Computationally Efficient Dynamic Code Assignment Schemes With Call Admission Control (DCA-CAC) for OVSF-CDMA Systems

Jun-Seong Park, Lei Huang, and C.-C. Jay Kuo, *Fellow, IEEE*

***Abstract*—This paper investigates computationally efficient suboptimal dynamic code assignment (DCA) schemes with call admission control (CAC) for orthogonal variable spreading factor code-division multiple-access systems. We examine two different approaches. The first approach reduces the complexity of the DCA scheme by partitioning the total resource (either capacity or service class) into several mutually exclusive subsets and assigns each subset of resource to a group of users in proportion to the corresponding traffic load. The second approach reduces the complexity of the optimal CAC scheme with the Markov decision process over an infinite time horizon by a suboptimal CAC policy, which is designed by observing the behavior of system dynamics over only two consecutive stages upon the arrival of a call. It is demonstrated by numerical evaluation that the proposed schemes achieve an average data throughput close to that of the optimal DCA-CAC performance while demanding a much lower computational complexity in their design and implementation.**

***Index Terms*—Call admission control (CAC), code-division multiple access (CDMA), dynamic code assignment (DCA), Markov decision process (MDP), orthogonal variable spreading factor (OVSF).**

I. Introduction

ORTHOGONAL variable spreading factor (OVSF) codes are employed in wideband code-division multiple-access (CDMA) systems to support the different data rates of multimedia services [1], [2]. In OVSF-CDMA systems, a set of orthogonal codes with different lengths is generated using different spreading factors, where a higher data rate can be achieved by using a lower spreading factor. How to effectively and dynamically assign OVSF codes to users to maximize the throughput of the system or reduce the blocking probability of users is critical to the successful deployment of OVSF-CDMA systems. Usually, the code assignment scheme is integrated with a call admission control (CAC) policy to lead to a complete solution. However, to determine the optimal code assignment scheme with CAC is computationally expensive. Although it can serve as a performance benchmark, the optimal scheme is difficult to apply in practical applications. Computationally efficient suboptimal dynamic code assignment (DCA) schemes with CAC are the main focus of this paper.

In recent years, several construction methods for the channelization codes have been discussed in connection with the development of CDMA-based mobile communication systems [3]–[6]. Their research focused on the structured design of an OVSF code tree [3]–[5] and the implementation of an efficient code reassignment scheme [5]. In [6], Sekine *et al.* proposed an algorithm to reduce the chances of code blocking. A joint design of code assignment and error control coding was proposed by Li and Yum [7]. They tried to admit more users and achieve higher throughput by combining code allocation and error control coding in an interference-limited situation. A DCA scheme with a greedy CAC policy for OVSF-CDMA systems was proposed by Minn and Siu [8]. In their scheme, a call request is never rejected as long as the system can accommodate this new call request in addition to calls already in progress through code reassignment. To achieve the maximum average throughput of DCA, an optimal CAC policy was proposed by Park and Lee [9] using the Markov decision process (MDP). However, the optimal CAC policy demands a high computational complexity in its design and implementation due to a large number of states associated with the MDP model. With a linearly increasing capacity, the number of states and the resultant computational complexity and memory requirement exponentially increase. Although optimal fixed code assignment (FCA) schemes were proposed in [9] as an alternative to reduce the complexity, its performance notably degrades compared to that of optimal DCA schemes. Generally speaking, the lack of scalability and the difficulty of online implementation hinder the deployment of optimal solutions in practical OVSF-CDMA systems. Here, we investigate suboptimal DCA-CAC schemes with high throughput, which is close to the optimal solution, while the complexity of their implementation is still manageable in real-world OVSF-CDMA systems. In particular, we present two approaches to deal with this problem.

The first approach is motivated by the fact that the computational complexity associated with the optimal DCA is directly proportional to the size of the corresponding MDP that

Manuscript received March 12, 2005; revised May 6, 2006 and October 27, 2006. This work was supported in part by the Integrated Media Systems Center, which is a National Science Foundation Engineering Research Center, under Cooperative Agreement EEC-9529152. Any opinions, findings, and conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the National Science Foundation. The review of this paper was coordinated by Prof. E. Sourour.

J.-S. Park is with the FLYVO R&D Center, POSDATA Company Ltd., Bundang 463-775, Korea (e-mail: junspark@posdata.co.kr).

L. Huang is with the Department of Electrical Engineering and Computer Science, Loyola Marymount University, Los Angeles, CA 90045 USA (e-mail: lhuang@lmu.edu).

C.-C. J. Kuo is with the Department of Electrical Engineering and Integrated Media Systems Center, University of Southern California, Los Angeles, CA 90089 USA (e-mail: cckuo@sipi.usc.edu).

Digital Object Identifier 10.1109/TVT.2007.904506

exponentially grows with system capacity. Thus, we propose partitioning-based DCA schemes by dividing the total resource (capacity or service class) into several mutually exclusive subsets of resource, and the calls that belong to each group exclusively share the partial resource through the optimal DCA-CAC. By means of resource partitioning and partial resource sharing, we can significantly reduce the size of the MDP and the associated computational complexity, thus achieving good scalability of design and implementation. At the same time, the optimal CAC policy can be applied to each group to achieve good performance with little degradation in system throughput.

The second approach is based on the following two observations about the optimal CAC policy. First, the impact of a CAC decision made upon the arrival of a call on the resultant throughput rapidly decreases as time evolves. Second, the deliberate rejection of a call request in the presence of sufficient resource for better long-term throughput only occurs in a few states of the MDP. As a result, we develop a suboptimal CAC scheme that considers the throughput performance with respect to a few stages ahead (instead of the infinite time horizon). For example, when a new call request arrives, we may evaluate and compare the expected throughput based on the system dynamics of two consecutive stages under the *accept* decision and the *reject* decision and accordingly choose the decision that results in a higher throughput. Due to the above two observations, the long-term average throughput of the proposed suboptimal CAC policy is expected to be close to that of the optimal CAC policy. The proposed suboptimal CAC policy can be implemented and operated online as the CAC decision is instantaneously made upon the arrival of a call request.

The rest of this paper is organized as follows: In Section II, the system model is presented, the optimal DCA and FCA code assignment schemes are briefly reviewed, and their performance and complexity are analyzed. In Section III, two hybrid DCA schemes (i.e., capacity-partitioning and class-partitioning schemes) are proposed, and their complexity is analyzed. In Section IV, the design and implementation of a suboptimal CAC policy is discussed. The numerical results are presented in Section V, where we compare the throughput performance and complexity of the proposed hybrid DCA and suboptimal CAC schemes with that of the optimal FCA and DCA. Concluding remarks and future research directions are given in Section VI.

II. Optimal Code Assignment Schemes

A. System Model

Let us consider M classes of calls arriving according to independent Poisson processes. Calls of class-k requesting the transmission rate of $2^k R$ arrive at the rate of λ_k, $k = 0, 1, 2, \ldots, M-1$, where R is the basic data transmission rate (or the codes with the highest spreading factor in the OVSF code tree), and k indicates the service class that corresponds to the transmission rate of $2^k R$. The call duration for all classes is assumed to be exponentially distributed with mean $1/\mu$. The load of traffic that requests the transmission rate of $2^k R$ is defined as $\rho_k \triangleq \lambda_k/\mu$. The capacity of a given OVSF code tree $C_{\max}$ is defined in the number of the leaf node code with rate R.

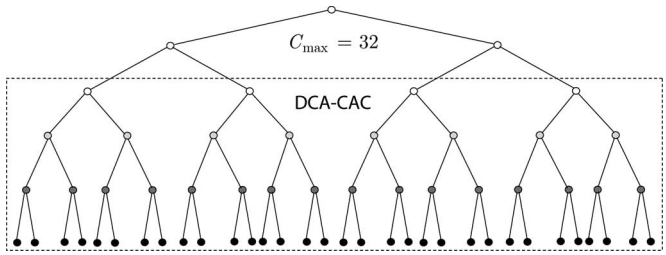


Fig. 1. DCA for $C_{\max} = 32$.

B. DCA With Optimal CAC Policy

The greedy call admission policy of DCA that was introduced in [8] may not achieve the maximal data throughput in the long run. For this reason, we considered a CAC that jointly works with the code reassignment scheme used in DCA so that the average throughput of the system is maximized. Fig. 1 illustrates the DCA for four service classes supporting the rates of $\{R, 2R, 4R, 8R\}$ and $C_{\max} = 32$. From this figure, we see that the total resource is completely shared by all classes of users in DCA.

We follow the same MDP formulation used in [9] to model the time dynamics of DCA with CAC. In the system model described in Section II-A, an action indicating accept or reject is introduced upon each arrival of call requests in the MDP model [10], [11].

