Code Placement and Replacement Strategies for Wideband CDMA OVSF Code Tree Management

Yu-Chee Tseng, *Member*, *IEEE Computer Society*, and Chih-Min Chao

Abstract—The use of OVSF codes in WCDMA systems has offered opportunities to provide variable data rates to flexibly support applications with different bandwidth requirements. Two important issues in such an environment are the *code placement problem* and *code replacement problem*. The former may have significant impact on code utilization and, thus, code blocking probability, while the latter may affect the code reassignment cost if dynamic code assignment is to be conducted. The general objective is to make the OVSF code tree as compact as possible so as to support more new calls by incurring less blocking probability and less reassignment costs. Earlier studies about these two problems either do not consider the structure of the OVSF code tree or cannot utilize the OVSF codes efficiently. To reduce the call blocking probability and the code reassignment cost, we propose two simple yet efficient strategies that can be adopted by both code placement and code replacement: *leftmost* and *crowded-first*. Numerical analyses on call blocking probability and bandwidth utilization of OVSF code trees when code reassignment is supported are provided. Our simulation results show that the crowded-first strategy can significantly reduce, for example, the code blocking probability by 77 percent and the number of reassignments by 81 percent, as opposed to the random strategy when the system is 80 percent fully loaded and the max SF = 256.

Index Terms—Mobile computing, OVSF, personal communication services, WCDMA, wireless communication, 3G.

1 Introduction

Mobile communications have come into our daily life recently. In the second-generation (2G) CDMA systems, such as IS-95, users are each assigned a single *Orthogonal Constant Spreading Factor (OCSF)* [20], [21]. Services provided in existing 2G systems are typically limited to voice, facsimile, and low-bit-rate data. To support higher-rate services, multiple OCSF codes may be used [11], [15], [17].

Beyond these 2G services, higher-rate services, such as file transfer and QoS-guaranteed multimedia applications, are expected to be supported by the third-generation (3G) systems. To satisfy different requirements, the system has to provide variable data rates. In the 3G wireless standards UMTS/IMT-2000 [1], [8], [12], [14], WCDMA has been selected as the technology for use in the UMTS terrestrial radio access (UTRA) FDD operation by European Telecommunication Standards Institute (ETSI). WCDMA can flexibly support mixed and variable-rate services. Such flexibility can be achieved by the use of Orthogonal Variable Spreading Factor (OVSF) codes as the channelization codes [10], [12]. OVSF has the ability to support higher and variable data rates with a single code using one transceiver, making its hardware less complex than the multicode OCSF [9].

- *Y.-C. Tseng is with the Department of Computer Science and Information Engineering, National Chiao-Tung University, Taiwan. Email: yctseng@csie.nctu.edu.tw.*
- *C.-M. Chao is with the Department of Computer Science and Information Engineering, National Central University, Taiwan. Email: cmchao@axp1.csie.ncu.edu.tw.*

Manuscript received 1 May 2002; revised 1 Oct. 2002; accepted 17 Oct. 2002. For information on obtaining reprints of this article, please send e-mail to: tmc@computer.org, and reference IEEECS Log Number 1-052002.

In this paper, we focus on the environment where one single OVSF code is available for each call (i.e., the multi-OVSF-code case such as that in [7], [19] is not considered). OVSF codes can be represented as a code tree [3], [16]. The data rates provided are always a power of two with respect to the lowest-rate codes. Two issues that are critical to the utilization of an OVSF code tree are the *code placement problem* and *code replacement problem*. The former addresses how to place a new call in the code tree to avoid the tree becoming too fragmented; it may have significant impact on the code utilization of the system. This is very similar to the traditional memory management problem in Operating System design [2]. The latter addresses how to relocate codes when a new call arrives finding no proper place to accommodate it. This can reduce code blocking, but will incur code reassignment costs.

This paper addresses both code placement and replacement issues in a WCDMA system where OVSF code trees are used for downlink transmission from base stations to mobile terminals. The general objective is to make the OVSF code tree as compact as possible so as to support more new calls by incurring less blocking probability and less reassignment costs. Three strategies which can be adopted by both code placement and code replacement are proposed: *random*, *leftmost*, and *crowded-first*. The random strategy is used here for comparison purposes. The leftmost strategy tries to place codes starting from the leftmost available position, while the crowded-first strategy tries to allocate a code in a subtree that has the most occupied codes. Our code replacement solution is built on top of the DCA algorithm in [18]. Numerical analyses on call blocking probability and bandwidth utilization of OVSF code trees when code reassignment is supported are provided. Our

1536-1233/02/$17.00 © 2002 IEEE Published by the IEEE CS, CASS, ComSoc, IES, & SPS

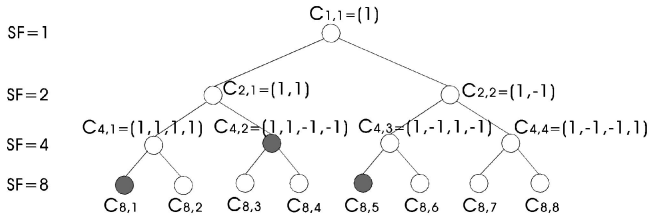


Fig. 1. Tree representation of OVSF codes. Busy codes are marked by gray.

simulation results have demonstrated that both the leftmost and the crowded-first strategies perform pretty well compared to the naive random strategy. Earlier studies about these two problems either do not consider the structure of the OVSF code tree or cannot utilize the code tree efficiently. A more detailed comparison to existing works [4], [6], [7], [13], [18], [19], [22] is in Section 2.

Section 2 defines the problems to be solved and provides some reviews. Section 3 presents our code placement and replacement strategies. Some performance analysis results are in Section 4. Simulation results are in Section 5. Conclusions are drawn in Section 6.

2 Problem Statement and Literature Review

In WCDMA, two operations are applied to user data. The first one is *channelization*, which transforms every data bit into a code sequence. The length of the code sequence per data bit is called the *spreading factor (SF)*, which is typically a power of two [16]. The second operation is *scrambling*, where it applies a scrambling code to the spread signal. Scrambling codes are used to separate the signals from different sources, while channelization codes are used to separate transmissions from a single source.

The *Orthogonal Variable Spreading Factor (OVSF)* codes are used as the channelization codes in UTRA. The possible OVSF codes can be represented by a code tree, as shown in Fig. 1. Each OVSF code can be denoted as $C_{SF,k}$, where SF is the spreading factor and k is the branch number, $1 \leq k \leq \text{SF}$. The number of codes at each level is equal to the value of SF. All codes in the same layer are orthogonal, while codes in different layers are orthogonal if they do not have an ancestor-descendant relationship. Leaf codes have the minimum data rate, which is denoted by $1R$. The data rate is doubled whenever we go one level up the tree. For example, in Fig. 1, $C_{4,1}$ has rate $2R$, and $C_{2,1}$ has rate $4R$. The "transmission unit" that can be assigned to a user is codes. Two users should not be given two codes that are not orthogonal.

