Reducing Internal and External Fragmentations of OVSF Codes in WCDMA Systems With Multiple Codes

Chih-Min Chao, Yu-Chee Tseng, *Senior Member, IEEE*, and Li-Chun Wang, *Member, IEEE*

***Abstract*—In the third-generation partnership project technical specification, orthogonal variable spreading factor (OVSF) codes are used as channelization codes. The use of OVSF codes can provide variable data rates to flexibly support applications with different bandwidth requirements. Most works in the literature assume that only one single OVSF code is used to support one connection. This may sometimes waste the scarce wireless bandwidth since the allocated bandwidth will increase exponentially as the spreading factor decreases, i.e., a user may be "overserved." In this paper, we consider the possibility of using multiple OVSF codes to support a connection. We show how using multiple codes can reduce the internal and external fragmentations of an OVSF code tree. The tradeoff between bandwidth utilization and hardware complexity of a multicode system is analyzed. The result shows that using two or three codes will be quite cost effective. Several multicode assignment and reassignment strategies, namely, random, leftmost, crowded-first-space, and crowded-first-code, are proposed based on such an environment. Our simulation results show significant increase in code tree utilization and significant reduction in code blocking probability by using the crowded-first-space and crowded-first-code schemes.**

***Index Terms*—Mobile communication, OVSF, personal communication services, third generation (3G), WCDMA, wireless communication.**

I. Introduction

SERVICES provided in existing second-generation (2G) personal communication systems (PCSs) are typically limited to voice, facsimile, and low-bit-rate data. It is expected that higher-bit-rate services such as file transfer and quality-of-service-guaranteed multimedia applications will be supported in the next-generation [third generation (3G) or beyond] systems. To support these different services, the 3G wireless standards Universal Mobile Telecommunications System (UMTS)/International Mobile Telecommunications 2000 [1], [3], [8], [9], [11] have proposed to use wideband code division multiple access (WCDMA) techniques to provide higher transmission rates.

This paper focuses on the downlink transmission from base stations to mobile terminals. To provide higher and variable data rates, two schemes are proposed in the 3G wireless standard: multicode-CDMA (MC-CDMA) and orthogonal variable spreading factor CDMA (OVSF-CDMA). In MC-CDMA, multiple orthogonal constant spreading factor (SF) codes can be assigned to a user [10], [12]. The maximum data rate a user can receive depends on the number of transceivers in the devices. Therefore, a higher rate implies higher cost. In OVSF-CDMA, a single OVSF code is assigned to each user. Higher data rates are provided by using lower SFs.

OVSF codes can be represented as a binary code tree [2], [14]. The data rates provided by OVSF codes are always powers of 2 with respect to the lowest basic rate $R_{\rm b}$. Thus, the possible rates that are supported are: $R_{\rm b}$, $2R_{\rm b}$, $4R_{\rm b}$, $8R_{\rm b}$, etc. Since the gap is becoming larger as the rate increases, in some cases a request may be "overserved" by a too large rate. For instance, a call requiring a rate of $9R_{\rm b}$ may be given a $16R_{\rm b}$ code. The wasted capacity may be approaching 100% as the SF decreases. This is referred to as the internal fragmentation problem. To relieve the internal fragmentation problem and better utilize the scarce wireless bandwidth, one possibility is to use multiple (smaller) OVSF codes to support a call. For example, a call requesting rate $9R_{\rm b}$ can be supported by a $1R_{\rm b}$ code and an $8R_{\rm b}$ code. This direction has been explored in [6], [7], and [15]; however, not much has been addressed regarding what is the best number of OVSF codes to be used. In this paper, we conduct some analysis to address the tradeoff between hardware complexity and code efficiency under such an environment.

In addition to the internal fragmentation problem, while connections are arriving and leaving the system, an OVSF code tree may become too fragmented to support newly arrived calls even if there are sufficient spaces in the code tree. This is referred to as the external fragmentation problem. Solutions to this problem require intelligent code assignment and code reassignment strategies. The former addresses how to assign code(s) to a new call in the code tree to avoid the tree becoming too fragmented. The latter addresses how to relocate code(s) when a new call arrives finding no proper place to accommodate

Manuscript received December 13, 2002; revised July 28, 2003; accepted May 2, 2004. The editor coordinating the review of this paper and approving it for publication is Y. Fang. The work of Y.-C. Tseng was supported by the MOE Program for Promoting Academic Excellence of Universities under Grant 89-E-FA04-1-4, by the National Science Council of Taiwan, R.O.C. under Grant NSC92-2213-E009-076 and Grant NSC92- 2219-E009-013, by the Institute for Information Industry and MOEA, R.O.C. under the Handheld Device Embedded System Software Technology Development Project, and by the Lee and MTI Center of NCTU.

C.-M. Chao is with the Department of Information Management, Tamkang University, Taipei 25165, Taiwan, R.O.C. (e-mail: cmchao@axp1.csie.ncu.edu.tw).

Y.-C. Tseng is with the Department of Computer Science and Information Engineering, National Chiao-Tung University, Hsin-Chu 30050, Taiwan, R.O.C. (e-mail: yctseng@csie.nctu.edu.tw).

L.-C. Wang is with the Department of Communication Engineering, National Chiao-Tung University, Hsin-Chu 30050, Taiwan, R.O.C. (e-mail: lichun@cc.nctu.edu.tw).

Digital Object Identifier 10.1109/TWC.2005.850332

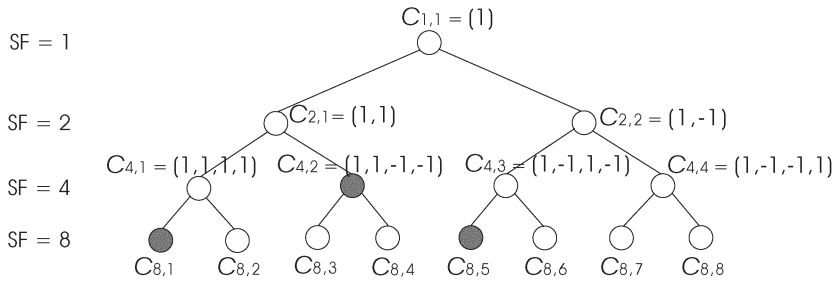

Fig. 1. A code blocking scenario (busy codes are marked by gray).

it but the remaining capacity in the system is sufficient [13]. Code reassignment involves costs, which should be minimized. This is very similar to the traditional memory management problem in operating system design [16]. All these may have significant impact on bandwidth utilization and call blocking probability.