The state space of the MDP, which is denoted by I_D, is the union of two sets, i.e., the set of states indicating that the epoch begins with an arrival and the set of states indicating that the epoch begins with a departure. Letting the state with the arrival of a j-class call and the state initiated by the departure be denoted by $(\boldsymbol{k}, j)$ and $(\boldsymbol{k})$, respectively, then the extended state space I_D has the form of

$$I_D(C_{\max}) = \left\{ (\boldsymbol{k}, j) \,\middle|\, \sum_{i=0}^{M-1} 2^i k_i \le C_{\max}, 0 \le j \le (M-1) \right\} \cup \left\{ (\boldsymbol{k}) \,\middle|\, \sum_{i=0}^{M-1} 2^i k_i \le C_{\max} \right\} \quad (1)$$

where $\boldsymbol{k} \triangleq (k_0, k_1, \ldots, k_{M-1})$, with k_l denoting the number of class-l calls in progress.

The total throughput of the system at each time t's epoch $\Omega_e(x(t), u(t))$ can be described by the combination of the MDP state $x(t)$ and the action $u(t)$ taken at the beginning of time t's epoch. To be more specific, $u(t)$ is defined as follows: For time t such that $x(t) = (\boldsymbol{k}, j) \in I_D(C_{\max})$, $u(t) = a$ if the decision made at the beginning of the epoch is *accept*, and $u(t) = r$ if the decision made at the beginning of the epoch is *reject*. For time t such that $x(t) = (\boldsymbol{k}) \in I_D(C_{\max})$ at the beginning of its epoch, no decision needs to be made, and thus, $u(t)$ is null.

When action $u(t)$ is chosen at time $t (0 \le t < \infty)$, the average throughput $\mathcal{T}_D(u)$ is the total data rate averaged over the

entire time horizon, i.e.,

$$\mathcal{T}_D(u) = \lim_{T\to\infty} \frac{1}{T} \int_0^T \Omega_e\left(x(t), u(t)\right) dt \tag{2}$$

where

$$\Omega_e\left(x(t), u(t)\right) = \begin{cases} [(\boldsymbol{k}+\boldsymbol{e}_j)\cdot(1,2,4,8)]\times R, & x(t)=(\boldsymbol{k},j), u(t)=a \\ [\boldsymbol{k}\cdot(1,2,4,8)]\times R, & x(t)=(\boldsymbol{k},j), u(t)=r \\ [\boldsymbol{k}\cdot(1,2,4,8)]\times R, & x(t)=(\boldsymbol{k}), u(t)=\text{Null} \end{cases} \tag{3}$$

where $\cdot$ denotes the dot product and $\boldsymbol{e}_j \triangleq (e_0, e_1, \ldots, e_{M-1})$ is a unit vector indexed by service class $j \in \{0, 1, \ldots, M-1\}$, for which $e_h = 1$ if $h = j$; otherwise, $e_h = 0$.

The optimal DCA-CAC scheme can be obtained through linear programming. The objective is to find an action $u(t)$ at time t that maximizes the average throughput of the system in the long run. The optimal DCA will be used as a building block in the proposed hybrid DCA algorithm.

We examine the computational complexity based on the size of the state space of the MDP model. The number of states in the corresponding MDP model exponentially increases with the capacity C $(= 2^i, i \geq 0)$ of a given OVSF code tree. Let S be the set of all service classes supported by system capacity C. Then, we have $S = \{s_1, s_2, \ldots, s_L\}$, where $s_i \in \{0, 1, \ldots, \log_2 C - 1\}$. With S and C of a given OVSF code tree, we can approximate the number of states $|I_D|$ of the corresponding MDP by evaluating the volume under the plane formed by $(C_{s_1}, C_{s_2}, \ldots, C_{s_L})$ in the L-dimensional Cartesian coordinate, where C_{s_i} is the total number of available OVSF codes in class s_i. Considering the action set incorporated into the MDP formulation, we can approximate the actual number of linear equations $\Lambda(S, C)$ required to solve the MDP using the linear programming solution method as

$$\begin{aligned} \Lambda(S,C) &= (\text{no. of states w/o arrival}) + (\text{no. of states w/ arrival}) \\ &= (\text{no. of states w/o arrival}) + (\text{no. of classes}) \\ &\quad \times (\text{no. of actions}) \times (\text{no. of states w/o arrival}) \\ &\sim \frac{1}{L} C_{s_1} C_{s_2} \ldots Cs_L + 2L \cdot \frac{1}{L} C_{s_1} C_{s_2} \ldots Cs_L. \end{aligned} \tag{4}$$

For example, $\Lambda(\{0,1,2,3\}, 16) \sim 2304$, and $\Lambda(\{0,2\}, 8) \sim 40$. The linear programming that corresponds to the MDP with the number of variables Λ requires $O(\Lambda)$ iterations and a total computational complexity of $O(\Lambda^3)$ when the simplex algorithm is used to find an optimal solution [12]. Therefore, it is reasonable to roughly view $\Lambda(S, C)^3$ as the computational complexity associated with DCA-CAC. From the implementation viewpoint, the number of states determines the size of the table that holds all state–action pairs.

C. FCA With Fixed Set Partitioning

From the above analysis, the complexity associated with the optimal DCA-CAC scheme exponentially increases as the depth of the OVSF code tree becomes larger. To mitigate its complexity, we proposed an alternative approach of FCA together with fixed set partitioning in [9]. This scheme partitions the whole set of OVSF codes into mutually exclusive groups of codes and uniquely assigns a group to each service class. The number of codes for each class is fixed in this way, and as a result, code reassignment can be avoided. To make its throughput comparable with that of DCA, we also determined the optimal partition of codes such that the average throughput of the system is maximized.

TABLE I
NUMERICAL EVALUATION OF $\Theta(M, C_{\max})$

	$M=1$	$M=2$	$M=3$	$M=4$
$C_{max}=16$	1	9	25	35
$C_{max}=32$	1	17	81	165
$C_{max}=64$	1	33	289	969
$C_{max}=128$	1	65	1089	6545
$C_{max}=256$	1	129	4225	47905

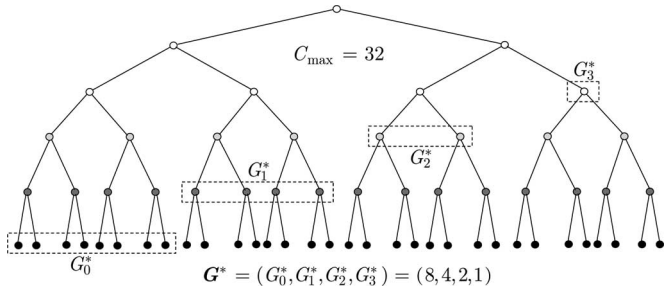


Fig. 2. FCA with fixed set partitioning, partition number $M = 4$, and maximum capacity $C_{\max} = 32$.

For the system model described in Section II-A, we use G_k to denote the total number of codes in a group that supports the data rate of class k. Each group of codes is uniquely assigned to one of the service classes. Then, there exists a finite number of code partitioning, which is represented by $\boldsymbol{G} \triangleq (G_0, G_1, \ldots, G_{M-1})$, that satisfies the maximum capacity constraint $G_0 + 2G_1 + 4G_2 + \cdots + 2^{M-1}G_{M-1} = C_{\max}$. The number of all possible partitions $\Theta(M, C_{\max})$ can be evaluated by

$$\Theta(M, C_{\max}) = \left| \left\{ (\boldsymbol{k}) \,\middle|\, \sum_{i=0}^{M-1} 2^i k_i = C_{\max} \right\} \right|. \tag{5}$$

Table I shows the numerical evaluation of $\Theta(M, C_{\max})$ with different M and $C_{\max}$. Note that $\Theta(M, C_{\max})$ is obviously much less than $\Lambda(S, C_{\max})$. Based on the statistics of incoming traffic, we find the optimal partition $\boldsymbol{G}^* = (G_0^*, G_1^*, \ldots, G_{M-1}^*)$ that maximizes the average throughput. Fig. 2 illustrates an FCA for four service classes $(M = 4)$ that support the rates of $\{R, 2R, 4R, 8R\}$ and $C_{\max} = 32$. From the figure, we see that FCA completely partitions the total resource among each class of users.