Now, we define the problems to be solved in this paper. When a new call arrives requesting for a code of rate kR, where k is a power of two, we have to allocate a free code of rate kR for it. The **Code Placement Problem** is to address the allocation policy when multiple such free codes exist in the code tree. When no such free code exists but the remaining capacity of the code tree is sufficient (i.e., summation of data rates of all free codes is $\geq kR$), two possibilities occur. The first one is to reject this call, for which we call *code blocking*. The second is to relocate some codes in the code tree to "squeeze" a free space for the new call. This is called the **Code Replacement Problem**. These two problems are very similar to the traditional memory management problems in the Operation System.

Consider the scenario in Fig. 1. If a new call x requesting rate $1R$ arrives, any free code at the bottom of the code tree can be assigned to it. However, a bad placement may cause deficiency in the future. Suppose that a placement strategy chooses code $C_{8,8}$ for x. Later on, when another call x' arrives requesting a rate $2R$, it will be rejected for no space available. On the contrary, if the placement strategy chooses $C_{8,2}$ or $C_{8,6}$ for x, then x' can be placed in $C_{4,4}$. When a placement strategy cannot find a proper free code, one possibility is to perform code reassignment. Take the scenario in Fig. 1 again. When a new call arrives requesting a rate $4R$, we can either vacate the subtree $C_{2,1}$ or $C_{2,2}$. The former will cause two codes being relocated, while the latter only one code being relocated. So, the latter could be a better choice. Code replacement involves both determining which subtree is to be vacated and where to place those codes being relocated.

2.1 Related Works

The work in [13] is targeted at the integration of voice and data calls in WCDMA systems. It recommends that voice calls (with minimum bit rate $1R$) always be placed to the left-hand side of the code tree. Every time when a voice call terminates, the rightmost voice call is reassigned to the code just released. The goal is to keep larger free space at the right-hand side of the code tree. This strategy may incur a large number of unnecessary code reassignments. The strategy in [6] tries to place smaller calls on the left-hand side of the code tree and larger calls of the right-hand side. Whenever a call leaves, the same reassignment strategy is applied to readjust the code tree. As a result, the same problem of large unnecessary reassignments may exist. The time complexities of the above two strategies are both O(SF) in the worst case. The strategy proposed in [4] divides the code tree into several regions. Each region is reserved to support one specific data rate. When a region runs out of capacity, the codes from other regions are borrowed. The code tree partitioning strategy, which is obtained by the users' request probabilities, has great impact on system performance. But, it is difficult to find out, especially when the call patterns change dynamically. Even if an optimal partitioning can be found, the code tree may still become fragmented because of the borrowing activities. The time complexity is proportional to the size of the region being searched. It is O(SF) in the worst case. In [22], a *compact index* is defined for each code in the code tree. A code with a smaller compact index is regarded as more suitable to be assigned. This code assignment strategy is time-consuming since all compact indices need to be recalculated before each code assignment. In the worst case, the time complexity is $\mathrm{O}(\mathrm{M}log\mathrm{M})$, where M is the maximum SF. The basic idea of this strategy is similar to our crowded-first strategy. However, as will be shown in Section 3.1, our crowded-first strategy has a lower computation complexity at the expense of a little larger storage requirement.

A dynamic code reassignment algorithm is proposed in [18] to determine the subtree that can be vacated with the minimum cost, where the number of calls being relocated is

used as the performance metric. The scheme does not specifically address the code placement strategy when a new call arrives. Also, when relocating existing calls, it does not specify where to place those relocated codes. This strategy will be used by our scheme as a guideline when identifying which codes need to be relocated when reassignment is conducted.

The works in [7] and [19] assume that a call can be supported by more than one code in the OVSF code tree and, thus, extends to a multi-OVSF-code environment. Only code assignment strategies are considered. These two strategies focus on determining the number of codes and their data rates that should be assigned to a new call. However, they fail to address the exact positions in the OVSF code tree where these codes should be placed. In fact, as will be shown later, the locations of codes do play an important role in assignment, which significantly affects the system performance.

3 Code Placement and Replacement Strategies

3.1 Placement Schemes

When a call requesting kR arrives, where k is a power of two, we need to find a code to accommodate this call. Assuming that no code reassignment will be conducted, we propose three strategies below:

1. *Random.* If there is one or more than one code in the code tree with a rate kR, randomly pick any one and assign it to the call. Otherwise, the call is rejected.
2. *Leftmost.* If there is one or more than one code in the code tree with a rate kR, pick the leftmost one in the code tree and assign it to the call. Otherwise, the call is rejected. The intuition is to always vacate a larger capacity in the right-hand side of the code tree so as to accommodate higher-rate calls in the future.
3. *Crowded-first.* If there is one or more than one code in the code tree with a rate kR, pick the one whose ancestor code has the least free capacity. More specifically, suppose codes x and x' are both of rate kR. We will compare the free capacities in their ancestors, say y and y', respectively. The one with less free capacity (i.e., more crowded) will be picked to accommodate the new call. When there are ties, we will go one level up by comparing y and y''s ancestors. This is repeated until the subtree with the least free capacity is found. One special case is that y may represent the same subtree as y'. If so, we will follow the leftmost rule to pick the code on the left-hand side.

For example, consider the code tree in Fig. 2a. Suppose a new call arriving requesting a rate $2R$. By the random strategy, any of the codes $C_{16,1}$, $C_{16,5}$, $C_{16,6}$, $C_{16,7}$, $C_{16,8}$, $C_{16,15}$ may be picked to serve the new call. By the leftmost strategy, $C_{16,1}$ will be chosen. By the *crowded-first* strategy, we will compare these codes' ancestors. Among them, both $C_{8,1}$ and $C_{8,8}$ will be fully occupied if $C_{16,1}$ and $C_{16,15}$ are chosen, respectively. So, we further compare their ancestors, $C_{4,1}$ and $C_{4,4}$, but again this is a tie. After going one level up, we will find that $C_{2,2}$ is more crowded than $C_{2,1}$, so $C_{16,15}$ will be selected to serve the new call.

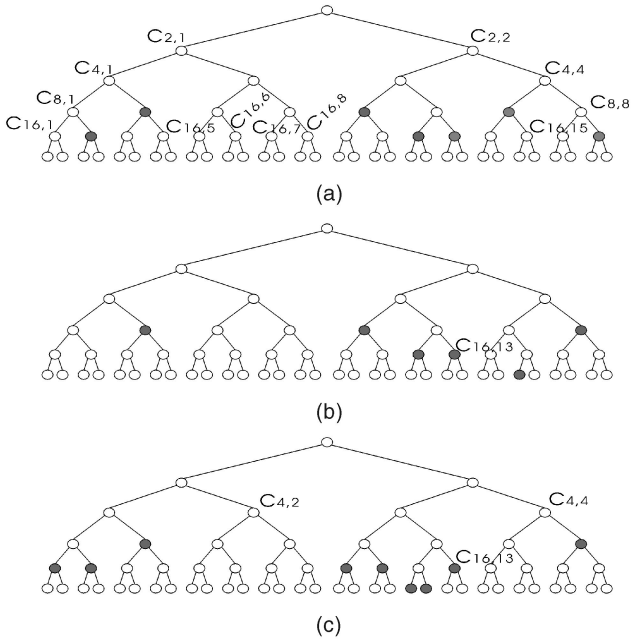


Fig. 2. Code placement examples (max SF = 32).

