steady state (i.e., $t \to \infty$). Thus, $P^z_{i,0} = \lim_{t\to\infty} P^z_{i,0}(t) \forall i \in [1, M]$.

(A.2) The objects are requested with equal probability (i.e., $\alpha = 0$). Thus, $P^z_0 = P^z_{i,0} \forall i \in [1, M]$.

Also, in order to simplify the use of the balance equations, we assume the following.

(A.3) Objects do not expire (i.e., $\mu = 0$).

In the CMT [35], [36], $f : (0, 1] \to (0, 1]$ is a contraction mapping and converges to a unique fixed point, P^*_0, starting from any initial point, $P^0_0 \in (0, 1]$, if we have the following.

(C.1) $f : (0, 1] \to (0, 1]$ is a continuous function.

(C.2) $|f'(P_0)| < 1 \forall P_0 \in (0, 1]$. Also, (C.2) can be generalized to the following:

(C.2.1) $|g'(P_0)| < 1 \forall P_0 \in (0, 1]$, $g(P_0) = f(f(P_0))$.

For FIFO, according to (11) and using (A.2), we have

$$\epsilon^z_{i,j} = (\beta - \lambda) P_0^{z-1}. \tag{13}$$

Using (13) and (A.3) to solve the balance equations and calculate the limiting probability P^z_0, we have:

$$P^z_0 = \frac{(M-1)P_0^{z-1}}{(M-1)P_0^{z-1} + C} = \frac{\rho P_0^{z-1}}{\rho P_0^{z-1} + 1}$$

where $\rho \approx M/C > 1$, for large M, or equivalently

$$f(P_0) = \frac{\rho P_0}{\rho P_0 + 1}. \tag{14}$$

From (14), $f(P_0)$ satisfies (C.1). Also

$$f'(P_0) = \frac{\rho}{[\rho P_0 + 1]^2} \tag{15}$$

$$f'\left(f(P_0)\right) = \frac{\rho^2}{\left[\rho^2 P_0 + \rho P_0 + 1\right]^2}. \tag{16}$$

Note that the analysis in Section V-A calculates P^1_0 assuming that all objects are not in cache (i.e., $P^0_0 = 1$). Hence, the analysis initializes $P_0 = 1$ then P_0 decreases iteratively until reaching the fixed point $P^*_0 = (\rho - 1)/\rho$, according to (14). Therefore, we are only concerned that the maximum slope of $f(P_0)$, which occurs at the fixed point P^*_0, is less than 1 (i.e., $|f'(P^*_0)| < 1$, or $|f'(f(P^*_0))| < 1$). From (15), $f(P_0)$ satisfies (C.2) $\forall P_0 \in [P^*_0, 1]$ even for very small ρ (e.g., for $\rho = 2$, $f'(P^*_0) = f'(0.5) = 0.5$). Also, (16) shows that the analysis converges to $P^*_0 = (\rho - 1)/\rho$ when ρ is small even when P_0 is initialized by a small value that is bigger than zero (e.g., for $\rho = 10$ and $P_0 = 0.1$, $f'(f(0.1)) = 0.694 < 1$).

Similarly for LRU, according to (10) and using (A.2), we have

$$\epsilon^z_j = \lambda(M-1)x_j^{z-1} \tag{17}$$

where $x^z_j = \sum_{w=0}^{j} P^z_w$. Using (17) and the balance equations to calculate the limiting probabilities P^z_C, P^z_{C-1}, and P^z_0, we have

$$P^z_C = \frac{1}{(M-1)} \sum_{w=0}^{C-1} P^z_w = \frac{1}{M} \tag{18}$$

$$P^z_{C-1} = \frac{M-1}{(M-1)x_{C-1}^{z-1} + 1} P^z_C \tag{19}$$

$$P^z_0 = (M-1)x_1^{z-1} P^z_1. \tag{20}$$

For P_j^z, where $0 < j < C-1$

$$P_j^z = \frac{(M-1)x_{j+1}^{z-1}}{\left[(M-1)x_j^{z-1}+1\right]} P_{j+1}^z. \tag{21}$$

Therefore, from (18)–(21), we have

$$\begin{aligned} P_0^z &= (M-1)P_C^z \prod_{j=1}^{C-1} \frac{(M-1)x_j^{z-1}}{\left[(M-1)x_j^{z-1}+1\right]} \\ &\approx \prod_{j=1}^{C-1} \frac{Mx_j^{z-1}}{\left[Mx_j^{z-1}+1\right]} \qquad \text{for large } M \approx M-1 \\ &\approx \prod_{j=1}^{C-1} \frac{Mb_j^{z-1}P_0^{z-1}}{\left[Mb_j^{z-1}P_0^{z-1}+1\right]} \end{aligned} \tag{22}$$

where $b_j^{z-1} = x_j^{z-1}/P_0^{z-1}$ is constant. According to (20) and (21), the summation of the probabilities P_w^{z-1}, where $0 < w < C$, ultimately equals a scaled version of P_0^{z-1}, which is constant. From (22), $f(P_0)$ satisfies (C.1).