The average throughput $\mathcal{T}_F(\boldsymbol{G})$ of the entire system with a fixed set partition $\boldsymbol{G} = (G_0, G_1, \ldots, G_{M-1})$ is

$$\mathcal{T}_F(\boldsymbol{G}) = \sum_{k=0}^{M-1} (1 - P_k) \frac{\lambda_k}{\mu} (2^k R) \tag{6}$$

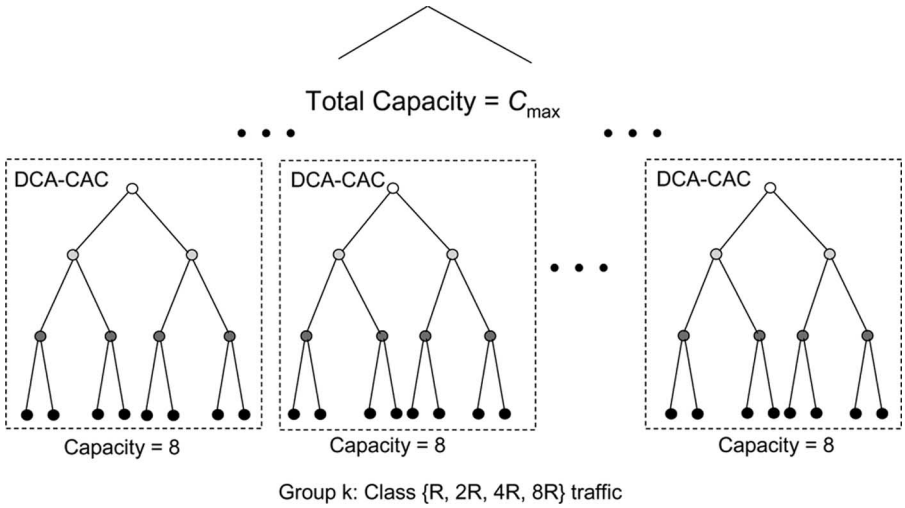


Fig. 3. Capacity-partitioning DCA scheme for the maximum capacity $C_{\max}$.

where $\lambda = \lambda_0 + \lambda_1 + \cdots + \lambda_{M-1}$, and P_k is the blocking probability of the code group corresponding to class-k and can be computed using Erlang's formula [13] as

$$P_k = \frac{\rho_k^{G_k}/G_k!}{\sum_{n=1}^{G_k} \rho_k^n/n!}. \tag{7}$$

Then, an exhaustive search can be used to find the optimal partition of the OVSF tree $\boldsymbol{G}^* = (G_0^*, G_1^*, \ldots, G_{M-1}^*)$, which maximizes the average throughput $\mathcal{T}_F$ as

$$\boldsymbol{G}^* = \arg\max_{\boldsymbol{G}} \mathcal{T}_F(\boldsymbol{G}).$$

Once the optimal partition $\boldsymbol{G}^*$ is determined, the number of codes in each code group is unchanged, and each code group serves only the corresponding class of calls independent of one another. The code reassignment can thus be avoided. In this way, we can design a minimal-complexity code assignment scheme and use it as a benchmark for the performance of the hybrid DCA schemes proposed in Section III.

In contrast with the high complexity of DCA-CAC design associated with the linear programming of MDP, much lower complexity is required by FCA. FCA only has to evaluate (6) for each possible partition $\boldsymbol{G}$, and therefore, it requires $O(\Theta)$ iterations, where Θ is the number of possible partitions, which is evaluated in (5), for the total capacity $C_{\max}$ and M groups.

III. Hybrid DCA Schemes

The optimal DCA-CAC scheme in Section II-B achieves maximum throughput at the cost of the highest complexity through complete sharing of resources. The optimal FCA scheme significantly reduces the complexity at the cost of a lower throughput through complete partitioning of the total resources. In this section, we propose hybrid schemes that employ both resource partitioning and optimal DCA-CAC to achieve good tradeoff between throughput and complexity through partial sharing of the total resources.

From (4), the complexity of the optimal DCA scheme is proportional to $\Lambda^3(S, C)$ in the MDP and the number of classes sharing the same resources. To reduce the complexity and improve the scalability, we propose two hybrid schemes, i.e., a capacity-partitioning scheme that reduces the number of states in MDP, and a class-partitioning scheme that reduces the number of classes sharing the same resources. In the second scheme, we further reduce the complexity by capacity partitioning based on the class partition.

A. Capacity-Partitioning Hybrid DCA Scheme

The proposed capacity-partitioning DCA scheme first partitions the set of the total resource into multiple mutually exclusive groups, thus reducing the size of the capacity of each partition. Second, call requests are statistically divided into multiple groups, each of which is exclusively served by the corresponding partition of the resource. As a result, the number of states in each capacity partition is significantly reduced. Finally, we apply the optimal DCA scheme described in Section II-B to each capacity partition to achieve the maximal average throughput for the partial resource shared by each partition. Fig. 3 illustrates the capacity-partitioning DCA scheme.

Given an OVSF code tree with capacity $C = C_{\max}$ and a set of available classes $S = \{0, 1, \ldots, M-1\}$, $M \le \log_2 C_{\max}$, we partition $C_{\max}$ into N mutually exclusive groups of codes by $C_{\max} = C_1 + C_2 + \cdots + C_N$. The call requests are also statistically divided into N groups in such a way that user group i, $1 \le i \le N$, takes the group arrival rate of $\lambda^{(i)}$, $\sum_{i=1}^{N} \lambda^{(i)} = \lambda$, where λ is the total arrival rate of call requests. Then, it follows that for each group of users, $\lambda^{(i)} = \sum_{j \in S} \lambda_j^{(i)}$, $1 \le i \le N$. The arrival rate $\lambda^{(i)}$ allocated to user group i is proportionally determined by C_i as

$$\lambda^{(i)} = \lambda \cdot \frac{C_i}{\sum_{k=1}^{N} C_k}, \qquad 1 \le i \le N. \tag{8}$$

As a result, we have

$$\lambda_j^{(i)} = \lambda^{(i)} \cdot \frac{\rho_j}{\sum_{k \in S} \rho_k}, \qquad j \in S, 1 \le i \le N. \tag{9}$$

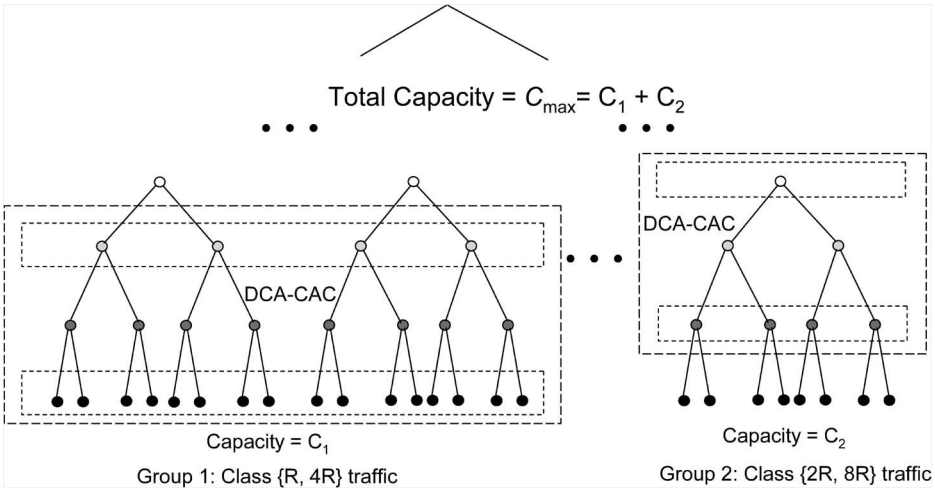


Fig. 4. Class-partitioning DCA scheme for the maximum capacity $C_{\max}$.

Let us use $\Gamma(S, C_i)$ to denote the maximal average throughput generated by the OVSF code tree with (S, C_i). Then

$$\Gamma(S, C_{\max}) \geq \Gamma(S, C_1) + \Gamma(S, C_2) + \cdots + \Gamma(S, C_N) \quad (10)$$

where $\sum_{i=1}^{N} C_i = C_{\max}$. Consequently, the computational complexity of the DCA with $(S, C_{\max})$ reduces to $\sum_{i=1}^{N} \Lambda(S, C_i)$ instead of $\Lambda(S, C_{\max})$ in the optimal DCA. It can easily be seen that the right-hand side of (10) approaches the optimal performance $\Gamma(S, C_{\max})$ associated with the entire OVSF code tree as N becomes smaller.

B. Class-Partitioning Hybrid DCA Scheme

The proposed class-partitioning DCA scheme first partitions the set of classes into multiple mutually exclusive groups, thus reducing the number of classes inside each partition. Second, the total resource capacity is divided among each class partition according to the traffic load. As a result, the number of states in each class partition is significantly reduced. Finally, we apply the optimal DCA scheme described in Section II-B to each class partition to achieve the maximal average throughput for the partial resource shared by each partition. Fig. 4 illustrates the class-partitioning DCA scheme.

Given an OVSF code tree with capacity $C = C_{\max}$ and a set of available classes $S = \{0, 1, \ldots, M-1\}$, $M \leq \log_2 C_{\max}$, we partition S into N mutually exclusive groups of classes by

$$S = S_1 \cup S_2 \cup \cdots \cup S_N, \; S_i \cap S_j = \phi \text{ for } i \neq j. \quad (11)$$

Then, a partial amount C_i of the total resource is uniquely assigned to group S_i in such a way that $\sum_{i=1}^{N} C_i = C_{\max}$. The amount C_i allocated to group S_i is proportionally determined by the total traffic load generated by users that belong to group S_i. That is

$$C_i = C_{\max} \cdot \frac{\sum_{j \in S_i} \rho_j}{\sum_{j=0}^{M-1} \rho_j} \quad (12)$$

where ρ_j is the traffic load of class-j calls.