Still assuming a new call of rate $2R$, two more examples are in Figs. 2b and 2c. In Fig. 2b, the random and leftmost strategies may make the code tree more fragmented, while the crowded-first strategy will pick $C_{16,13}$, which can best fit the new call. In Fig. 2c, no place exists to best fit a $2R$ call. However, the crowded-first strategy will still pick $C_{16,13}$, which can leave a largest free capacity, $8R$, for future calls.

Theorem 1. *Given any code request to a partially occupied code tree, the crowded-first strategy will result in a code tree that has the largest code among all the possible code trees after the assignment.*

Proof. If there is no sufficiently large free code, the theorem holds trivially. Otherwise, let x be the largest free code before the assignment. If there is another free code x' of the same size as x, one of x and x' must remain free after the assignment. So, the theorem holds, too. Otherwise, let y be the second largest free code. Consider the code request, say r. If $r > y$, either x or one of x's descendants will be assigned to r. The largest remaining code is either y or one of $x's$ descendants. Thus, the theorem holds, too. If $r \leq y$, either y or one of y's descendants will serve as candidates to accommodate r. When searching bottom-up in the code tree, we will find that y's ancestor is more crowded than x. This ensures that x's descendants will not be assigned to r, which proves this theorem. □

The computation time required for the random strategy depends on the requested data rates. To find a free kR code, it is sufficient to search the level with rate kR. For example, if a request asks for a code with $\mathrm{SF} = 4$, four searches are needed in the worst case since the number of codes at each level equals the value of SF. Thus, the time complexity is proportional to the SF of the request, i.e., O(SF). Each code needs to store whether it is free or not.

The storage requirement is thus $\mathrm{SF}+\frac{\mathrm{SF}}{2}+\cdots+1=O(\mathrm{SF})$. For the leftmost strategy, it is easy to see that the time and storage complexity are also O(SF). For the crowded-first strategy, we must compare the free capacities of candidate codes' ancestor(s). To facilitate the comparison, each internal code should store the free capacity in its subtree (this additional storage requirement is also O(SF)). To find a free code with a specific SF, we have to search all candidate free codes of spreading factor SF. For each candidate, we may have to compare the free capacity of each of its ancestor codes. When we go one level up, the number of ancestors reduces to $\frac{\mathrm{SF}}{2}$ and further reduces to $\frac{\mathrm{SF}}{4}$ if we go two levels up. It is easy to see that the searching cost is $\mathrm{SF}+\frac{\mathrm{SF}}{2}+\cdots+1=O(\mathrm{SF})$. Besides the searching cost, an update cost has to be incurred whenever a code is allocated/released. When a code of spreading factor SF is allocated/released, all its ancestors have to update their remaining capacities. The cost is $\mathrm{O}(log\mathrm{SF})$.

3.2 Replacement Schemes

When the code tree is used for a long time, it is sometimes inevitable that the tree may become fragmented. In this case, a new call requesting for a rate kR may be rejected even if the total amount of free capacity in the code tree is $\geq kR$. This may result in low utilization of the code tree.

To resolve this problem, code reassignment can be conducted to squeeze a large-enough space for the new call. Code reassignment incurs costs on the system. In [18], a dynamic code assignment (DCA) algorithm is proposed based on *code pattern search* to find a branch of rate kR in the code tree which can be vacated with the minimum cost (minimum-cost branch). However, where to place those relocated codes is not addressed in [18]. Placement strategies for those relocated codes may have an impact on the system performance in the future, too.

In this paper, we look for a total solution. Our solution is, in fact, built on top of the DCA algorithm proposed in [18]. In case a new call arrives requesting a rate kR, but no free code of such a rate exists, the following steps are taken:

1. If the total amount of free capacity in the code tree is $\geq kR$, apply the DCA algorithm in [18] to find the minimum-cost branch with rate kR. Otherwise, the call is rejected.
2. For the busy codes in the branch found above, we relocate them one-by-one starting from those with higher rates. For each busy code being relocated, we replace it based on any of our earlier strategies (random, leftmost, or crowded-first). Note that it is possible that no free code exists to accommodate the relocated call. If so, this will trigger another round of the DCA algorithm and, thus, another round of relocation. This is repeated recursively until all busy codes are relocated. Such a process is guaranteed to complete because there is sufficient free capacity in the code tree. Also, note that, for each relocated call, we should stick with the same placement strategy consistently.

For example, in the code tree in Fig. 3, suppose a new call arrives requesting a rate $8R$. Code $C_{4,1}$ is the minimum-cost branch because it contains only two busy codes, $C_{8,1}$ and $C_{32,7}$. When relocating $C_{8,1}$, another reassignment will be triggered. This time, $C_{8,8}$ is selected as the minimum-cost branch because it contains only one busy code $C_{32,29}$. Based on our crowded-first strategy, $C_{32,29}$ will be replaced to $C_{32,12}$. Then, $C_{8,1}$ can be replaced to $C_{8,8}$. Last, $C_{32,7}$ will be replaced to $C_{32,22}$. Now, $C_{4,1}$ is vacated to accommodate the new $8R$-call.

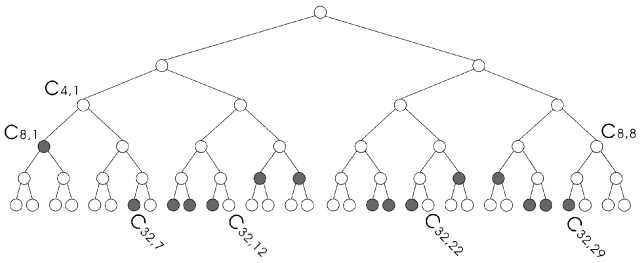


Fig. 3. A code replacement example.

4 Performance Analysis

In this section, we establish an analytical model to evaluate the performance of an OVSF code tree to support calls. We consider a code tree with a certain maximum SF. Calls may arrive requesting transmission rates ranging in $1R$, $2R$, $4R$, and $8R$. Calls requesting each transmission rate will be modeled by distinct arrival and service rates. The goal is to analyze the call blocking probability and bandwidth utilization of the code tree when code reassignment is supported. Note that, although the maximum transmission rate is $8R$, our analytic result can be easily extended to a higher transmission rate. Also note that, when code reassignment is supported, a call can always be accepted as long as the total remaining capacity of the code tree (either fragmented or nonfragmented) is at least as large as the requested capacity. As a result, our result is a very general one, which is independent of what code placement strategy is used for (comparison of strategy-dependent performances will be studied in the next section through simulations). The analytic results will then be verified by simulation experiments.