The code assignment and reassignment problems in WCDMA systems have been widely studied [4], [5], [17], [18]. Most of these strategies are under a single-code-per-user assumption. In a multicode-per-user environment, more considerations need to be addressed, including the number of codes to be assigned to the user, the ordering of code assignment, the colocation of codes, and the assignment of individual codes. This paper addresses code assignment and reassignment issues together where multi-OVSF codes can be assigned to a user. The general objective is to make the OVSF code tree as compact as possible in order to support more new calls, either with less blocking or with less reassignment cost. We propose several multicode assignment and reassignment schemes, namely, random, leftmost, crowded-first-space, and crowded-first-code. The random scheme is used here for comparison reasons. The leftmost scheme is similar to the first-fit memory management algorithm [16], which stops searching when the first big enough space is found. The crowded-first-space scheme is similar to the best fit algorithm [16], where a space that is larger than but closest to the request will be allocated. However, when ties occur, the consideration in an OVSF code tree is quite different from that in a memory space. Another memory allocation algorithm, the buddy systems [16], is also similar to our OVSF environment in that the provided capacities are always powers of 2. It is similar to the best fit algorithm and has similar problems in handling ties. Simulations have been conducted to evaluate the performance of the proposed schemes. The results justify that the crowded-first-space and the crowded-first-code schemes can provide more compact code trees, thus significantly reducing the code blocking probability and code reassignment cost. For example, by allowing at most two codes for each call, these schemes can reduce the code reassignment costs by 82 and 83%, respectively, as opposed to the random scheme, when the system is 80% loaded with a $\text{maximum spreading factor} = 256$.

The rest of the paper is organized as follows. In Section II, we introduce the generation of orthogonal Walsh codes and motivate our work by observing the internal and external fragmentation problems in a code tree. Section III analyzes the internal fragmentation problem. The proposed multicode assignment and reassignment schemes are presented in Section IV, which is followed by simulation results in Section V. Concluding remarks are given in Section VI.

II. Backgrounds and Motivations

In an OVSF-CDMA system, each base station manages a code tree for downlink transmission. The resource units in an OVSF code tree are codes. Therefore, base stations are responsible for utilizing their code trees efficiently to increase the performance of the system. In this section, we will introduce the background of OVSF code trees and motivate this paper by defining the internal and external fragmentation problems.

A. OVSF Code Tree

OVSF codes are used as channelization codes in the UMTS terrestrial radio access network. These OVSF codes can be represented by a code tree as shown in Fig. 1. Each OVSF code can be denoted as $C_{\mathrm{SF},k}$, where k is the branch number $1 \leq k \leq \mathrm{SF}$. The root code is $C_{1,1} = (1)$. The codes at the second level are $C_{2,1} = (1,1)$ and $C_{2,2} = (1,-1)$. The codes at the kth level are defined by spawning two codes (C, C) and $(C, \overline{C})$ from each code (C) at the $(k-1)$th level, where $\overline{C}$ is the bitwise complement of C. The number of levels is determined by the maximum SF.

The number of codes at each level is equal to the value of SF. All codes in the same level are orthogonal while codes in different levels are orthogonal if they do not have an ancestor–descendant relationship. The data rate is halved whenever we go one level down the tree. Leaf codes have the minimum data rate, which is called the basic rate R_{b}, of the system. For example, in Fig. 1, $C_{8,1}$ has the rate of $1R_{\mathrm{b}}$, while $C_{4,1}$ and $C_{2,1}$ have rates of $2R_{\mathrm{b}}$ and $4R_{\mathrm{b}}$, respectively.

B. Internal Fragmentation and External Fragmentation

Internal fragmentation occurs when the allocated data rate to a call is larger than what is requested. The reason is that the allocatable code rate is always a power of 2 with respect to the basic rate R_{b}. For example, we must assign $128R_{\mathrm{b}}$ to a request for rate $67R_{\mathrm{b}}$ if only one code can be assigned. There is a waste of $61/67 \approx 91\%$ in bandwidth because of internal fragmentation. Such deficiency can be alleviated by assigning multiple codes to a request. For example, the waste can be reduced to $1/67 \approx 1.5\%$ if we can assign two codes $(64R_{\mathrm{b}} + 4R_{\mathrm{b}})$ to the request.

External fragmentation occurs when the code tree has a number of low-rate codes such that calls requesting for higher

rates can easily get rejected even if there is still sufficient capacity remaining in the code tree. Fig. 1 shows an example where codes $C_{4,2}$, $C_{8,1}$, and $C_{8,5}$ are occupied. The remaining capacity is $4R_{\rm b}$. But a new call requesting a rate of $4R_{\rm b}$ will be rejected because there is no such code available. We call such a situation code blocking. Code blocking may reduce the system's utilization. One possible solution to external fragmentation is to allocate codes more carefully when requests arrive, to which we refer as the code assignment problem. The other remedy is to conduct code replacement by moving occupied codes around, to which we refer as the code reassignment problem. In the earlier scenario, we can vacate code $C_{2,2}$ by moving $C_{8,5}$ to $C_{8,2}$. However, if three codes are allowed to support a connection, the above relocation is even unnecessary. One of the purposes of this paper is to consider the above assignment and reassignment problems in a system allowing multiple codes.

III. Reducing Internal Fragmentation by Multiple Codes

Internal fragmentation occurs when a user receives more than the needed resource. Using multiple codes can more closely match the users' requirements. However, more codes incur higher hardware cost. This section analyzes the relationship between the internal fragmentation and the allowable number of codes.