Note that the difference between the $\max(b_j^{z-1}) = b_{C-1}^{z-1}$ and $\min(b_j^{z-1}) = b_1^{z-1}$ is small, especially when P_0^{z-1} is large (or when M/C is large). Therefore, the product in (22) that uses the true values of b_j^{z-1} can be approximated by using only one value, $\eta \in [\min(b_j^{z-1}), \max(b_j^{z-1})]$, such that

$$P_0^z \approx \frac{\left[M\eta P_0^{z-1}\right]^{C-1}}{\left[M\eta P_0^{z-1}+1\right]^{C-1}}, \qquad \eta \text{ is constant.} \tag{23}$$

From (23), we have

$$f'(P_0) = \frac{C-1}{P_0} \frac{[M\eta P_0]^{C-1}}{[M\eta P_0+1]^C}. \tag{24}$$

For large P_0 (i.e., $P_0 \gg 1/M$), (24) can be rewritten as

$$f'(P_0) \approx \frac{C}{M\eta P_0^2}. \tag{25}$$

Assuming $\eta \approx (1+(1/P_0))/2$, $f(P_0)$ satisfies (C.2), where (25) guarantees that $|f'(P_0)| < 1$ even for small M/C at small P_0. For example, if $M/C = 10$ and $P_0 = 0.2$, then $f'(P_0) = 25/30 < 1$.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their detailed comments that helped considerably in improving the paper.