The maximal average throughput generated by the OVSF code tree with (S_i, C_i) is denoted by $\Gamma(S_i, C_i)$. Then

$$\Gamma(S, C_{\max}) \geq \Gamma(S_1, C_1) + \Gamma(S_2, C_2) + \cdots + \Gamma(S_N, C_N) \quad (13)$$

where $\bigcup_{i=1}^{N} S_i = S$, and $\sum_{i=1}^{N} C_i = C_{\max}$. Consequently, the computational complexity of the DCA with $(S, C_{\max})$ reduces to $\sum_{i=1}^{N} \Lambda(S_i, C_i)$ instead of $\Lambda(S, C_{\max})$ in the optimal DCA. It can easily be seen that the right-hand side of (13) approaches the optimal performance $\Gamma(S, C_{\max})$ associated with the entire OVSF code tree as N becomes smaller.

IV. DCA With Suboptimal CAC Policy

As an alternative approach to reduce the complexity associated with the optimal DCA-CAC scheme, we investigate suboptimal CAC schemes for the complete sharing DCA in this section. Based on the detailed examination of system evolution of the MDP formulated for the DCA-CAC in Section II-B, we propose the suboptimal CAC policy that simplifies the evaluation of MDP reward rate upon every call arrival.

A. MDP Analysis of CAC for Code Assignment

We recall the MDP with the state space I_D formulated in Section II-B. For a given CAC policy $\pi = \{\nu_0, \nu_1, \ldots\}$ of our interest with $\nu_n(i) \in D(i) = \{a, r\}$, for all $i \in I_D$ and n, where n is the time index discretized by uniformization of the continuous-time MDP, the average throughput per stage J_π is

$$J_\pi(x_0) = \lim_{N \to \infty} \frac{1}{N} \mathbb{E} \left\{ \sum_{n=0}^{N-1} g\left(x_n, v_n(x_n)\right) \right\} \quad (14)$$

for an initial state $x_0 \in I_D$, where $g(i, u)$ is the expected throughput when the system is in state i, and action u is applied in (15), shown at the bottom of the next page. Note here that the throughput assigned to $x_n = (\boldsymbol{k}, j)$ depends on the decision made at stage n.

The optimal policy $\pi^* = \{\nu_0^*, \nu_1^*, \ldots\}$ maximizes (14) for any initial state x_0 and is obtained from the following optimization:

$$\max_{\nu_n(x_n)\in D(x_n)} \mathbb{E}\left\{g_n\left(x_n, \nu_n(x_n)\right) + J_{n+1}\left(f_n\left(x_n, \nu_n(x_n)\right)\right)\right\} \tag{16}$$

where J_{n+1} is the average throughput achieved from stage $n+1$ to ∞, and $f_n(\cdot)$ is a function of state transition from stage n to $n+1$ with control $\nu_n(x_n)$, i.e., $x_{n+1} = f_n(x_n, \nu_n(x_n))$. The simplest solution for the above optimization is derived by the linear programming technique, as shown in [9], [14], and [15].

The greedy policy $\pi^g = \{\nu_0^g, \nu_1^g, \ldots\}$ employed in DCA [8] takes a call request as long as it does not violate the capacity constraint $\sum_{i=0}^{M-1} 2^i k_i \le C_{\max}$ and can be described by

$$\nu_n^g(x_n) = \begin{cases} a, & x_n = (\boldsymbol{k}, j) \text{ and } \boldsymbol{k} + \boldsymbol{e}_j \in I_D \\ r, & x_n = (\boldsymbol{k}, j) \text{ and } \boldsymbol{k} + \boldsymbol{e}_j \notin I_D \\ \text{no action}, & x_n = (\boldsymbol{k}) \end{cases} \tag{17}$$

for all n.

Once any stationary CAC policy $\pi = \{\nu(x) | x \in I_D\}$ is determined, the average throughput of the entire system can be evaluated by Markovian analysis. We use $p(s,t;d)$ to denote the transition probability from state s to t with decision $d = \nu(s)$, $d \in D(s)$. Then, all possible transition probabilities of the corresponding Markov process are described as follows. For states $s = (\boldsymbol{k}, j) \in I_D$ with $\nu(s) = a$ for $j \in \{0, 1, \ldots, M-1\}$, we have

$$p(s,t;a) = \begin{cases} \lambda_\zeta \cdot \eta, & t = (\boldsymbol{k} + \boldsymbol{e}_j, \zeta) \\ k_\xi \mu \cdot \eta, & t = (\boldsymbol{k} + \boldsymbol{e}_j - \boldsymbol{e}_\xi) \\ 1 - \left(\sum_n \lambda_n + \mu \sum_{m \in B(\boldsymbol{k})} k_m\right) \cdot \eta, & t = (\boldsymbol{k} + \boldsymbol{e}_j) \\ 0, & \text{otherwise} \end{cases} \tag{18}$$

where $\boldsymbol{i} \in I_D, \zeta, \xi \in \{0, 1, \ldots, M-1\}$, and $B(\boldsymbol{k}) = \{m | \boldsymbol{k} + \boldsymbol{e}_j - \boldsymbol{e}_m \in I_D)\}$. For states $s = (\boldsymbol{k})$ taking no action and $s = (\boldsymbol{k}, j) \in I_D$ with $\nu(s) = r$ for $j \in \{0, 1, \ldots, M-1\}$, we have

$$p(s,t;r) = \begin{cases} \lambda_\zeta \cdot \eta, & t = (\boldsymbol{k}, \zeta) \\ k_\xi \mu \cdot \eta, & t = (\boldsymbol{k} - \boldsymbol{e}_\xi) \\ 1 - \left(\sum_n \lambda_n + \mu \sum_{m \in B(\boldsymbol{k})} k_m\right) \cdot \eta, & t = (\boldsymbol{k}) \\ 0, & \text{otherwise} \end{cases} \tag{19}$$

where $\boldsymbol{i} \in I_D, \zeta, \xi \in \{0, 1, \ldots, M-1\}$, and $B(\boldsymbol{k}) = \{m | \boldsymbol{k} - \boldsymbol{e}_m \in I_D\}$.

The average throughput per stage J_π under a stationary CAC policy π can be rewritten by

$$\begin{aligned} J_\pi &= \lim_{T\to\infty} \frac{1}{T} \sum_{s \in I_D} g\left(s, \nu(s)\right) \tau(T, s) \\ &= \sum_{s \in I_D} g(s) \varpi(s), \quad \text{with probability 1} \end{aligned} \tag{20}$$

where $\tau(T, s)$ is the total time that the system spends in state s up to time T, and $\varpi(s)$ is the steady-state probability of the Markov chain, i.e., the fraction of time spent in state s in the long run. Note that J_π becomes independent of an initial state x_0 under any stationary CAC policy. By solving the linear equations $\boldsymbol{\varpi} Q = \boldsymbol{0}$ with constraint $\sum_{s \in I_D} \varpi(s) = 1$, we can find the steady-state distribution $\boldsymbol{\varpi}$ and, consequently, the average throughput J_π as well.

B. One-Stage Lookahead Policy

An effective way to reduce the computation required by the MDP is to truncate the time horizon and make a decision at each stage based on the lookahead result in a small number of stages [14]. The simplest choice is the one-stage lookahead policy, which uses the control $\nu_n(x_n)$ at stage n and state x_n to achieve the following objective:

$$\max_{\nu_n(x_n)\in D(x_n)} \mathbb{E}\left\{g_n\left(x_n, \nu_n(x_n)\right) + \tilde{J}_{n+1}\left(f_n\left(x_n, \nu_n(x_n)\right)\right)\right\} \tag{21}$$

where $\tilde{J}_{n+1}$ is an approximation of the expected throughput J_{n+1} generated over the infinite time horizon starting from $x_{n+1} = f_n(x_n, \nu_n(x_n))$. Fig. 5 illustrates the time dynamics when the system is in stage n, and two different decisions (i.e., *accept* and *reject*) are made at stage n. Depending on the decision made at stage n, the system shows a different state evolution and, as a result, generates a different average throughput at stages n and $n+1$. J_{n+1} is the actual average throughput generated from stage $n+1$ to ∞ when an optimal decision is made at stage n.