In the analysis, we assume that calls may request for any rate between $1R_{\rm b}$ and $mR_{\rm b}$, where $m = 2^k$ for some integer k. The actual arrival rate for $iR_{\rm b}$ is assumed to be λ_i, $i = 1 \ldots m$ (i.e., excluding those that are rejected). Thus, the actual demand is $Q = \sum_{i=1}^{m} \lambda_i \times iR_{\rm b}$. Our goal is to derive the amount of capacity loss due to internal fragmentation, assuming that at most n codes can be used by each request. Given any request $iR_{\rm b}$, let $f^n(i)$ be the minimal rate that can be assigned to the request. For example, one can easily derive that $f^1(4) = 4R_{\rm b}$, $f^1(6) = 8R_{\rm b}$, $f^2(6) = 6R_{\rm b}$, $f^2(7) = 8R_{\rm b}$, $f^3(7) = 7R_{\rm b}$, and $f^3(15) = 16R_{\rm b}$. Let $N(i)$ be the number of 1s in i's binary representation (for example, $N(4) = N(100_2) = 1$ and $N(7) = N(111_2) = 3$). The function $f^n(i)$ can be derived recursively as

$$f^n(i) = \begin{cases} 2^{\lceil \lg i \rceil} R_{\rm b}, & n = 1 \\ iR_{\rm b}, & n \geq 2, n \geq N(i) \\ 2^{\lfloor \lg i \rfloor} R_{\rm b} + f^{n-1}\left(i - 2^{\lfloor \lg i \rfloor}\right) R_{\rm b}, & \text{otherwise.} \end{cases}$$

For example, when $n = 1$, only one code is available for each request, so $f^1(4) = 2^{\lceil \lg 4 \rceil} = 4R_{\rm b}$ and $f^1(6) = 2^{\lceil \lg 6 \rceil} = 8R_{\rm b}$. With more than one code, we can translate i into a binary number and look at the number of 1s in it, i.e., $N(i)$. If $n \geq N(i)$, each binary 1 corresponds to one code, so $N(i)$ codes are sufficient and the request is assigned a capacity perfectly matching the demand. For example, $f^3(6) = f^3(110_2) = 6R_{\rm b}$ and $f^3(13) = f^3(1101_2) = 13R_{\rm b}$. If $n < N(i)$, we assign one code to serve the leading binary 1 and recursively call f^{n-1} to resolve the remaining demand of rate $i - 2^{\lfloor \lg i \rfloor}$ by using the remaining $n-1$ codes. For example, $f^2(13) = f^2(1101_2) = 2^3 + f^1(13 - 2^3) = 2^3 + 2^3 = 16R_{\rm b}$ and $f^3(23) = f^3(10111_2) = 2^4 + f^2(23 - 2^4) = 2^4 + 2^2 + f^1(23 - 2^4 - 2^2) = 2^4 + 2^2 + 2^2 = 24R_{\rm b}$. Intuitively, we use the first $n-1$ codes to support the leading $n-1$ binary 1s and the last code to cover the remaining capacity. Note that the number of codes to be used may be less than n in the case of $n < N(i)$. For example, in the previous example, $f^3(23) = (2^4 + 2^2 + 2^2)R_{\rm b} = (2^4 + 2^3)R_{\rm b}$, so either three or two codes can be used. In the rest of this paper, given any request $iR_{\rm b}$, we will assume that $n' = \min\{n, N(i)\}$ codes will be assigned to the request.

Theorem 1: Given any request $iR_{\rm b}$, suppose that at most n codes can be used. Then the minimum capacity that can match the request is $f^n(i)$.

Proof: When $n \geq N(i)$, it is sufficient to use $N(i)$ codes to perfectly match the demand, so $f^n(i) = iR_{\rm b}$ and the theorem holds trivially. It remains to prove the theorem for $n < N(i)$. When $n < N(i)$, the request $iR_{\rm b}$ has to be overserved. Specifically, at least $N(i) - n + 1$ binary 1s of i will be overserved. The amount of overserved capacity produced by $f^n(i)$ is minimal since the leading $n-1$ binary 1s have been covered, which means that the overserved capacity is minimal. ■

Thus, given n assignable codes, the actual capacity that is assigned to users is $A_n = \sum_{i=1}^{m} \lambda_i \times f^n(i)$. The average amount of internal fragmentation when using n codes can be written as $A_n - Q$. Since A_n is the actual capacity allocated to users, the value $I_n = (A_n - Q)/A_n$ can be regarded as the internal fragmentation ratio. We will demonstrate some simulation results based on the above formulation in Section V.

IV. Reducing External Fragmentation by Multicode Assignment and Reassignment

In the previous section, we only consider capacity loss due to internal fragmentation when using multiple codes. When placing individual codes on the code tree, their locations in the tree also have impact on the utilization of the system. The term external fragmentation refers to the situation when the code tree is too fragmented to accept new requests even if there is sufficient remaining capacity in the code tree. The purpose of this section is to propose code assignment and reassignment strategies to alleviate the external fragmentation problem when multiple codes need to be allocated for a request.

A. Multicode Assignment

Given a call request $iR_{\rm b}$, assuming that at most n codes can be used, our goal is to allocate sufficient capacity for the request from the code tree. From earlier derivation, we would allocate $f^n(i)$ for the request. Let $2^{k_1}R_{\rm b}, 2^{k_2}R_{\rm b}, \ldots, 2^{k_{n'}}R_{\rm b}$ be the n' codes calculated by $f^n(i)$, $n' \leq n$. Without loss of generality, let $k_1 > k_2 > \cdots \geq k_{n'}$. There are three dimensions when considering the code allocation problem.

- Ordering of assignment: This refers to which of the n' codes should be allocated first and which later. We propose two strategies here: increasing and decreasing. By the increasing strategy, codes with lower rates are assigned earlier than those with higher rates. By the decreasing strategy, codes with higher rates are assigned first.

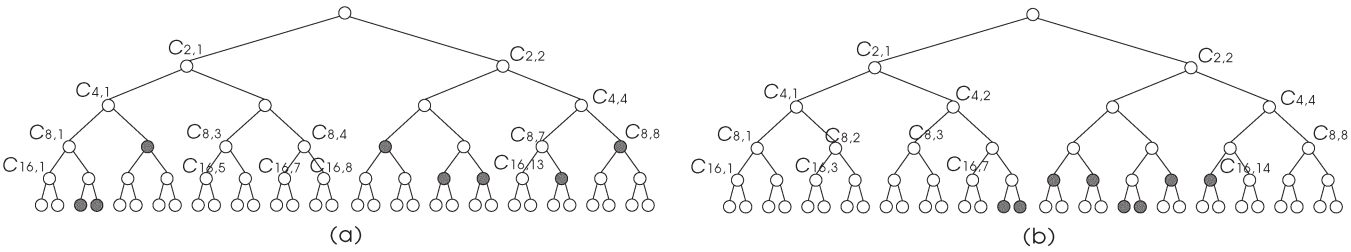

Fig. 2. Code assignment examples ($\max \mathrm{SF} = 32$).