The problem will be modeled by a Markov chain. The state of the code tree will be represented by the transmission rates of all calls currently being supported by the code tree. These rates will be denoted by a sequence of numbers. For example, when there are three calls with transmission rates $8R$, $2R$, and $1R$ in the system, the state of the code tree will be denoted by (821) and there are four calls with rates $4R$, $4R$, $1R$, and $1R$ in the system, the state of the code tree will be denoted by (4411). We comment that the order of numbers in a state is insignificant in our representation. For example, (4411) = (4141) = (1144). The reason is that, when code reassignment is supported, these states always have the same amount of remaining capacity in the code tree. However, we regard (4411) and (4222) as different states because calls with different transmission rates may have different arrival and service rates.

For example, in Fig. 4., we list some of the states in a code tree with a maximum SF of 64. The way that we group these

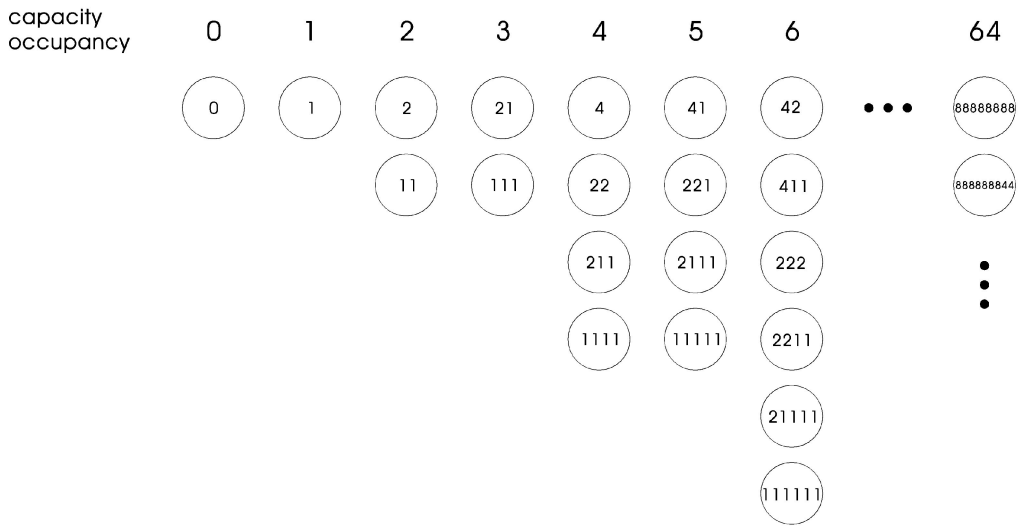


Fig. 4. States of a code tree with maximum SF of 64.

states is by total capacities being occupied. For example, when the capacity occupancy is 2, there are two states, (2) and (11), and when the capacity occupancy is 6, there are six states, (42), (411), (222), (2211), (21111), and (111111).

To obtain all possible states in the system, we define a generation function $f(x)$ as follows:

$$f(x) = \frac{1}{1-x} \times \frac{1}{1-x^2} \times \frac{1}{1-x^4} \times \frac{1}{1-x^8}. \quad (1)$$

The number of states which have a capacity occupancy of c can be obtained by calculating the coefficient of x^c in $f(x)$. The calculation can be interpreted by the *partition of an integer* problem [5] with parts 1, 2, 4, and 8. Note that, interestingly, this coefficient is independent of the value of the maximum SF since $f(x)$ is not affected by it. The total number of states of the system is the sum of all coefficients of x^c for $c = 0, 1, \ldots, \max$ SF. In Table 1, we demonstrate the number of states with a capacity occupancy of cR and the total number of states for different sizes of code trees.

After generating all possible states, the next step is to obtain the state transition diagram. For example, we illustrate in Fig. 5 all possible transitions into and out of state (211). Here, we assume that maximum SF = 8. Parameters λ_i and μ_i represent call arrival rate and call service rate, respectively, for calls requesting for transmission rate iR. Note that the call departure rate to another state is proportional to the number of codes of the same transmission rate in the source state. Also, note that the dashed lines actually represent nonexisting transitions.

To obtain all transitions, we need to define some notations. Note that, in this paper, we assume that all calls arrive according to a Poisson process and depart according to an exponential process.

- Φ: the maximum SF.
- λ_i: the arrival rate for calls requesting a transmission rate iR.
- μ_i: the service rate for calls with a transmission rate iR.
- P_s: the steady-state probability for the code tree remaining in state s.
- $N_i(s)$: the number of calls with a transmission rate iR in state s. (For example, $N_4(411) = 1$, $N_1(4111) = 3$.)
- C_s: the capacity occupancy in state s. (For example, $C_{(411)} = 6$ and $C_{(4411)} = 10$.)
- $s \oplus i$: the state after adding a new call of transmission rate iR into state s. (For example, $(411) \oplus 2 = (4211)$.)
- $s \ominus i$: the state after subtracting an existing call of transmission rate iR from state s. (For example, $(4211) \ominus 2 = (411)$.)
- $F(s, i)$: the feasible function, where s is a state and $i = 1, 2, 4$, or 8 such that

TABLE 1
(a) The Numbers of States at Different Capacity Occupancies and (b) the Total Numbers of States at Different Maximum SFs

capacity occupancy	8	16	32	64	128	256
no. of states	10	35	165	969	6545	47905

(a)

maximum SF	8	16	32	64	128	256
total no. of states	36	201	1625	17361	222241	3160641

(b)

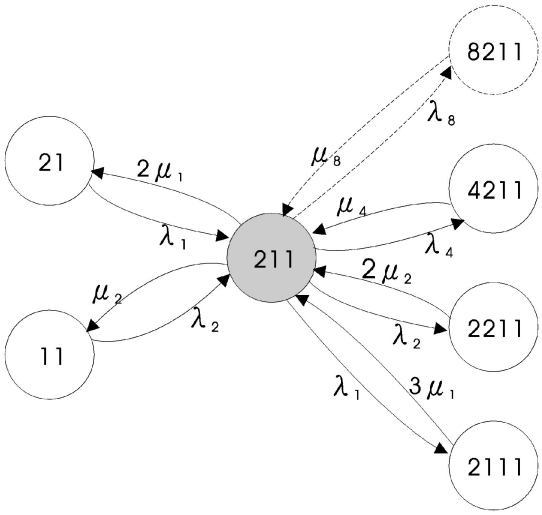


Fig. 5. Transitions into and out of state (211) in a code tree with maximum SF of 8.

$$F(s,i) = \begin{cases} 1 & \text{if } C_{s\oplus i} \le \Phi \\ 0 & \text{otherwise.} \end{cases}$$

(Intuitively, this denotes whether adding a call with a transmission rate of iR into state s is legal or not. If the new capacity occupancy exceeds the maximum SF, this is illegal and is denoted as 0.)