Since it cannot be evaluated by observing only two stages n and $n+1$, we try to approximate J_{n+1} to $\tilde{J}_{n+1}$ in designing a suboptimal CAC policy. The accuracy of the approximation method significantly affects the resulting overall performance of the designed CAC policy. In our design, we choose

$$\begin{aligned} \tilde{J}_{n+1}\left(x_n, \nu_n(x_n)\right) = \mathbb{E}\left[g_{n+1}\left(x_{n+1}, \nu_{n+1}^g(x_{n+1})\right)\right] \\ \cdot \delta\left(x_n, \nu_n(x_n)\right) \end{aligned} \tag{22}$$

$$g\left(x_n, \nu_n(x_n)\right) = \begin{cases} \left[(\boldsymbol{k} + \boldsymbol{e}_j) \cdot (1, 2, \ldots, 2^{M-1})\right] \times R, & x_n = (\boldsymbol{k}, j), \nu_n(x_n) = a \\ \left[\boldsymbol{k} \cdot (1, 2, \ldots, 2^{M-1})\right] \times R, & x_n = (\boldsymbol{k}, j), \nu_n(x_n) = r \\ \left[\boldsymbol{k} \cdot (1, 2, \ldots, 2^{M-1})\right] \times R, & x_n = (\boldsymbol{k}), \nu_n(x_n) = \text{Null} \end{cases} \tag{15}$$

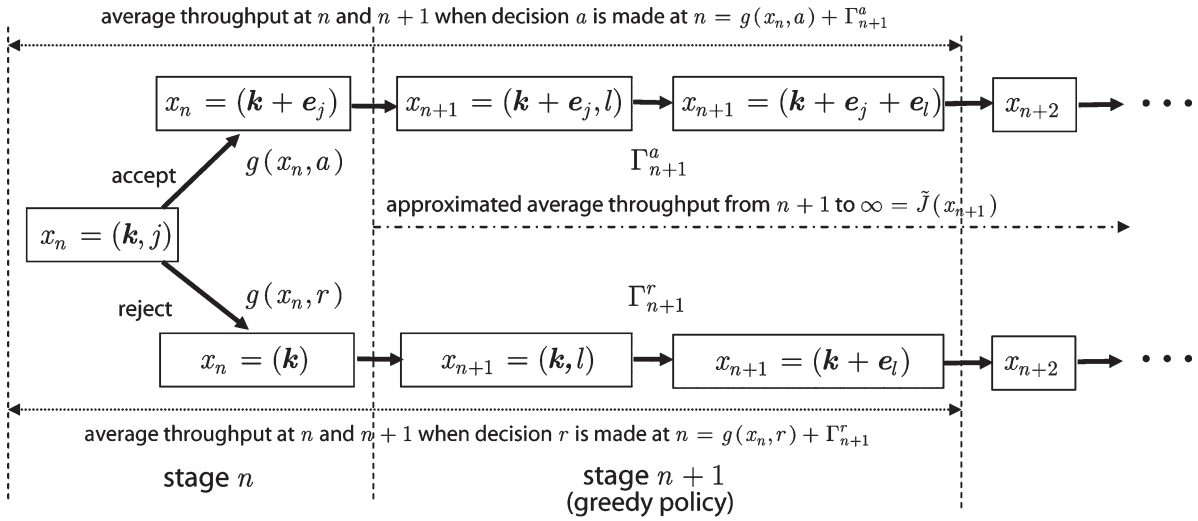


Fig. 5. System dynamics and throughput at stage n and $n+1$ depending on a different decision at stage n.

where $\delta(x_n, \nu_n(x_n))$ is a gain factor to reflect the actual effect of a decision at stage n propagating over the infinite time horizon. The next subsection describes in detail how to choose $\delta(\cdot)$.

For each state $x_n = (\boldsymbol{k}, j)$ with an arrival of class-j at stage n, the average throughput in the next stage $n+1$ upon a decision of *accept* at stage n is evaluated as

$$\begin{aligned}\Gamma^a_{n+1} &\triangleq \mathbb{E}\left[g_{n+1}\left(x_{n+1}, \nu^g_{n+1}(x_{n+1})\right) | x_n = (\boldsymbol{k}, j), \nu_n(x_n) = a\right] \\ &= \sum_\zeta \frac{\lambda_\zeta}{\eta} \cdot \left[(\boldsymbol{k} + \boldsymbol{e}_j + \boldsymbol{e}_\zeta) \cdot (1, 2, \ldots, 2^{M-1})\right] \times R \\ &\quad + \sum_\xi \frac{k_\xi \mu}{\eta} \cdot \left[(\boldsymbol{k} + \boldsymbol{e}_j - \boldsymbol{e}_\xi) \cdot (1, 2, \ldots, 2^{M-1})\right] \times R \\ &\quad + \left[1 - \sum_\zeta \frac{\lambda_\zeta}{\eta} - \sum_\xi \frac{k_\xi \mu}{\eta}\right] \\ &\quad \cdot \left[(\boldsymbol{k} + \boldsymbol{e}_j) \cdot (1, 2, \ldots, 2^{M-1})\right] \times R \end{aligned} \tag{23}$$

and the average throughput in the next stage $n+1$ upon a decision of *reject* at stage n is computed as

$$\begin{aligned}\Gamma^r_{n+1} &\triangleq \mathbb{E}\left[g_{n+1}\left(x_{n+1}, \nu^g_{n+1}(x_{n+1})\right) | x_n = (\boldsymbol{k}, j), \nu_n(x_n) = r\right] \\ &= \sum_\zeta \frac{\lambda_\zeta}{\eta} \cdot \left[(\boldsymbol{k} + \boldsymbol{e}_\zeta) \cdot (1, 2, \ldots, 2^{M-1})\right] \times R \\ &\quad + \sum_\xi \frac{k_\xi \mu}{\eta} \cdot \left[(\boldsymbol{k} - \boldsymbol{e}_\xi) \cdot (1, 2, \ldots, 2^{M-1})\right] \times R \\ &\quad + \left[1 - \sum_\zeta \frac{\lambda_\zeta}{\eta} - \sum_\xi \frac{k_\xi \mu}{\eta}\right] \cdot \left[(\boldsymbol{k}) \cdot (1, 2, \ldots, 2^{M-1})\right] \times R \end{aligned} \tag{24}$$

where $\eta \triangleq \sum \lambda_j + C_{\max}\mu$. Then, the case-by-case derivation of $(\Gamma^a_{n+1} - \Gamma^r_{n+1})$ is listed in (25), shown at the bottom of the page, where each case is defined by the following:

Case I: $(\boldsymbol{k} + \boldsymbol{e}_j) \in I_D$, $(\boldsymbol{k} + \boldsymbol{e}_j + \boldsymbol{e}_\zeta) \in I_D$, and $(\boldsymbol{k} + \boldsymbol{e}_j - \boldsymbol{e}_\xi) \in I_D$;

Case II: $(\boldsymbol{k} + \boldsymbol{e}_j) \in I_D$, $(\boldsymbol{k} + \boldsymbol{e}_j + \boldsymbol{e}_\zeta) \notin I_D$, and $(\boldsymbol{k} + \boldsymbol{e}_j - \boldsymbol{e}_\xi) \notin I_D$;

$$\Gamma^a_{n+1} - \Gamma^r_{n+1} = \begin{cases} \sum_\zeta \frac{\lambda_\zeta}{\eta} 2^j R + \sum_\xi \frac{k_\xi \mu}{\eta} 2^j R + \left[1 - \sum_\zeta \frac{\lambda_\zeta}{\eta} - \sum_\xi \frac{k_\xi \mu}{\eta}\right] 2^j R, & \text{Case I} \\ \sum_\zeta \frac{\lambda_\zeta}{\eta} (2^j - 2^\zeta) R + \sum_\xi \frac{k_\xi \mu}{\eta} (2^j + 2^\xi) R + \left[1 - \sum_\zeta \frac{\lambda_\zeta}{\eta} - \sum_\xi \frac{k_\xi \mu}{\eta}\right] 2^j R, & \text{Case II} \\ \sum_\zeta \frac{\lambda_\zeta}{\eta} 2^j R + \sum_\xi \frac{k_\xi \mu}{\eta} (2^j + 2^\xi) R + \left[1 - \sum_\zeta \frac{\lambda_\zeta}{\eta} - \sum_\xi \frac{k_\xi \mu}{\eta}\right] 2^j R, & \text{Case III} \\ \sum_\zeta \frac{\lambda_\zeta}{\eta} (2^j - 2^\zeta) R + \sum_\xi \frac{k_\xi \mu}{\eta} 2^j R + \left[1 - \sum_\zeta \frac{\lambda_\zeta}{\eta} - \sum_\xi \frac{k_\xi \mu}{\eta}\right] 2^j R, & \text{Case IV} \\ 0, & \text{Case V} \end{cases} \tag{25}$$

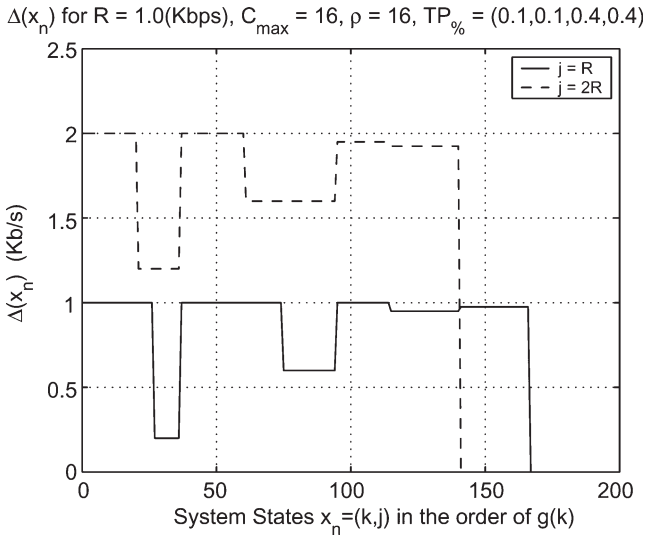


Fig. 6. Comparison of $\Delta((\boldsymbol{k},1))$ and $\Delta((\boldsymbol{k},2))$ for $C_{\max}=16$, $\rho=16$, and TP $=(0.1,0.1,0.4,0.4)$.