- Colocation of codes: This refers to the relative locations of the n' codes in the code tree. Here we propose two strategies. By the united strategy, we will try to find a sufficiently large subtree that can accommodate all the n' codes simultaneously. By the separated strategy, each of the n' codes is assigned independently. Note that by the former strategy, when the request is released, the code tree will have a higher chance of having larger codes than the latter strategy; this is because the codes are closer to each other in the code tree.
- Assignment of individual codes: When considering the assignment of each single code, there may exist multiple candidate codes that can satisfy the request. We propose four strategies below. Let $2^{k_j} R_{\mathrm{b}}$, $j \leq n'$, be the request under consideration.
  1) Random: We randomly pick any of the candidate codes that have the rate $2^{k_j} R_{\mathrm{b}}$ and assign it to the request.
  2) Leftmost: When there is more than one candidate code with rate $2^{k_j} R_{\mathrm{b}}$, we pick the leftmost one in the code tree and assign it to the call. The intuition here is to always vacate a larger capacity in the right-hand side of the code tree (to avoid the code tree becoming too fragmented), which can be used to accommodate higher-rate calls in the future.
  3) Crowded-first-space: We pick the candidate whose ancestor code has the least free capacity. Specifically, suppose codes x and x' are both candidate codes. We compare the free capacities of their ancestors, say y and y', respectively. The one with less free capacity (i.e., more crowded) will be chosen for use. Whenever there are ties, we will go one level up by comparing the ancestor codes of y and y' and choose the more crowded one. This is repeated until the most crowded subtree is identified. One special case is $y = y'$ (i.e., we trace to the same subtree). If so, we follow the leftmost strategy to pick the code on the left-hand side. Intuitively, this strategy also tries to place busy codes (and thus free codes) together.
  4) Crowded-first-code: This strategy is similar to the previous one except that when comparing candidates we will adopt the numbers of occupied codes (instead of space) in the subtrees rooted by y and y' as the comparison metric. Still, crowded ones are chosen first. The intuition is we regard a code (instead of its rate) as a unit, since all its capacity will be released at the same time when the request leaves.

Below, we use an example to explain the assignment of one code when $n = 2$. Consider the code tree in Fig. 2(a). Suppose that a new call arriving is requesting a rate of $2R_{\mathrm{b}}$. Then one code can support this request (i.e., $n' = 1$). There are six candidates with rate $2R_{\mathrm{b}}$: $C_{16,1}$, $C_{16,5}$, $C_{16,6}$, $C_{16,7}$, $C_{16,8}$, and $C_{16,13}$. By the random strategy, any of the candidates may be picked to serve the new call. By the leftmost strategy, $C_{16,1}$ will be chosen. By the crowded-first-space strategy, we will go one level up to check these codes' ancestors. Among them, both $C_{8,1}$ and $C_{8,7}$ will be fully occupied if $C_{16,1}$ and $C_{16,13}$ are chosen, respectively. So we further compare their ancestors, $C_{4,1}$ and $C_{4,4}$, but again this is a tie. After further going one level up, we find $C_{2,2}$ has less capacity left than $C_{2,1}$ does, so $C_{16,13}$ will be selected to serve the new call. By the crowded-first-code strategy, after going one level up, we find that the ancestors $C_{8,1}$, $C_{8,3}$, $C_{8,4}$, and $C_{8,7}$ have 2, 0, 0, and 1 occupied codes, respectively. By our definition, $C_{8,1}$ is most crowded, so $C_{16,1}$ is chosen to serve the new call.

Next, we consider a multicode assignment example based on the code tree in Fig. 2(b). Suppose that a new call needing a rate of $6R_{\mathrm{b}}$ arrives. Two codes, $4R_{\mathrm{b}}$ and $2R_{\mathrm{b}}$, are needed to support this request. Considering the colocation of codes, suppose we decide to adopt the separated strategy. Then these two codes will be allocated independently. If the decreasing and the leftmost strategies are adopted, $C_{8,1}$ and $C_{16,3}$ will be allocated. On the contrary, if the increasing strategy is adopted, then $C_{16,1}$ and $C_{8,2}$ will be allocated. By the crowded-first-space strategy, $\{C_{8,8}, C_{16,14}\}$ and $\{C_{16,14}, C_{8,8}\}$ will be allocated when the decreasing and increasing strategies are adopted, respectively. By the crowded-first-code strategy, $\{C_{8,3}, C_{16,7}\}$ and $\{C_{16,7}, C_{8,3}\}$ will be allocated when the decreasing and increasing strategies are adopted, respectively. If the united strategy is adopted, only codes $C_{4,1}$, $C_{4,2}$, and $C_{4,4}$, which have free capacity larger than $6R_{\mathrm{b}}$, will be considered. However, if the increasing strategy is adopted, $C_{4,2}$ will not be a candidate since the only free $4R_{\mathrm{b}}$ code $C_{8,3}$ will be fragmented by the $2R_{\mathrm{b}}$ subrequest, thus blocking the later $4R_{\mathrm{b}}$ subrequest. By the leftmost strategy, $\{C_{8,1}, C_{16,3}\}$ will be allocated to serve the new call for both increasing and decreasing strategies. By the crowded-first-space strategy, $\{C_{8,8}, C_{16,14}\}$ will be allocated (for both increasing and decreasing strategies) since $C_{2,2}$ has less capacity left than $C_{2,1}$. By the crowded-first-code strategy, if decreasing strategy is applied, $C_{8,3}$ and $C_{16,7}$ will be allocated because $C_{4,2}$ has more occupied codes than $C_{4,4}$. If increasing strategy is adopted, $C_{16,14}$ and $C_{8,8}$ will be allocated to the new call.

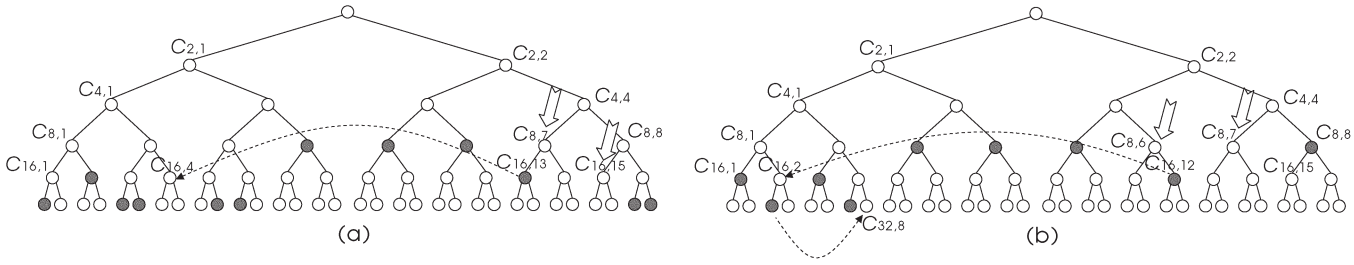

Fig. 3. Code reassignment examples ($\max \mathrm{SF} = 32$).