From the flow equilibrium property, we can derive from each state s the following equation (the left-hand side contains the flow-out traffics, while the right-hand side the flow-in traffics):

$$\begin{aligned} P_s \cdot \sum_{i=1,2,4,8} (N_i(s)\cdot\mu_i) + P_s \cdot \sum_{i=1,2,4,8} (F(s,i)\cdot\lambda_i) = \\ \sum_{i=1,2,4,8} (P_{s\oplus i}\cdot F(s,i)\cdot N_i(s\oplus i)\cdot\mu_i) + \\ \sum_{i=1,2,4,8;N_i(s)>0} (P_{s\ominus i}\cdot\lambda_i). \end{aligned} \quad (2)$$

Also, the summation of steady-state probabilities of all states equals 1,

$$\sum_{\forall s} P_s = 1. \quad (3)$$

By solving the above equations, we can obtain the steady-state probability P_s of each state s.

4.1 Call Blocking Probability

Call blocking occurs when a call arrives requesting a transmission rate iR, but adding such a call into the current state will lead to an illegal state. This is dependent on the maximum SF of the code tree. Given any integer i, $0 \le i \le \Phi$, let's define the *accumulated probability* $P_a(i)$ to be

$$P_a(i) = \sum_{\forall s: C_s = i} P_s. \quad (4)$$

This represents the probability summation of all those states which have a capacity occupancy of i in the system. For example,

$$P_a(6) = P_{(42)} + P_{(411)} + P_{(222)} + P_{(2211)} + P_{(21111)} + P_{(111111)}.$$

Let's denote by $P_B(\Phi)$ the *call blocking probability* of the system, given the maximum SF of Φ. This probability can be derived as

$$P_B(\Phi) = \frac{\lambda_1 \cdot P_a(\Phi) + \lambda_2 \cdot \sum_{i=\Phi-1}^{\Phi} P_a(i) + \lambda_4 \cdot \sum_{i=\Phi-3}^{\Phi} P_a(i) + \lambda_8 \cdot \sum_{i=\Phi-7}^{\Phi} P_a(i)}{\lambda_1+\lambda_2+\lambda_4+\lambda_8}. \quad (5)$$

4.2 Bandwidth Utilization

The utilization of a code tree, denoted by U_Φ, can be obtained by summing utilizations of states with the same capacity occupancy divided by the total capacity of the code tree, i.e.,

$$U_\Phi = \frac{\sum_{i=1}^{\Phi} P_a(i)\cdot i}{\Phi}. \quad (6)$$

4.3 Numerical Results

We have implemented the above formulations by Mathematica 3.0 to derive the steady-state probabilities, call blocking probability, and bandwidth utilization (the details are in the Appendix). The implementation is on an IBM PC-compatible computer. The results are shown in Table 2 and Table 3 for $\Phi = 16$ (indicated by "theoretical"). Unfortunately, limited by the computing and space factors, we were not able to calculate the results for $\Phi > 16$ due to the state-explosion problem (recall Table 1, when $\Phi = 32$, there are 1,625 states with 1,625 flow-equilibrium equations). Solutions to this problem may count on more advanced software tools or high-performance computers or on further reducing the number of states in our Markov chain (which can be led to future research).

To verify the correctness of our analysis, we have also developed a simulator to calculate the call blocking probability and bandwidth utilization. Each value in the simulation curve is obtained from an average of 100 simulation runs, where each run contains 4,000 accepted calls. The results are shown in Table 2 and Table 3. We see that our numerical analysis fits very closely with the simulation results. This is also true for other call patterns (e.g., $8R:4R:2R:1R = 4:4:1:1$) since we use individual variable λ_i and individual variable μ_i for each code rate in our analysis.

5 Simulation Results

We have implemented a simulator to evaluate the performance of the proposed strategies. Two kinds of max SF, 64 and 256, were tested. New calls arrived in a Poisson distribution. Each call might request a rate of $8R$, $4R$, $2R$, or $1R$. Different combinations of these code rates were tested. Call duration is exponentially distributed with a mean of four time units. Six different policies are simulated: RN, LN, CN, RR, LL, and CC, where the first letter indicates the code placement strategy and the second letter indicates the code replacement strategy (R = random, L = leftmost, and C = crowded-first). N means that code replacement is not implemented. Note that it makes no sense to combine different strategies, such as RL and LC, in the placement and replacement parts.

TABLE 2
Call Blocking Probability at Different Traffic Load when max SF = 16

	call blocking (μ=1/4)		call blocking (μ=1/3)		call blocking (μ=1/2)	
λ	theoretical	simulation	theoretical	simulation	theoretical	simulation
1/30	0.0168	0.017	0.0101	0.0103	0.0049	0.0048
2/30	0.0521	0.0529	0.0332	0.0334	0.0175	0.0168
3/30	0.093	0.0937	0.0621	0.0618	0.033	0.0332
4/30	0.1341	0.1341	0.093	0.0931	0.0521	0.0521
5/30	0.1734	0.1733	0.124	0.1235	0.0724	0.0723
6/30	0.21	0.2104	0.1541	0.1548	0.0938	0.093
7/30	0.2437	0.2442	0.1828	0.1848	0.116	0.1137
8/30	0.2746	0.2746	0.21	0.2108	0.1355	0.1341
9/30	0.3029	0.3026	0.2355	0.2366	0.1545	0.1541
10/30	0.3289	0.3295	0.2595	0.2588	0.1736	0.1734
11/30	0.3527	0.3524	0.2819	0.2824	0.1927	0.1921
12/30	0.3747	0.3749	0.3029	0.304	0.2108	0.21

(call arrival rate $\lambda_1 = \lambda_2 = \lambda_4 = \lambda_8 = \lambda$, call service rate $\mu_1 = \mu_2 = \mu_4 = \mu_8 = \mu$)

TABLE 3
Code Tree Utilization at Different Traffic Load when max SF = 16

	utilization (μ=1/4)		utilization (μ=1/3)		utilization (μ=1/2)	
λ	theoretical	simulation	theoretical	simulation	theoretical	simulation
1/30	0.1215	0.1215	0.0922	0.0924	0.062	0.0621
2/30	0.2285	0.229	0.1772	0.1781	0.1215	0.1221
3/30	0.3182	0.3187	0.2525	0.2524	0.1772	0.1775
4/30	0.3924	0.3914	0.3182	0.3177	0.2285	0.2288
5/30	0.4537	0.453	0.3752	0.3746	0.2755	0.2753
6/30	0.5047	0.5051	0.4245	0.4243	0.3182	0.3187
7/30	0.5475	0.5475	0.4673	0.4689	0.3571	0.3583
8/30	0.5837	0.5827	0.5047	0.505	0.3924	0.3935
9/30	0.6146	0.614	0.5375	0.5381	0.4245	0.4253
10/30	0.6412	0.6414	0.5663	0.5661	0.4537	0.4533
11/30	0.6643	0.6642	0.5918	0.5923	0.4804	0.4813
12/30	0.6845	0.6843	0.6146	0.6149	0.5047	0.5052

(call arrival rate $\lambda_1 = \lambda_2 = \lambda_4 = \lambda_8 = \lambda$, call service rate $\mu_1, = \mu_2 = \mu_4 = \mu_8 = \mu$).