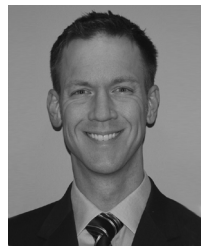
REFERENCES

[1] S. Nagaraj, *Web Caching and Its Applications*. Norwell, MA: Kluwer, 2004.
[2] H. Che, Z. Wang, and Y. Tung, "Analysis and design of hierarchical web caching systems," in *Proc. 20th Annu. IEEE INFOCOM*, Apr. 2001, vol. 3, pp. 1416–1424.
[3] F. Guo and Y. Solihin, "An analytical model for cache replacement policy performance," *Perform. Eval. Rev.*, vol. 34, pp. 228–239, Jun. 2006.
[4] J. Reineke, D. Grund, C. Berg, and R. Wilhelm, "Timing predictability of cache replacement policies," *Real-Time Syst.*, vol. 37, no. 2, pp. 99–122, Nov. 2007.
[5] J. Reineke and D. Grund, "Relative competitive analysis of cache replacement policies," *SIGPLAN Not.*, vol. 43, pp. 51–60, Jul. 2008.
[6] E. Gelenbe, "A unified approach to the evaluation of a class of replacement algorithms," *IEEE Trans. Comput.*, vol. C-22, no. 6, pp. 611–618, Jun. 1973.
[7] P. R. Jelenkovic, "Asymptotic approximation of the move-to-front search cost distribution and least-recently-used caching fault probabilities," *Ann. Appl. Probab.*, vol. 9, no. 2, pp. 430–464, 1999.
[8] P. Jelenkovic and A. Radovanovic, "Asymptotic insensitivity of least-recently-used caching to statistical dependency," in *Proc. 22nd Annu. IEEE INFOCOM*, Mar. 2003, vol. 1, pp. 438–447.
[9] P. R. Jelenkovic and M. S. Squillante, "Critical sizing of LRU caches for dependent requests," *J. Appl. Probab.*, vol. 43, no. 4, pp. 1013–1027, Aug. 2006.
[10] A. Wolman, M. Voelker, N. Sharma, N. Cardwell, A. Karlin, and H. M. Levy, "On the scale and performance of cooperative web proxy caching," in *Proc. 17th ACM Symp. Oper. Syst. Principles*, Dec. 1999, vol. 33, no. 5, pp. 16–31.
[11] P. Rodriguez, C. Spanner, and E. W. Biersack, "Analysis of web caching architectures: Hierarchical and distributed caching," *IEEE/ACM Trans. Netw.*, vol. 9, no. 4, pp. 404–418, Aug. 2001.
[12] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and zipf-like distributions: Evidence and implications," in *Proc. 18th Annu. IEEE INFOCOM*, Mar. 1999, vol. 1, pp. 126–134.
[13] V. S. Mookerjee and Y. Tan, "Analysis of a least recently used cache management policy for web browsers," *Oper. Res.*, vol. 50, no. 2, pp. 345–357, Aug. 2000.
[14] A. Dan and D. Towsley, "An approximate analysis of the LRU and FIFO buffer replacement schemes," *Perform. Eval. Rev.*, vol. 18, no. 1, pp. 143–152, May 1990.
[15] S. Bakiras, S. Bakiras, and T. Loukopoulos, "Combining replica placement and caching techniques in content distribution networks," *Comput. Commun.*, vol. 28, pp. 1062–1073, Mar. 2005.
[16] N. Laoutaris, G. Smaragdakis, A. Bestavros, I. Matta, and I. Stavrakakis, "Distributed selfish caching," *IEEE Trans. Parallel Distrib. Syst.*, vol. 18, no. 10, pp. 1361–1376, Oct. 2007.
[17] S. Jin and A. Bestavros, "GISMO: Generator of streaming media objects and workloads," *Perform. Eval. Rev.*, vol. 29, no. 3, pp. 2–10, 2001.
[18] M. Busari and C. Williamson, "On the sensitivity of web proxy cache performance to workload characteristics," in *Proc. 20th Annu. IEEE INFOCOM*, Apr. 2001, vol. 3, pp. 1225–1234.
[19] M. Zink, K. Suh, Y. Gu, and J. Kurose, "Watch global, cache local: YouTube network traffic at a campus network: Measurements and implications," in *Proc. ACM Multimedia Comput. Netw.*, 2008, vol. 6818, no. 681805, pp. 1–13.
[20] S. Ghandeharizadeh and S. Shayandeh, "An evaluation of two domical block replacement techniques for streaming media in wireless home networks," in *Proc. Multimedia Int. Symp.*, Dec. 2008, pp. 372–377.
[21] A. Abhari and M. Soraya, "Workload generation for YouTube," *Multimedia Tools App.*, vol. 46, no. 1, pp. 91–118, Jan. 2010.
[22] H. Gomaa, G. Messier, R. Davies, and C. Williamson, "Peer-assisted caching for scalable media streaming in wireless backhaul networks," in *Proc. IEEE Global Telecommun. Conf.*, Dec. 2010, pp. 1–5.
[23] H. Chen and Y. Xiao, "On-bound selection cache replacement policy for wireless data access," *IEEE Trans. Comput.*, vol. 56, no. 12, pp. 1597–1611, Dec. 2007.
[24] C.-Y. Chow, H. V. Leong, and A. Chan, "Peer-to-peer cooperative caching in mobile environments," in *Proc. Dist. Comput. Syst. Workshops, Int. Conf.*, Aug. 2004, vol. 4, pp. 528–533.
[25] C.-Y. Chow, H. V. Leong, and A. T. Chan, "GroCoca: Group-based peer-to-peer cooperative caching in mobile environment," *IEEE J. Sel. Areas Commun.*, vol. 25, no. 1, pp. 179–191, Jan. 2007.
[26] H. Gomaa, G. Messier, R. Davies, and C. Williamson, "Media caching support for mobile transit clients," in *Proc. IEEE Int. Conf. Wireless Mobile Comput. Netw. Commun.*, Marrackesh, Morocco, Nov. 2009, pp. 79–84.
[27] J. Zhao, P. Zhang, G. Cao, and C. R. Das, "Cooperative caching in wireless p2p networks: Design, implementation, and evaluation," *IEEE Trans. Parallel Distrib. Syst.*, vol. 21, no. 2, pp. 229–241, Feb. 2010.
[28] A. Dingle and T. Pártl, "Web cache coherence," *Comput. Netw. ISDN Syst.*, vol. 28, pp. 907–920, 1996.
[29] S. Podlipnig and L. Böszörmenyi, "A survey of web cache replacement strategies," *ACM Comput. Surv.*, vol. 35, no. 4, pp. 374–398, Dec. 2003.
[30] W. F. King, "Analysis of paging algorithms," in *Proc. IFIP Congr.*, 1971, pp. 697–701.
[31] A. Mahanti, D. Eager, and C. Williamson, "Temporal locality and its impact on web proxy cache performance," *Perform. Eval.*, vol. 42, no. 2–3, pp. 187–203, 2000.