B. Code Reassignment

When a code tree is used for long time, it is sometimes inevitable that the tree may become fragmented. In this case, a new call requesting for a rate iR_{b} may be rejected even if the total amount of free capacity in the code tree is $\geq f^n(i)$. This is the external fragmentation problem and may result in low utilization of the code tree.

To resolve this problem, code reassignment can be conducted to move current codes around so as to squeeze a large-enough space for the request. In [13], a dynamic code assignment (DCA) scheme was proposed to solve the single-code reassignment problem. Given a fragmented code tree and a code request (of rate in the power of 2), the DCA scheme can determine a subtree that incurs the least reassignment cost, where the cost is defined to be the number of reassignments to vacate the identified subtree.

In this paper, we consider multicode reassignment. Our scheme utilizes the DCA scheme as a basic construction block. However, when moving codes around, we also consider where to place those codes that are migrated so as to reduce the potential future reassignment cost (this issue is ignored in [13]). Note that for consistency, the placement of migrated codes should follow the same code assignment strategies as described in Section IV-A. Also note that only separated reassignment of migrated codes is applicable since DCA only vacates one target at a time. The integrated solution is outlined as follows.

1) Given a request for rate iR_{b}, first run the code assignment scheme as described in Section IV-A. If this succeeds, exit the procedure. Otherwise, check whether the remaining capacity of the code tree is $\geq f^n(i)$ or not. If so, go to the next step; otherwise, reject the request and exit.
2) Run the code assignment again and assign codes one-by-one depending on the chosen code assignment strategies. However, in this time, whenever we fail to find a location to accommodate any rate $2^{k_j}R_{\mathrm{b}}$, $j \leq n'$, the DCA scheme is initiated to determine a subtree, say T, of rate $2^{k_j}R_{\mathrm{b}}$ (this step is guaranteed to succeed by [13]). Then for occupied codes in T, we relocate them one-by-one either in decreasing or increasing strategy. The replacement of each code is regarded as a code assignment process and should follow the strategies chosen in Section IV-A to perform the relocation (for example, if the leftmost strategy is chosen, then the replacement should search locations following the same rule). This is repeated until all busy codes in T are migrated. Note that when relocating each code in T, it is possible that no free code exists to accommodate this reallocated code. If so, another round of DCA (and thus another round of relocation) will be triggered recursively. This relocation process is guaranteed to complete since the total free capacity is sufficiently large.

For example, consider the code tree in Fig. 3(a). Suppose that $n = 2$ and a new call requests for a rate of $6R_{\mathrm{b}}$. The request is split into two subrequests $4R_{\mathrm{b}}$ and $2R_{\mathrm{b}}$ and here we assume a decreasing strategy. The total free capacity is $9R_{\mathrm{b}}$, but there is no space to fit $4R_{\mathrm{b}}$ and thus reassignment is required. In this example, code $C_{8,7}$ will be chosen by DCA as the minimum-cost branch to be vacated. By the leftmost strategy, the busy code $C_{16,13}$ will be moved to $C_{16,4}$, after which $C_{8,7}$ is free to accommodate the $4R_{\mathrm{b}}$ subrequest. The subrequest $2R_{\mathrm{b}}$ will be placed in $C_{16,15}$ directly without reassignment.

A more complex example is in Fig. 3(b), which has a free capacity of $8R_{\mathrm{b}}$. Still, let $n = 2$. A new call requests for a rate of $7R_{\mathrm{b}}$, which is split into two $4R_{\mathrm{b}}$ subrequests. The first subrequest can be placed in $C_{8,7}$. For the second subrequest, the DCA will choose $C_{8,6}$ to be vacated. When relocating the first busy code $C_{16,12}$, the DCA will be triggered again, which will identify $C_{16,2}$ as minimum-cost subtree. Then the busy code $C_{32,3}$ will be moved to $C_{32,8}$. Now $C_{16,12}$ can be reassigned to $C_{16,2}$ and $C_{8,6}$ is vacated to support the second subrequest.

V. Simulation Results

First, we demonstrate some simulation results about the internal fragmentation ratio derived in Section III. We consider three types of traffics (m is the maximum request rate).

- More small calls: the arrival rate for each of the request capacities $1R_{\mathrm{b}} \ldots (m/2)R_{\mathrm{b}}$ is λ, while that for each of the remaining request capacities is $(\lambda/4)$.
- Uniform: the arrival rates for all kinds of requests are the same.
- More large calls: the arrival rate for each of the request capacities $1R_{\mathrm{b}} \ldots (m/2)R_{\mathrm{b}}$ is $(\lambda/4)$, while that for each of the remaining request capacities is λ.

Tables I–III show the internal fragmentation ratios for these traffic types under different values of n and m. As we can see, the ratios normally improve significantly when n is increased from 1 to 2. Less significant improvement can be obtained when increasing n to 3, and after $n \geq 4$ there is very little benefit. Since this reflects the hardware complexity, an n of 2 or 3 will be quite cost effective. Also note that since we only consider

TABLE I
INTERNAL FRAGMENTATION RATIO UNDER THE "MORE SMALL CALLS" TRAFFIC PATTERN

n	$m=8$	$m=16$	$m=32$	$m=64$	$m=128$	$m=256$
1	0.1316	0.1758	0.2076	0.2272	0.2381	0.2439
2	0.015	0.0374	0.0598	0.0765	0.0871	0.0932
3	0	0.0033	0.0104	0.0185	0.0251	0.0296
4	0	0	0.0008	0.0029	0.0055	0.0078
5	0	0	0	0.0002	0.0008	0.0016
6	0	0	0	0	0.0001	0.0003
7	0	0	0	0	0	0.0001
8	0	0	0	0	0	0