In the following, we make observations from three aspects:

1. *Impact of Code Placement.* In this experiment, we adopt different code placement strategies without using code replacement. We observe two metrics:
   a. the *code blocking probability* for a new call being rejected because of no code available, but the total amount of free capacity in the code tree is sufficiently large (note that code blocking is different from the call blocking defined in Section 4) and
   b. the *utilization* of the code tree.

   The traffic pattern used here is $8R : 4R : 2R : 1R = 1 : 1 : 1 : 1$ (calls have equal probability to request for rates $8R$, $4R$, $2R$, $1R$).

   Fig. 6a shows the code blocking probability at different traffic loads when maximum SF = 256. We can see that the crowded-first strategy performs the best, which is followed by the leftmost, and then the random strategy. At light load, the blocking probability is quite insensitive to the code placement strategy. After the system is around 60 percent fully loaded (at around load = 160), the code placement strategy will have a significant impact on blocking probability. For example, when the code tree is about 80 percent fully loaded (at around load = 200), the blocking probabilities are 5.5 percent, 1.6 percent, and 1.2 percent for RN, LN, and CN, respectively. Fig. 6b shows the same simulation when maximum SF = 64. The trend is similar, but code placement strategy starts to have an impact when the system is about 37 percent fully loaded. This indicates an interesting phenomenon that code placement is more important when SF is smaller. Here, we do not consider RR, LL, or CC because no code blocking will happen when code replacement strategies are implemented.

   Next, we compare the utilization of the code tree obtained by different strategies. Figs. 7a and 7b show the results when maximum SF = 256 and 64, respectively. Note that, in order to know the maximum possible utilization, we also draw the utilization curve of CC as an upper bound (with code replacement, no new call will be rejected as long as the code tree has sufficient remaining capacity). Still,

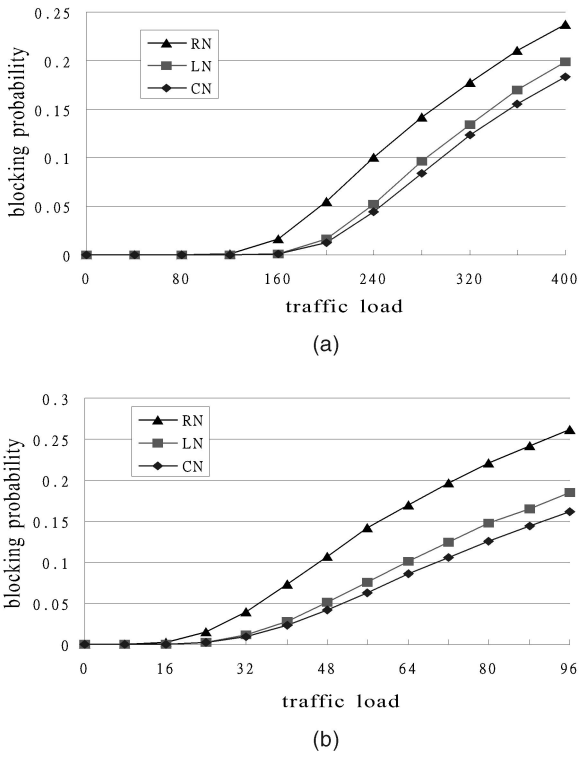


Fig. 6. Blocking probability at different traffic loads: (a) SF = 256 and (b) SF = 64.

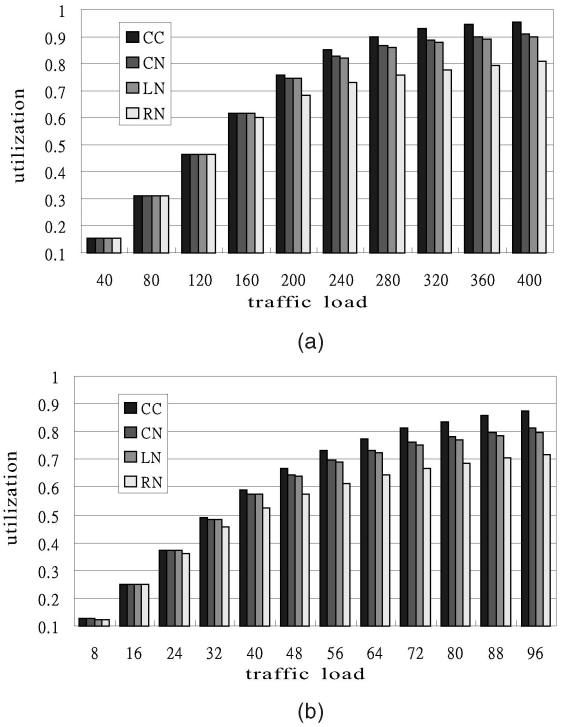


Fig. 7. Code tree utilization at different traffic loads: (a) SF = 256 and (b) SF = 64.

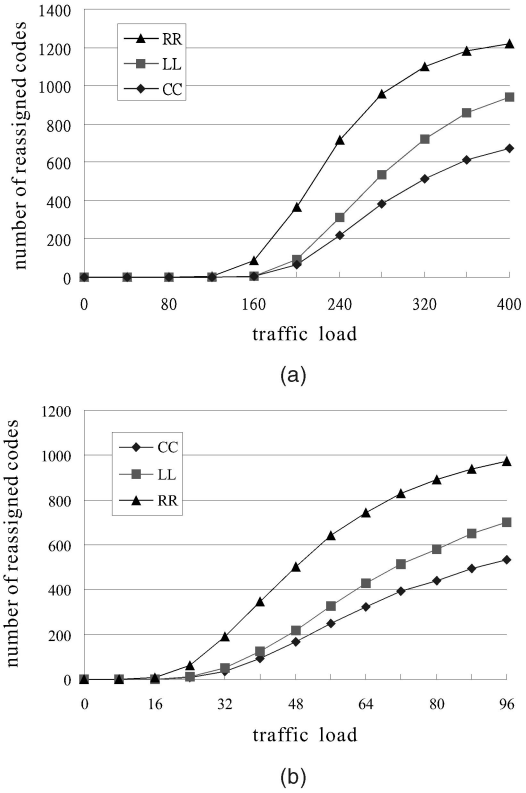


Fig. 8. Reassigned costs at different traffic loads: (a) SF = 256 and (b) SF = 64.