[32] E. G. Coffman and P. Jelenkovic, "Performance of the move-to-front algorithm with Markov-modulated request sequences," *Oper. Res. Lett.*, vol. 25, no. 3, pp. 109–118, Oct. 1999.
[33] S. Ross, *Introduction to Probability Models*, 10th ed. New York: Academic, 2010.
[34] L. Kleinrock, *Queueing Systems*. New York: Wiley-Interscience, 1975.
[35] P. J. Olver, "Applied mathematics lecture notes: Nonlinear systems," 2012 [Online]. Available: http://www.math.umn.edu/olver/appl.html
[36] D. P. Mandic and V. S. Goh, *Complex Valued Nonlinear Adaptive Filters: Noncircularity, Widely Linear and Neural Models*. Hoboken, NJ: Wiley, 2009.
[37] A. S. Tanenbaum and A. S. Woodhull, *Operating Systems: Design and Implementation*. Upper Saddle River, NJ: Prentice-Hall, 1997.
[38] N. Megiddo and D. Modha, "Outperforming LRU with an adaptive replacement cache algorithm," *Computer*, vol. 37, no. 4, pp. 58–65, Apr. 2004.
[39] E. J. O'Neil, P. E. O'Neil, and G. Weikum, "The LRU-K page replacement algorithm for database disk buffering," in *Proc. ACM SIGMOD Conf.*, 1993, pp. 297–306.
[40] D. Lee, J. Choi, J. H. Kim, S. H. Noh, S. L. Min, Y. Cho, and C. S. Kim, "LRFU: A spectrum of policies that subsumes the least recently used and least frequently used policies," *IEEE Trans. Comput.*, vol. 50, no. 12, pp. 1352–1361, Dec. 2001.
[41] B. E. Brewington and G. Cybenko, "How dynamic is the web?," *Comput. Netw.*, vol. 33, no. 1–6, pp. 257–276, Jun. 2000.

Hazem Gomaa (S'12) received the B.Sc. degree in electrical and communication engineering from Alexandria University, Alexandria, Egypt, in 2004, and the M.Sc. degree in electrical and computer engineering from University of Calgary, Calgary, AB, Canada, in 2008, and is currently pursuing the Ph.D. degree in electrical and computer engineering at the University of Calgary.

His research interests include design and analysis of Web caching systems.



Geoffrey G. Messier (S'91–M'98) received the B.S. degrees in electrical engineering and computer science with great distinction from the University of Saskatchewan, Saskatoon, SK, Canada, in 1996, the M.Sc. degree in electrical engineering from the University of Calgary, Calgary, AB, Canada, in 1998, and the Ph.D. degree in electrical and computer engineering from the University of Alberta, Edmonton, AB, Canada, in 2004.

From 1998 to 2004, he was employed with the Nortel Networks CDMA Base Station Hardware Systems Design Group, Calgary, AB, Canada. At Nortel Networks, he was responsible for radio channel propagation measurements and simulating the physical layer performance of high-speed CDMA and multiple-antenna wireless systems. Currently, he is an Associate Professor with the Department of Electrical and Computer Engineering, University of Calgary . His research interests include data networks, physical-layer communications, and communications channel propagation measurements.

Carey Williamson (M'91) He holds a B.Sc. degree (Honours) from the University of Saskatchewan, Saskatoon, SK, Canada, in 1985, and the Ph.D. degree from Stanford University, Stanford, CA, in 1991, both in computer science.

He is a Professor with the Department of Computer Science, University of Calgary, Calgary, AB, Canada. His research interests include Internet protocols, wireless networks, network traffic measurement, network simulation, and Web performance.

Robert Davies (M'10) received the B.Sc. and M.Sc. degrees in electrical engineering from the University of Calgary, Calgary, AB, Canada, in 1987 and 1989, respectively, and the Ph.D. degree in electrical and computer engineering from the University of Alberta, Edmonton, AB, Canada, in 1999.

He was a Senior Researcher with TELUS Communications, Calgary, AB, Canada, from 1989 to 1997. He was the Senior Scientist with TRLabs, Calgary, AB, Canada, from 1997 to 2011. He is currently Principal Investigator for RFID and sensor technologies with SAIT Polytechnic Calgary, Calgary, AB, Canada. He has 10 patents and 60 publications and participates in numerous other activities related to communication systems. He has been an Adjunct Professor with the Department of Electrical and Computer Engineering, University of Calgary, since 2000.

Dr. Davies is also a licensed Master Electrician.