TABLE II
INTERNAL FRAGMENTATION RATIO UNDER THE "UNIFORM" TRAFFIC PATTERN

n	$m=8$	$m=16$	$m=32$	$m=64$	$m=128$	$m=256$
1	0.1628	0.1963	0.2197	0.2338	0.2416	0.2457
2	0.0271	0.0498	0.0692	0.0826	0.0906	0.0951
3	0	0.0058	0.0141	0.0219	0.0275	0.0312
4	0	0	0.0015	0.004	0.0066	0.0087
5	0	0	0	0.0004	0.0011	0.002
6	0	0	0	0	0.0001	0.0003
7	0	0	0	0	0	0.0001
8	0	0	0	0	0	0

TABLE III
INTERNAL FRAGMENTATION RATIO UNDER THE "MORE LARGE CALLS" TRAFFIC PATTERN

n	$m=8$	$m=16$	$m=32$	$m=64$	$m=128$	$m=256$
1	0.1799	0.2075	0.2262	0.2374	0.2435	0.2467
2	0.0339	0.0566	0.0743	0.0858	0.0925	0.0962
3	0	0.0072	0.0162	0.0237	0.0289	0.032
4	0	0	0.0018	0.0046	0.0072	0.0092
5	0	0	0	0.0005	0.0013	0.0022
6	0	0	0	0	0.0002	0.0004
7	0	0	0	0	0	0.0001
8	0	0	0	0	0	0

internal fragmentation, the absolute value of λ is irrelevant to the results.

We have also implemented a simulator to evaluate the performance of the proposed strategies. The maximum SF is set to 256. New calls arrive in a Poisson process. Calls request for rates between $1R_{\rm b}$ and $mR_{\rm b}$ with equal probability, where $m=2^k$ with $k=3$, 4, or 5. Call duration is exponentially distributed with a mean of four time units. Our policy is denoted by five letters $a_1a_2a_3a_4a_5$, where $a_1=i$ or d (standing for increasing or decreasing strategy), $a_2=s$ or u (separated or united strategy), $a_3=r$, l, s, or c (random, leftmost, crowded-first-space, or crowded-first-code strategy), $a_4=r$ if code reassignment is adopted, and $a_5=n$. Note that whenever the corresponding strategy is not available, the letter will be set to x.

In the following, we make observations from four aspects. Each point below is from an average of 100 simulation runs, where each run contains at least 4000 accepted calls.

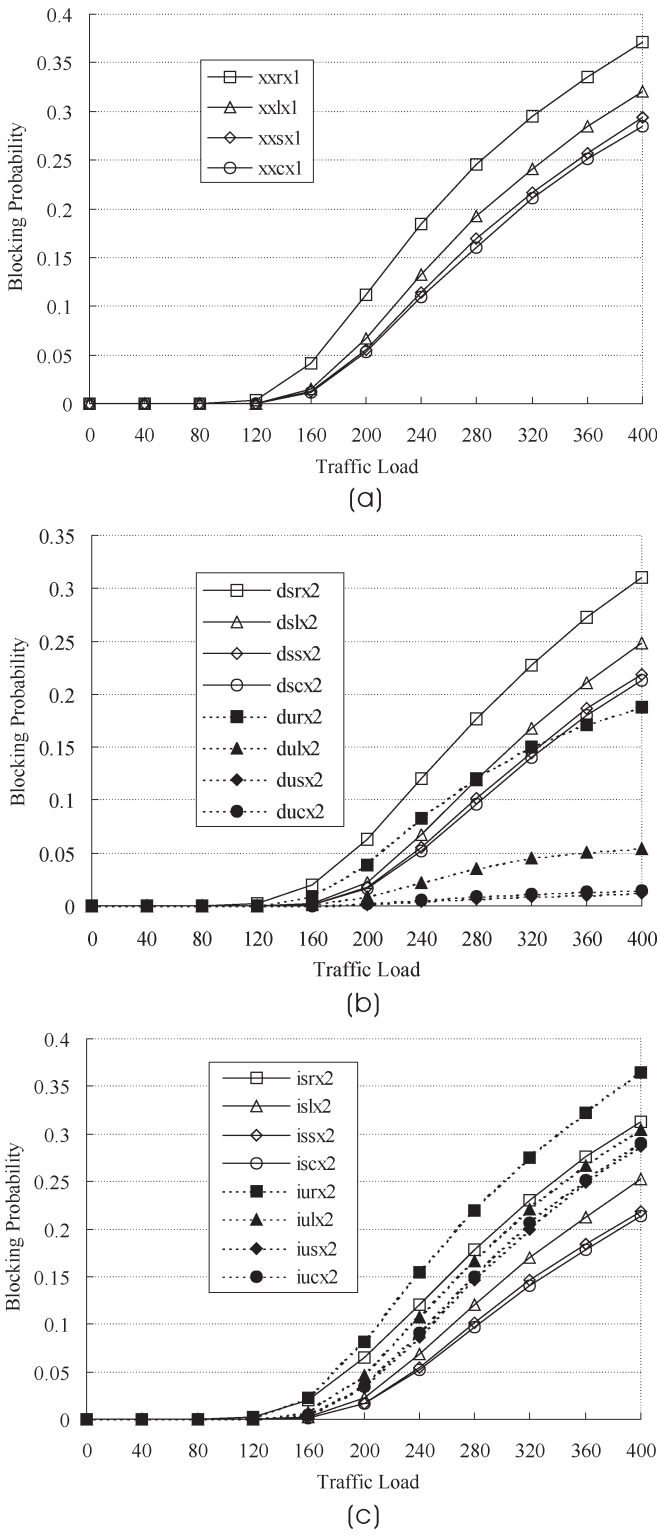

Fig. 4. Blocking probability versus traffic load: (a) $n=1$, (b) $n=2$ with the decreasing strategy, and (c) $n=2$ with the increasing strategy. $(m=8)$.

A. Impact of Assignment Strategies

This experiment tests different code assignment strategies without using code reassignment. We evaluate the code blocking probability, which is defined as a new call being rejected because of no code available but the total amount of free capacity in the code tree is sufficiently large. Fig. 4 shows the results. Fig. 4(a) stands for the case when only one code is used for each call. We see that the crowded-first-code performs the best, which is followed in turn by the crowded-first-space, leftmost, and then random strategies. Code assignment strategy will have significant impact on blocking probability after the system is

60% fully loaded (at around load $= 160$). For example, when the code tree is about 80% fully loaded (at around load $= 200$), the blocking probabilities are 11.2%, 6.7%, 5.5%, and 5.3% for these four strategies, respectively.