CN has the highest utilization, followed by LN, and then by RN, as the code tree is more heavily loaded.

2. *Impact of Code Replacement:* In this experiment, we adopt code replacement to observe its effect. We compare three strategies: RR, LL, and CC. The performance metric is the number of reassignments being taken. (Comparing code blocking probability makes less sense here because we can always conduct reassignment.)

   Fig. 8a shows the result when SF = 256. CC still outperforms LL, which is followed by RR. Code replacement will start to have a significant impact on performance after the code tree is around 50 percent fully loaded (load = 120). For example, when the code tree is about 80 percent fully loaded, the number of reassigned codes are 365, 94, and 67 for RN, LN, and CN, respectively. Fig. 8b shows the same simulation when SF = 64. The trend is similar and, again, code replacement will start to have an impact at a lighter load when SF is smaller.

   We also compare the number of calls that are accepted because of code relocation and the average number of relocations to support such calls. The result is in Table 4. The crowded-first scheme, due to its compact code assignment, has the least number of calls that are supported due to code relocation. The average numbers of replacements per call are relatively small for all three schemes and the crowded-first scheme still performs the best.

3. *Impact of Call Pattern.* In earlier experiments, we assume that calls request for rates $8R$, $4R$, $2R$, and R with equal probabilities. Here, we test more patterns: $8R : 4R : 2R : 1R = 4 : 4 : 1 : 1$, $4 : 1 : 1 : 4$, and $1 : 1 : 4 : 4$. Note that these are relative values. How calls arrive will depend on the given traffic loads. For example, when traffic load $= 240$, the

TABLE 4
Number of Calls (N) that Are Accepted Because of Code Relocation and Average Number of Relocations per Such Call (AVG)

	SF=64/load=96		SF=256/load=400	
	N	AVG	N	AVG
random	658	1.48	716	1.7
leftmost	485	1.45	570	1.65
crowded-first	370	1.43	418	1.6

number of $8R$ calls for pattern 1:1:1:1 and that for pattern 1:1:4:4 will be different. The effect on code blocking probability and number of reassignments are shown in Figs. 9 and 10, respectively. In Fig. 9, we see more advantages (larger gaps) of the crowded-first strategy over the other two strategies when the patterns are 4:4:1:1 and 4:1:1:4. This is reasonable because bad placement of smaller calls (such as $1R$ calls) will make the code tree more fragmented, thus blocking more larger calls. The advantage of the crowded-first strategy is reduced when the call pattern is 1:1:4:4 since there are fewer larger codes. In fact, 1R calls are never rejected as long as the code tree is not full. Similar results can be found in Fig. 10.

6 Conclusions

Wireless bandwidth is a precious resource. Thus, its resource management is an important issue. In this paper, we have addressed the code placement and replacement problems on WCDMA OVSF code trees. We have shown that the code placement and replacement strategies do have significant impacts on the utilization of code trees and, thus, the call blocking probability. The main idea is to keep the code tree less fragmented so as to accept more calls. A simulation model has been proposed to evaluate the utilization of a WCDMA OVSF code tree. Among the strategies we propose, the crowded-first strategy looks most promising. The result is expected to more efficiently utilize the scarce wireless bandwidth in WCDMA systems.

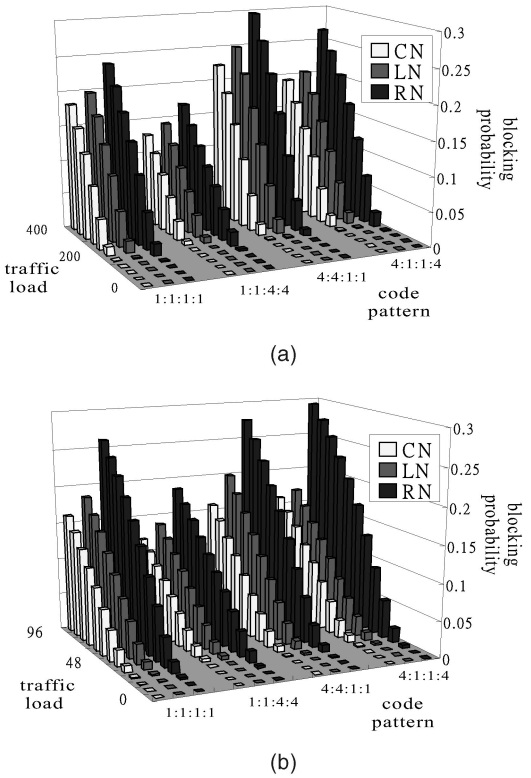


Fig. 9. The effect of different code patterns on blocking probability: (a) SF = 256 and (b) SF = 64.

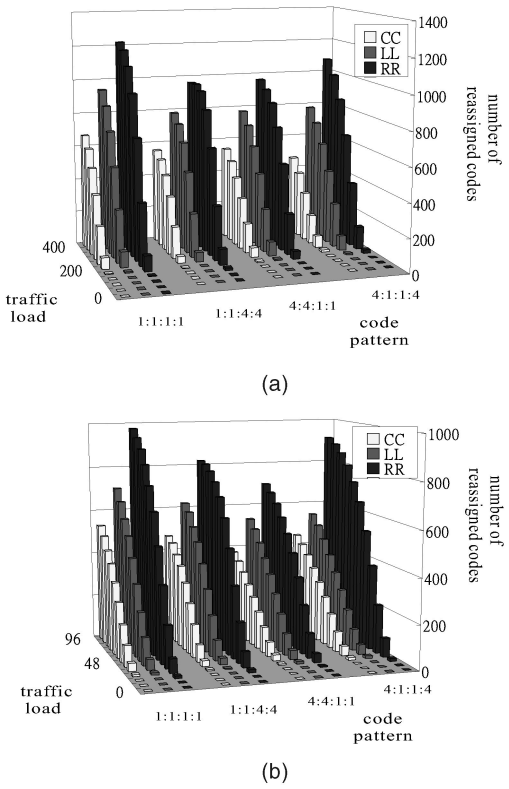


Fig. 10. The effect of different call patterns on reassignment cost: (a) SF = 256 and (b) SF = 64.

Appendix

Finally, we remark how we implemented our formulations by Mathematica. As pointed out in Section 4, the number of states is extremely large even for a small Φ. Manually deriving all state transitions and flow-equilibrium equations is almost infeasible and error-prone. To resolve this problem, we write a program to generate all the system states and the flow-equilibrium equations. These flow-equilibrium equations are in a pure text format, which can be fed into Mathematica to get the steady-state probabilities for all system states. The format in Mathematica to solve simultaneous equations is:

$$Solve[\{equation1, equation2, \ldots\}, \{variable1, variable2, \ldots\}],$$

where an $equation$ is of the format left_hand_side = = right_hand_side. For example, when $\Phi = 16$, the flow-equilibrium equation for state (211) is

$$P\_211 \times 2\mu + P\_211 \times \mu + P\_211 \times \lambda + P\_211 \times \lambda +$$
$$P\_211 \times \lambda + P\_211 \times \lambda == P\_8211 \times \mu + P\_4211 \times \mu +$$
$$P\_2211 \times 2\mu + P\_2111 \times 3\mu + P\_21 \times \lambda + P\_11 \times \lambda.$$

From the state probabilities, we then calculate the call blocking probability and bandwidth utilization as derived above.