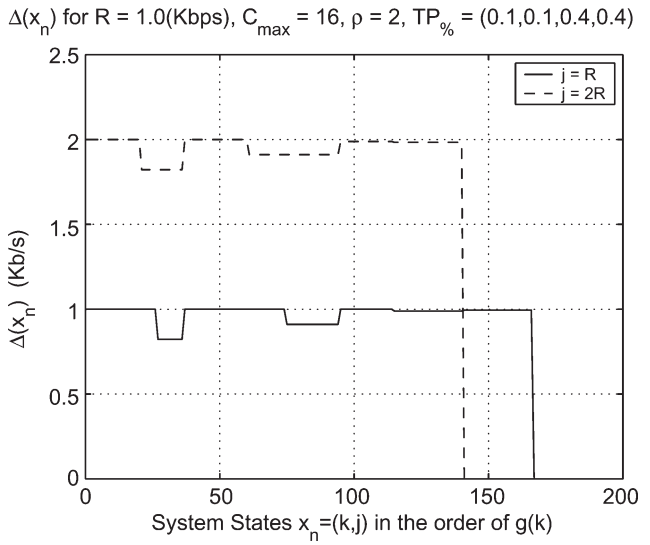


Fig. 7. Comparison of $\Delta((\boldsymbol{k},1))$ and $\Delta((\boldsymbol{k},2))$ for $C_{\max}=16$, $\rho=2$, and TP $=(0.1,0.1,0.4,0.4)$.

Case III: $(\boldsymbol{k}+\boldsymbol{e}_j)\in I_D$, $(\boldsymbol{k}+\boldsymbol{e}_j+\boldsymbol{e}_\zeta)\in I_D$, and $(\boldsymbol{k}+\boldsymbol{e}_j-\boldsymbol{e}_\xi)\notin I_D$;
Case IV: $(\boldsymbol{k}+\boldsymbol{e}_j)\in I_D$, $(\boldsymbol{k}+\boldsymbol{e}_j+\boldsymbol{e}_\zeta)\notin I_D$, and $(\boldsymbol{k}+\boldsymbol{e}_j-\boldsymbol{e}_\xi)\in I_D$;
Case V: $(\boldsymbol{k}+\boldsymbol{e}_j)\notin I_D$.

Based on the above derivation, the proposed suboptimal CAC policy $\tilde{\pi}$ is determined as follows. For state x_n, $\tilde{\nu}_n(x_n)=a$ if

$$\Delta_n(x_n)\stackrel{\Delta}{=}\left[g_n(x_n,\nu_n(x_n)=a)+\Gamma^a_{n+1}\cdot\delta\left(x_n,\nu_n(x_n)=a\right)\right]-\left[g_n(x_n,\nu_n(x_n)=r)+\Gamma^r_{n+1}\cdot\delta\left(x_n,\nu_n(x_n)=r\right)\right]>0.$$

Otherwise, $\tilde{\nu}_n(x_n)=r$. Note that it is a stationary policy that only depends on the states but not on the time stage.

C. Discussion on $\delta(\cdot)$

In summary, the proposed suboptimal CAC policy focuses on the average throughput at stages n and $n+1$ to make a decision at stage n. However, it is observed that, for a particular group of system states, $\Delta_n(x_n)>0$ becomes noticeably small compared to that of other states when $\delta(x_n,\nu_n(x_n))=1$. This property can be observed in Figs. 6 and 7, which show $\Delta_n(x_n)$ that was evaluated for an arrival of class R and $2R$ at $\rho=2,16$, respectively. The states are sorted on the order of $\tilde{g}(\boldsymbol{k})=\boldsymbol{k}\cdot(1,2,\ldots,2^{M-1})$, which represents how much resource is being used in state $\boldsymbol{k}$.

In Fig. 6 ($\rho=16$), we find a significant decrease of $\Delta_n(x_n)$ for some group of states, and the decrease becomes more noticeable for an arrival of low service class. Those states are likely to make a positive effect (increasing throughput) on the overall performance when the decision *reject* is made at stage n. Hence, for those states, the system is expected to enforce a call rejection to achieve the throughput gain over the infinite time horizon. Thus, we set $\delta(x_n,\nu_n(x_n))=1$ for $\nu_n(x_n)=a$ and $\delta(x_n,\nu_n(x_n))>1$ for $\nu_n(x_n)=r$ to obtain $\Delta_n(x_n)<0$. As we choose a larger $\delta(x_n,\nu_n(x_n))$ for $\nu_n(x_n)=r$, more groups of states that show the second (the third and so on) largest decrease of $\Delta_n(x_n)$ are selected for the enforcement of a call rejection. By adjusting the gain factor, we can design a better suboptimal CAC policy that further approaches the optimal one.

In Fig. 7 ($\rho=2$), we do not find any noticeable decrease of $\Delta_n(x_n)$, which means that any enforced call rejection is likely to make a negative effect (decreasing throughput) on the overall performance. Therefore, we merely follow a greedy policy when the system becomes lightly loaded. For the arrival of higher service classes $4R$ and $8R$, all states show almost the same value of $\Delta_n(x_n)$, so no rejection is enforced for those call arrivals. As ρ is larger, the decrease of $\Delta_n(x_n)$ for some group of states correspondingly becomes more noticeable, and $\Delta_n(x_n)$ may go down to zero. In that case, we can achieve $\Delta_n(x_n)<0$ even with $\delta(x_n,\nu_n(x_n))=1$, regardless of $\nu_n(x_n)$. Therefore, $\delta(\cdot)$ plays an important role in estimating the system behavior after stage $n+1$.

D. Complexity and Implementation Issues

The optimal CAC demands a high computational complexity in the design and implementation due to the large number of states associated with the corresponding MDP model. With the linearly increasing capacity, the number of states and the resulting computational complexity (offline) and required memory (online) exponentially increase. The online complexity of the MDP solution method is directly proportional to the size of the state space. Thus, the optimal CAC policy is virtually impossible to be implemented online for a large-scale OVSF code tree ($C_{\max}>32$). The offline complexity is associated with the size of storage that holds all the state–action pairs derived from any MDP solution method. However, the proposed suboptimal policy can be implemented online since the maximization with a given state x_n and the decision set $D=\{a,r\}$ is instantaneous. When a call arrives, the system instantaneously evaluates and compares $g(x_n,a)+\Gamma^a_{n+1}$ and $g(x_n,r)+\Gamma^r_{n+1}$ and makes a decision that generates more average throughput. We do not

TABLE II
DESCRIPTIVE COMPARISON OF COMPLEXITY AND PERFORMANCE AMONG THE PROPOSED SCHEMES WITH $C_{\max} = 256$ AND $M = 4$

	Optimal FCA ($\Theta(4, 256)$)	Hybrid DCA ($\Lambda(4, 64)$)	Suboptimal DCA	Optimal DCA ($\Lambda(4, 256)$)
Design type	off-line (search)	off-line (computation)	$\delta(\cdot)$	off-line (computation)
Operation type	classification	table look-up	on-line (computation)	table look-up
Computation for design	very light	moderate	heuristics	very heavy
Computation for operation	no	no	light	no
no. of states	47905 (search)	$\approx 500,000$ (MDP)	not applicable	$\approx 150,000,000$ (MDP)
Throughput Performance	good for heavy load	good for heavy load	close to optimum	optimal

need any offline heavy computation, as required by finding the optimal policy through the MDP formulation.

Although the complexity of each scheme has been discussed in the corresponding section, the comparison of all schemes in a table provides a better demonstration of the computational efficiency of our proposed hybrid and suboptimal schemes. It is, however, not easy to do a fair numerical comparison on the complexity of the proposed schemes since each of them has its unique feature of design and operation. Thus, we give a descriptive comparison of the proposed schemes in terms of the design and operation type, and the computational load and performance in Table II. Moreover, we provide the number of states associated with the Markov chain of each scheme to characterize the computational complexity. Note that the suboptimal DCA examines only two stages ahead, and it does not involve the full set of states in the computation.