Fig. 4(b) shows the same simulation when $n = 2$. Since more than one code is available for each request, our multicode assignment strategies can be adopted. Here the decreasing strategy is adopted. Interestingly, we find that the group of schemes following the united strategy outperforms all the other strategies. This is because codes belonging to the same request always leave the code tree at the same time, thus leaving more space for future larger codes. Among all, the crowded-first-space is the best, which is seconded by the crowded-first-code strategy.

Fig. 4(c) shows the results when $n = 2$ and the increasing strategy is adopted. Contrary to our earlier observation, the *issx2* and *iscx2* schemes, which adopt the separated strategy, perform the best. We believe that this is because lower-rate codes are assigned first, making it easier to occupy larger codes instead of perfect-matching ones.

To view the comparison from a different angle, we compare the effective utilization of the code tree in Fig. 5, where the effective utilization is defined as the average capacity of the code tree that is actually used. For example, for $n = 1$, a request of $5R_{\rm b}$ will be assigned to an $8R_{\rm b}$ code. So only 5/8 of the capacity is actually utilized, i.e., both internal and external fragmentations are taken into account in this metric. Note that in the figure, we also show the utilization of schemes *xxsr1*, *dssr2*, and *issr2*, for which code reassignment is adopted. In such cases, external fragmentation is completely removed, so the maximum utilization is always achieved, and this represents a reference of utilization upper bound. Consistent with the earlier observations, the crowded-first-code and crowded-first-space strategies are the best. In Fig. 5(a), at load $= 400$, both *xxsx1* and *xxcx1* achieve 95.5% of the utilization upper bound while only 94.3 and 88.5% are obtained for *xxlx1* and *xxrx1*, respectively. The same simulation with $n = 2$ is shown in Fig. 5(b). When compared with Fig. 5(a), we can see that the utilization is increased. For example, at load $= 400$, the increased utilization is 10.7, 10.7, 14.4, and 16.6% for *dusx2*, *ducx2*, *dulx2*, and *durx2*, respectively. This indicates the benefit of using multiple codes. Fig. 5(c) is the result for $n = 2$ with increasing strategy. Similar to an earlier experiment, *issx2* and *iscx2* have the best performance, but it is still not as good as decreasing strategies *dusx2* and *ducx2*.

To conclude, when comparing all of these figures, the decreasing and united strategies coupled with the crowded-first-code or crowded-first-space strategy (i.e., schemes *dusx2* and *ducx2*) are the best choices. The blocking probability can be kept below 2% even when the system is heavily loaded. And the utilization can reach 99.5% of the upper bound. This implies that these schemes can greatly relieve the external fragmentation problem.

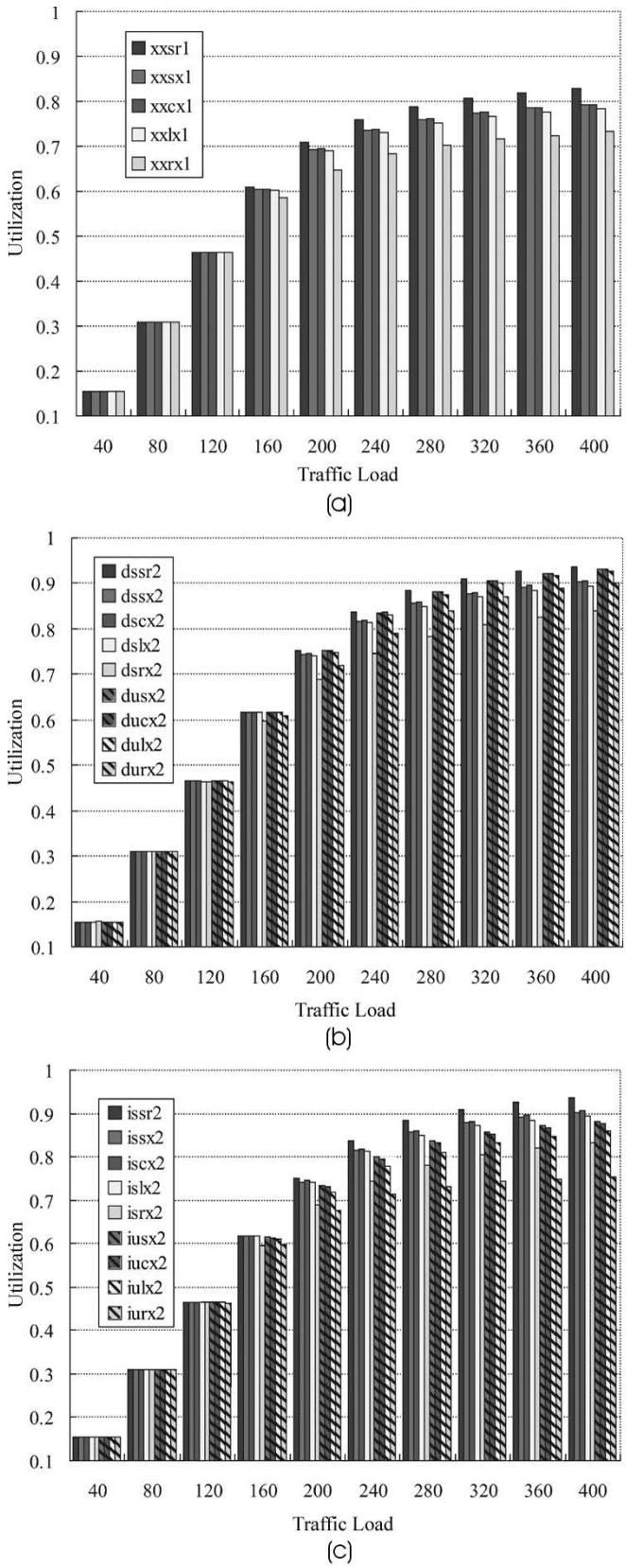

Fig. 5. Code tree utilization versus traffic load: (a) $n = 1$, (b) $n = 2$ with the decreasing strategy, and (c) $n = 2$ with the increasing strategy. $(m = 8)$.