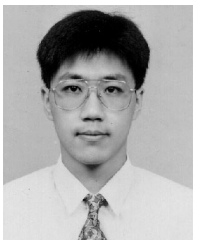
ACKNOWLEDGMENTS

This work is cosponsored by the Lee and MTI Center for Networking Research at the National Chiao Tung University and the MOE Program for Promoting Academic Excellence of Universities under grant numbers A-91-H-FA07-1-4 and 89-E-FA04-1-4.

REFERENCES

[1] Third Generation Partnership Project, Technical Specification Group Radio Access Network, Spreading and Modulation (FDD), http://www.3gpp.org, 1999
[2] A.S. Tanenbaum, *Modern Operating Systems.* pp. 81-87, Prentice Hall, 1992.
[3] F. Adachi, M. Sawahashi, and K. Okawa, "Tree-Structured Generation of Orthogonal Spreading Codes with Different Lengths for Forward Link of DS-CDMA Mobile Radio," *Electronic Letters,* vol. 33, pp. 27-28, Jan. 1997.
[4] R. Assarut, K. Kawanishi, U. Yamamoto, Y. Onozato, and M. Masahiko, "Region Division Assignment of Orthogonal Variable-Spreading-Factor Codes in W-CDMA," *Proc. IEEE Vehicular Technology Conf. Fall,* pp. 1884-1888, 2001.
[5] C.L. Liu, *Introduction to Combinatorial Mathematics.* 1968.
[6] W.-T. Chen, Y.-P. Wu, and H.-C. Hsiao, "A Novel Code Assignment Scheme for W-CDMA Systems," *Proc. IEEE Vehicular Technology Conf. Fall,* pp. 1182-1186, 2001.
[7] R.-G. Cheng and P. Lin, "OVSF Code Channel Assignment for IMT-2000," *Proc. IEEE Vehicular Technology Conf. Spring,* pp. 2188-2192, 2000.
[8] E. Dahlman, B. Gudmundson, M. Nilsson, and J. Skold, "UMTS/IMT-2000 Based on Wideband CDMA," *IEEE Comm. Magazine,* vol. 36, pp. 70-80, Sept. 1998.
[9] E. Dahlman and K. Jamal, "Wide-Band Services in a DS-CDMA Based FPLMTS System," *Proc. IEEE Vehicular Technology Conf. 1996,* pp. 1656-1660, 1996.
[10] A. Baier et al., "Design Study for a CDMA-Based Third-Generation Mobile Radio System," *IEEE J. Selected Areas in Comm.,* vol. 12, no. 4, pp. 733-743, May 1994.
[11] C.L. I et al., "IS-95 Enhancements for Multimedia Services," *Bell Labs. Technical J.,* pp. 60-87, Autumn 1996.
[12] F. Adachi, M. Sawahashi, and H. Suda, "Wideband DS-CDMA for Next-Generation Mobile Communications Systems," *IEEE Comm. Magazine,* vol. 36, pp. 56-69, Sept. 1998.
[13] R. Fantacci and S. Nannicini, "Multiple Access Protocol for Integration of Variable Bit Rate Multimedia Traffic in UMTS/IMT-2000 Based on Wideband CDMA," *IEEE J. Selected Areas in Comm.,* vol. 18, no. 8, pp. 1441-1454, Aug. 2000.
[14] H. Holma and A. Toskala, *WCDMA for UMTS.* John Wiley & Sons, 2000.
[15] C.L. I and D. Gitlin, "Multi-Code CDMA Wireless Personal Communications Networks," *Proc. ICC'95,* pp. 1060-1064, June 1995.
[16] J.G. Proakis, *Digital Communications.* McGraw-Hill, 1995.
[17] M.S.K. Ohno and F. Adachi, "Wideband Coherent DS-CDMA," *Proc. IEEE Vehicular Technology Conf. ,* pp. 779-783, 1995.
[18] T. Minn and K.-Y. Siu, "Dynamic Assignment of Orthogonal Variable-Spreading-Factor Codes in W-CDMA," *IEEE J. Selected Areas in Comm.,* vol. 18, no. 8, pp. 1429-1440, Aug. 2000.
[19] F. Shueh, Z.-E. P. Liu, and W.-S. E. Chen, "A Fair, Efficient, and Exchangeable Channelization Code Assignment Scheme for IMT-2000," *Proc. IEEE Int'l Conc. Personal Wireless Comm.,* pp. 429-433, 2000.
[20] T.S. Rappaport, *Wireless Communications: Principles and Practice.* Prentice Hall, 1996.
[21] V.K. Garg, *IS-95 CDMA and cdma2000.* Prentice Hall, 2000.
[22] Y. Yang and T.-S.P. Yum, "Nonrearrangeable Compact Assignment of Orthogonal Variable-Spreading-Factor Codes for Multi-Rate Traffic," *Proc. IEEE Vehicular Technology Conf. 2001 Fall,* pp. 938-942, 2001.

Yu-Chee Tseng received the BS and MS degrees in computer science from National Taiwan University and National Tsing-Hua University in 1985 and 1987, respectively. He worked for D-LINK, Inc., as an engineer in 1990. He obtained the PhD degree in computer and information science from the Ohio State University in January 1994. From 1994 to 1996, he was an associate professor in the Department of Computer Science, Chung-Hua University. He joined the Department of Computer Science and Information Engineering, National Central University, in 1996, and has been a full professor since 1999. Since August 2000, he has been a full professor in the Department of Computer Science and Information Engineering, National Chiao-Tung University, Taiwan. Dr. Tseng has served as a program committee member in many international conferences, as a program chair for the Wireless Networks and Mobile Computing Workshop, in 2000 and 2001, as an associate editor for *The Computer Journal*, and as a guest editor for the *ACM Wireless Networks*, *IEEE Transactions on Computers*, *Journal of Internet Technology*, and *Wireless Communications and Mobile Computing*. His research interests include wireless communication, network security, parallel and distributed computing, and computer architecture. Dr. Tseng is a member of the IEEE Computer Society.



Chih-Min Chao received the BS and MS degrees in computer science from Fu-Jen Catholic University and National Tsing-Hua University in 1992 and 1996, respectively. He is currently a PhD candidate in the Department of Computer Science and Information Engineering, National Central University. His research interests include mobile computing and wireless communication, with a current focus on resource management in WCDMA systems.