For the design process, the optimal DCA and hybrid DCA schemes require complicated offline policy computation whose complexity relates to the number of states involved in the MDP model. If we consider an OVSF code tree of $C_{\max} = 256$, the optimal DCA required around 150 000 000 states involved in the generation of an optimal policy. On the contrary, the hybrid DCA with four capacity partitions only requires around 500 000 states involved in policy computation. The suboptimal DCA does not require any offline MDP computation since its policy decision is instantaneous for each call arrival. We only have to find a reasonable value of $\delta(\cdot)$ in the design process. The optimal FCA involves an offline search of the optimal partition whose complexity depends on the number of available partitions.

For the operation process, only the suboptimal DCA scheme involves light computation to instantaneously determine a two-stage optimal policy for each call arrival. The other schemes require a minimum complexity associated with table lookup (for optimal and hybrid DCA schemes) or classification of service classes (for the FCA scheme).

V. SIMULATION RESULTS

The numerical simulation adopts the following parameters.

- The total number of service classes is $M = 4$ with the service rates equal to R, $2R$, $4R$, and $8R$, where R is the base transmission rate normalized to $R = 1.0$.
- The call arrival process is Poisson with mean arrival rate of $\lambda = \lambda_0 + \lambda_1 + \lambda_2 + \lambda_3 = 1, 2, \ldots, 16$ calls/unit of time.
- The call duration is exponentially distributed with a mean value of $1/\mu = 1$ unit of time. The performance varies with $\rho_k = \lambda_k/\mu$, so setting a constant μ is reasonable.

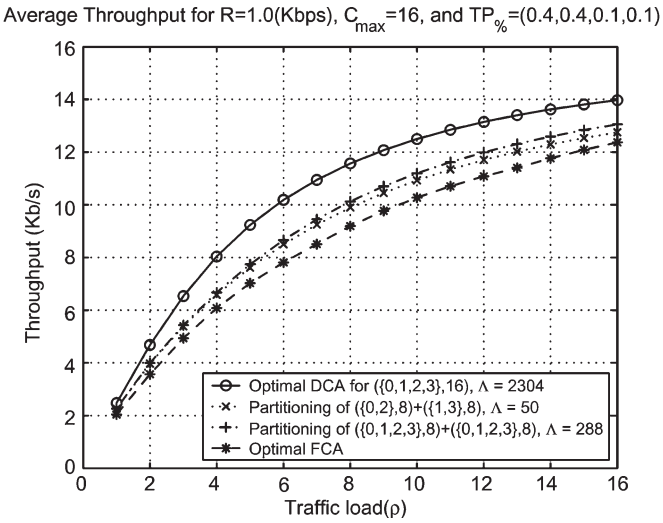


Fig. 8. Comparison of the average throughput when $C_{\max} = 16$ and TP $= (0.4, 0.4, 0.1, 0.1)$.

- The traffic load is $\rho = \lambda/\mu = 1, 2, \ldots, 16$.
- System capacity is the total number of the leaf codes $C_{\max} = 16$.
- The traffic profiles TP $\triangleq (\rho_0/\rho, \rho_1/\rho, \rho_2/\rho, \rho_3/\rho) = (0.4, 0.4, 0.1, 0.1)$ and (0.1, 0.1, 0.4, 0.4) are used.
- $\delta(x_n, \nu_n(x_n)) = 1$ for $\nu_n(x_n) = a$. For $\nu_n(x_n) = r$, it is chosen in such a way that only the group of states that shows the first largest decrease of $\Delta_n(x_n)$ is selected for call rejection.
- An ideal code-limited scenario is assumed. No interference-limited scenario is considered to define the system capacity.

Note that the assumption of an ideal code-limited scenario is made in the example to obtain a fixed channel capacity $C_{\max}$. However, the operation of the proposed schemes is not restricted to this assumption. In a realistic interference-limited scenario, $C_{\max}$ may vary with traffic conditions. In principle, our proposed schemes can be extended to such a scenario by adaptively performing code assignment with an updated $C_{\max}$ and traffic profiles. However, the detailed algorithm and the corresponding analysis go beyond the scope of this paper and can be an interesting topic for future research.

Figs. 8 and 9 show the performance comparison of the various code assignment schemes with different complexity. In Fig. 8, we see that the class-partitioning DCA scheme with the partition of $[(\{0, 2\}, 8), (\{1, 3\}, 8)]$ has an average throughput close to that of the optimum with 20 times less complexity. Also, the capacity-partitioning DCA scheme with

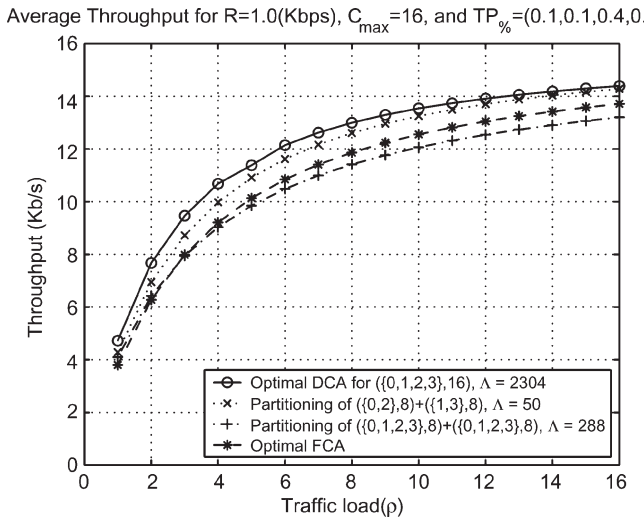


Fig. 9. Comparison of the average throughput when $C_{\max} = 16$ and TP $= (0.1, 0.1, 0.4, 0.4)$.

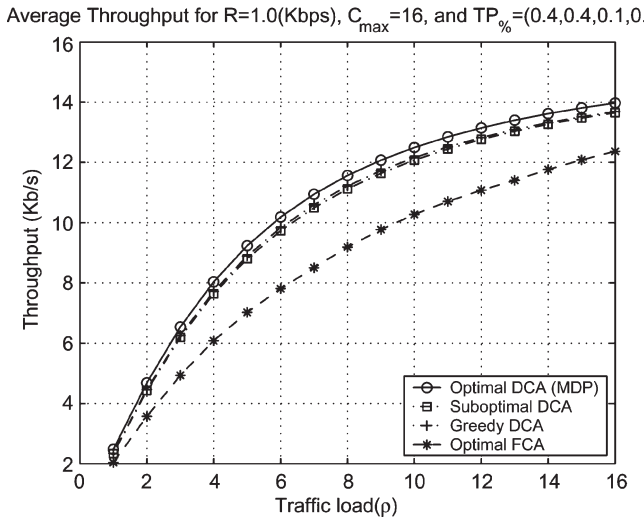


Fig. 10. Comparison of the average throughput when $C_{\max} = 16$ and TP $= (0.4, 0.4, 0.1, 0.1)$.

the partition of $[(\{0,1,2,3\},8),(\{0,1,2,3\},8)]$ generates an average throughput slightly higher than the previous one. We also observe that our partitioning-based schemes outperform the optimal FCA, which has the lowest complexity. Fig. 9 gives the performance comparison when the high-rate call requests become dominant. In this case, the class-partitioning DCA scheme achieves almost the same performance as that of the optimal DCA. The numerical results shown here demonstrate that our proposed hybrid DCA schemes achieve a good tradeoff between throughput performance and computational complexity.

Figs. 10 and 11 compare the average throughput of the optimal FCA, DCA with greedy CAC policy, DCA with the suboptimal CAC policy, and DCA with the optimal CAC policy for two different traffic profiles. As seen in Fig. 10, the greedy DCA shows almost the same performance as the suboptimal DCA, which means that the greedy DCA is a good alternative to the optimal DCA when low-rate call requests are dominant.

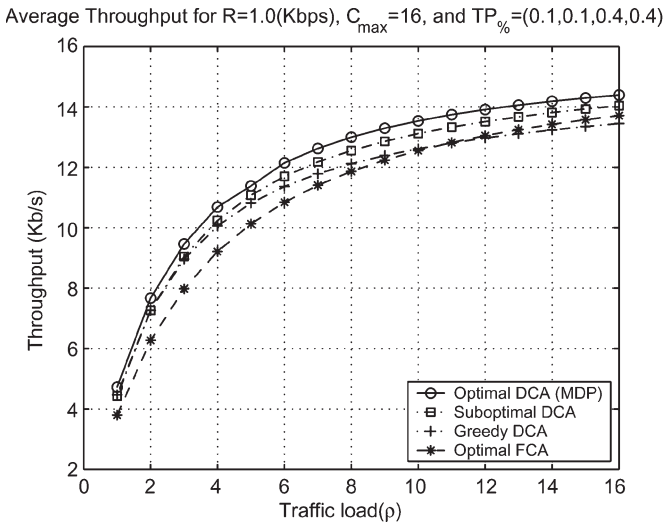


Fig. 11. Comparison of the average throughput when $C_{\max} = 16$ and TP $= (0.1, 0.1, 0.4, 0.4)$.

However, the optimal FCA does not perform well in a normal traffic condition where low-rate call requests are dominant.

In Fig. 11, we observe that the greedy DCA is a reasonable alternative to the optimal DCA when the traffic load is low, and the optimal FCA can be a good replacement for the optimal DCA when the system becomes heavily loaded. In summary, the greedy DCA can be used as an alternative to the optimal scheme when the traffic load is low or the traffic profile is normal. However, when the traffic enters into an extreme condition, the greedy DCA has to be replaced by better suboptimal schemes. The proposed suboptimal CAC policy not only generates the average throughput very close to the optimum but also offers consistent performance over the entire traffic load range. As pointed out in the previous section, the proposed suboptimal policy can be implemented online since the maximization with a given state and decision set is instantaneous.

VI. Conclusion and Future Work

Computationally efficient suboptimal DCA schemes with a CAC policy for OVSF-CDMA systems have been studied in this paper. Two hybrid DCA schemes have been proposed to reduce the design and implementation complexity by partitioning the total resource into smaller groups while achieving an average data throughput close to that of the optimal scheme by employing optimal DCA-CAC schemes in each partition. Numerical simulation results confirm that the two proposed code assignment schemes achieve satisfactory suboptimal performance over a wide range of traffic loads and profiles. Hence, they can be adopted as reasonable suboptimal solutions of code assignment for a large-scale OVSF code tree.

We have also proposed a suboptimal CAC policy for DCA. By designing a CAC policy from the observation of the behavior of system dynamics over only two consecutive stages starting from an event of call arrival, we can significantly reduce the complexity associated with MDP over the infinite time horizon from which the optimal policy is derived. By

approximation to the performance over infinite time horizon, we can achieve a high average data throughput that is close to the optimal performance. The superior performance of the proposed suboptimal CAC policy is presented via numerical analysis. Moreover, the possibility of online implementation makes it an attractive candidate in the real network environment where the traffic situation is varying over time.

This paper has focused on the one-cell environment. That is, given the capacity and traffic profiles of a cell, the proposed code assignment and CAC are independently performed at each cell. However, in a realistic network with multiple cells, there exists an interaction between cells, such as user mobility and interference among cells. This interaction may result in the capacity and/or traffic profiles of a cell varying with time. The extension of this paper to a more realistic network environment with multiple cells is an interesting yet challenging problem worth further investigation.

REFERENCES

[1] F. Adachi, M. Sawahashi, and H. Suda, "Wideband DS-CDMA for next-generation mobile communications systems," *IEEE Commun. Mag.*, vol. 36, no. 9, pp. 56–69, Sep. 1998.

[2] E. H. Dinan and B. Jabbari, "Spreading codes for direct sequence CDMA and wideband CDMA cellular networks," *IEEE Commun. Mag.*, vol. 36, no. 9, pp. 48–54, Sep. 1998.

[3] W.-T. Chen, Y.-P. Wu, and H.-C. Hsiao, "A novel code assignment scheme for W-CDMA systems," in *Proc. IEEE Veh. Technol. Conf.—Fall*, 2001, pp. 1182–1186.

[4] R.-G. Cheng and P. Lin, "OVSF code channel assignment for IMT-2000," in *Proc. IEEE Veh. Technol. Conf.—Spring*, 2000, pp. 2188–2192.

[5] Y. C. Tseng and C. M. Chao, "Code placement and replacement strategies for wideband CDMA OVSF code tree management," *IEEE Trans. Mobile Comput.*, vol. 1, no. 4, pp. 293–302, Oct./Dec. 2002.

[6] Y. Sekine, K. Kawanishi, U. Yamamoto, and Y. Onozato, "Hybrid OVSF code assignment scheme in W-CDMA," in *Proc. IEEE Pac. Rim Conf. Commun., Comput. Signal Process.*, Victoria, BC, Canada, Aug. 2003, pp. 384–387.

[7] N. Li and T.-S. P. Yum, "Allocating codes with coding scheme jointly to improve the system throughput of OVSF-CDMA systems," in *Proc. 9th Int. Conf. Commun. Syst.*, Singapore, Sep. 2004, pp. 50–54.

[8] T. Minn and K.-Y. Siu, "Dynamic assignment of orthogonal variable-spreading-factor codes in W-CDMA," *IEEE J. Sel. Areas Commun.*, vol. 18, no. 8, pp. 1429–1440, Aug. 2000.

[9] J.-S. Park and D. C. Lee, "Enhanced fixed and dynamic code assignment policies for OVSF-CDMA systems," in *Proc. Int. Conf. Wireless Netw.*, Las Vegas, NV, Jun. 2003, pp. 620–625.

[10] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. New York: Wiley, 1994.

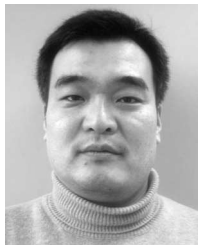
[11] E. A. Feinberg, "Contstrained semi-Markov decision processes with average rewards," *ZOR-Math. Methods Oper. Res.*, vol. 39, pp. 257–288, 1994.

[12] S.-C. Fang and S. Puthenpura, *Linear Optimization and Extensions: Theory and Algorithms*. Upper Saddle River, NJ: Prentice-Hall, 1993.

[13] L. Kleinrock, *Queueing Systems, Vol. I: Theory*. New York: Wiley, 1975.

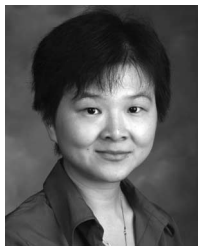
[14] D. P. Bertsekas, *Dynamic Programming and Optimal Control: Vol. 2*. Belmont, MA: Athena Scientific, 1995.

[15] S. M. Ross, *Applied Probability Models With Optimization Applications*. San Francisco, CA: Holden-Day, 1970.



Jun-Seong Park received the B.S. and M.S. degrees in electronics engineering from Seoul National University, Seoul, Korea, in 1991 and 1993, respectively, and the Ph.D. degree in electrical engineering from the University of Southern California, Los Angeles, in 2005.

From 1993 to 1996, he was a member of Technical Staff with KT Research Laboratory, Seoul, where he was involved in the development of Internet multimedia services. In 1999, he was a Research Engineer with LinCom Corporation, Los Angeles, CA, where he was involved in the performance evaluation of 2G wireless networks. Since 2005, he has been a Senior Research Engineer with FLYVO R&D Center, POSDATA Company, Ltd., Bundang, Korea. He is currently working on the design and performance evaluation of WiBro/Mobile WiMAX systems. His research interests are in the areas of radio resource management, cross-layer optimization, and system-level simulation for wireless networks.



Lei Huang received the B.S. and M.S. degrees in electrical engineering from Beijing University of Posts and Telecommunications, Beijing, China, in 1993 and 1996, respectively, and the Ph.D. degree in electrical engineering from the University of Southern California, Los Angeles, in 2003.

Since August 2003, she has been an Assistant Professor with the Department of Electrical Engineering and Computer Science, Loyola Marymount University, Los Angeles. Her research interests include digital image and video coding, multimedia applications in wired and wireless networks, quality of service, and network security.

C.-C. Jay Kuo (F'99) received the B.S. degree in electrical engineering from the National Taiwan University, Taipei, Taiwan, R.O.C., in 1980 and the M.S. and Ph.D. degrees in electrical engineering from the Massachusetts Institute of Technology, Cambridge, in 1985 and 1987, respectively.

He is the Director of the Signal and Image Processing Institute and a Professor of electrical engineering, computer science, and mathematics with the University of Southern California (USC), Los Angeles. He has guided about 80 students with their Ph.D. degrees and supervised 15 postdoctoral research fellows. His research group at USC currently consists of around 40 Ph.D. students. He is the coauthor of about 130 journal papers, 700 conference proceedings, and seven books. He has delivered more than 300 invited lectures at conferences, research institutes, universities, and companies. His research interests are in the areas of digital image/video analysis and modeling, multimedia data compression, communication and networking, and biological signal/image processing.

Dr. Kuo is a Fellow of The International Society for Optical Engineers and a member of Association for Computing Machinery. He was an IEEE Signal Processing Society Distinguished Lecturer in 2006. He is the Editor-in-Chief of the *Journal of Visual Communication and Image Representation* (an Elsevier journal) and has been the Editor for about ten international journals. He received the National Science Foundation Young Investigator Award and the Presidential Faculty Fellow Award in 1992 and 1993, respectively. He received the Northrop Junior Faculty Research Award from the USC Viterbi School of Engineering in 1994. He received the Best Paper Award from the Multimedia Communication Technical Committee of the IEEE Communication Society in 2005, the Best Student Paper Award from the IEEE Vehicular Technology Fall Conference in 2006, and the Best Paper Award from the IEEE Conference on Intelligent Information Hiding and Multimedia Signal Processing in 2006.