B. Impact of Reassignment Strategies

Next, we further experiment on reassignment strategies. The performance metric is the number of reassignments being taken (comparing blocking probability makes no sense here because with reassignment any strategy can accept a new call whenever there is sufficient free capacity). Fig. 6(a) shows the result

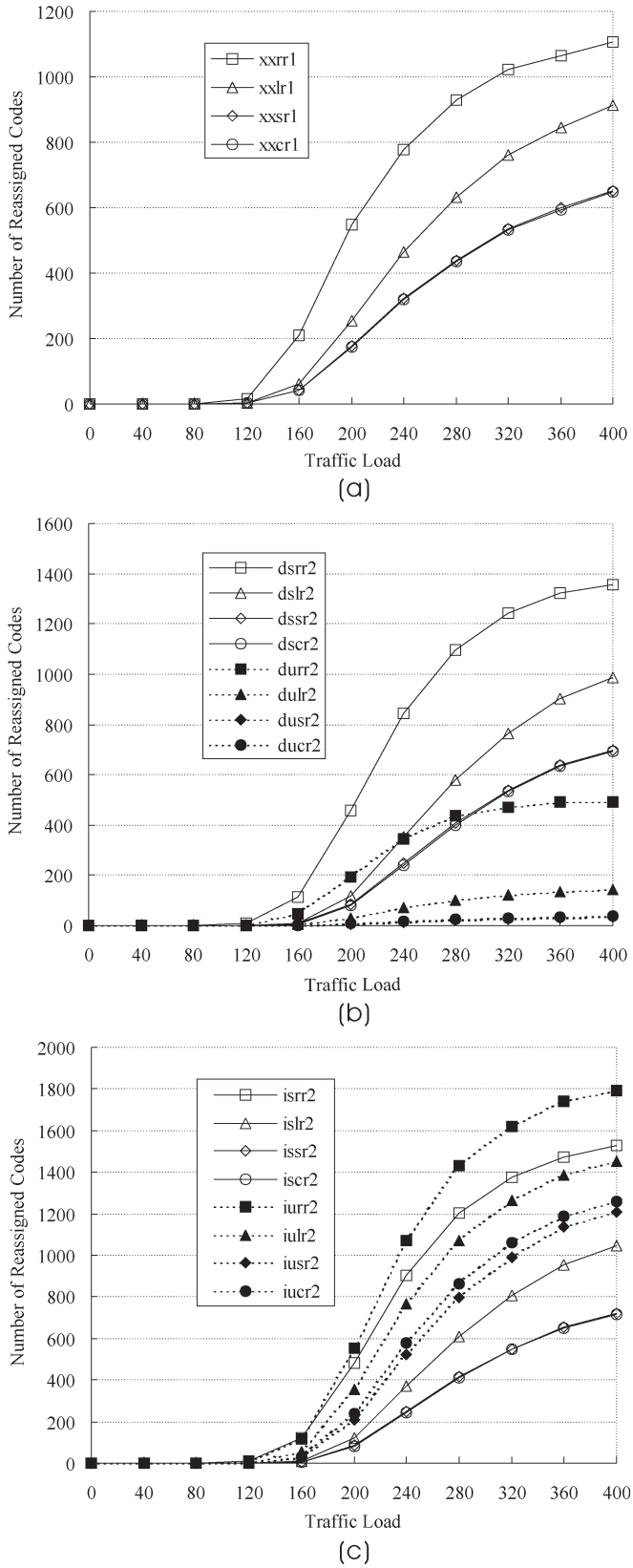

Fig. 6. Reassignment cost versus traffic load: (a) $n = 1$, (b) $n = 2$ with the decreasing strategy, and (c) $n = 2$ with the increasing strategy. $(m = 8)$.

when $n = 1$. This shows that the crowded-first-code and crowded-first-space strategies, while having the lowest blocking probabilities, also have the smallest migration costs. Code reassignment costs are increasing sharply after the code tree is around 50% fully loaded $(\text{load} = 120)$. For example, when the code tree is about 80% fully loaded $(\text{load} = 200)$, the reassignment costs are 549, 256, 174, and 174 for *xxrr1*, *xxlr1*, *xxsr1*, and *xxcr1*, respectively.

Fig. 6(b) shows the same simulation when $n = 2$ with the decreasing strategy. The migration costs when the united strategy is adopted tend to be lower than those when the separated strategy is adopted. Since the relative performance is similar to that in Figs. 4 and 5, we would suggest *dusr2* and *ducr2* as the best choices for use. Interestingly, the result also shows that these two schemes even have smaller migration costs than the case of $n = 1$ (note that $n = 2$ means that there are more busy smaller codes in the code tree since a request may be split). Thus, using multiple codes is very attractive to improve the performance of WCDMA systems. Fig. 6(c) shows the case when $n = 2$ with the increasing strategy. The *issr2* and *iscr2* schemes are the best but are still outperformed by earlier two *dusr2* and *ducr2* schemes.

We also compare the average number of relocations to accept an originally blocked call. When $\text{load} = 400$ and $n = 1$, there are 1.7, 1.65, 1.6, and 1.6 relocations for *xxrr1*, *xxlr1*, *xxsr1*, and *xxcr1*, respectively. When $\text{load} = 400$ and $n = 2$, the numbers of relocations are reduced to 1.18, 1.14, 1.05, and 1.05 for *durr2*, *dulr2*, *dusr2*, and *ducr2*, respectively. The average numbers of replacements per accepted new call are relatively small for all these four strategies. Again, the crowded-first-space and crowded-first-code schemes still perform the best.

C. Impact of Call Pattern

In earlier experiments, we assume that calls request for rates with equal probability. Here we test two more traffic patterns we defined in the beginning of this section: more small calls and more large calls. The effect on code blocking is shown in Fig. 7(a). Scheme *dusx2* performs the best in all three different call patterns, which is followed by *ducx2*. We see that the "more large calls" cases incur much larger blocking probabilities. Thus, adopting effective assignment strategies plays a more important role in such cases as opposed that in the "more small calls" cases. Fig. 7(b) shows the reassignment costs. The trend is generally similar.

D. Impact of n

Section III only analyzes the effect of internal fragmentation. In this experiment, we test different values of n, intending to observe the combined effect of both internal and external fragmentations. The maximal requested rate m is 8, 16, or $32R_{\text{b}}$. The combined decreasing, united, crowded-first-space strategy is adopted as the representing scheme in this comparison. Fig. 8(a) shows the results in terms of blocking probability. When comparing the related trends, we see significant improvement in the cases of $m = 8R_{\text{b}}$ and $16R_{\text{b}}$ when n is increased from 1 to 2 and in the case of $m = 32R_{\text{b}}$ when n is increased to 3. We believe that this is because with larger requests, there are more chances of having internal fragmentation. Thus, using

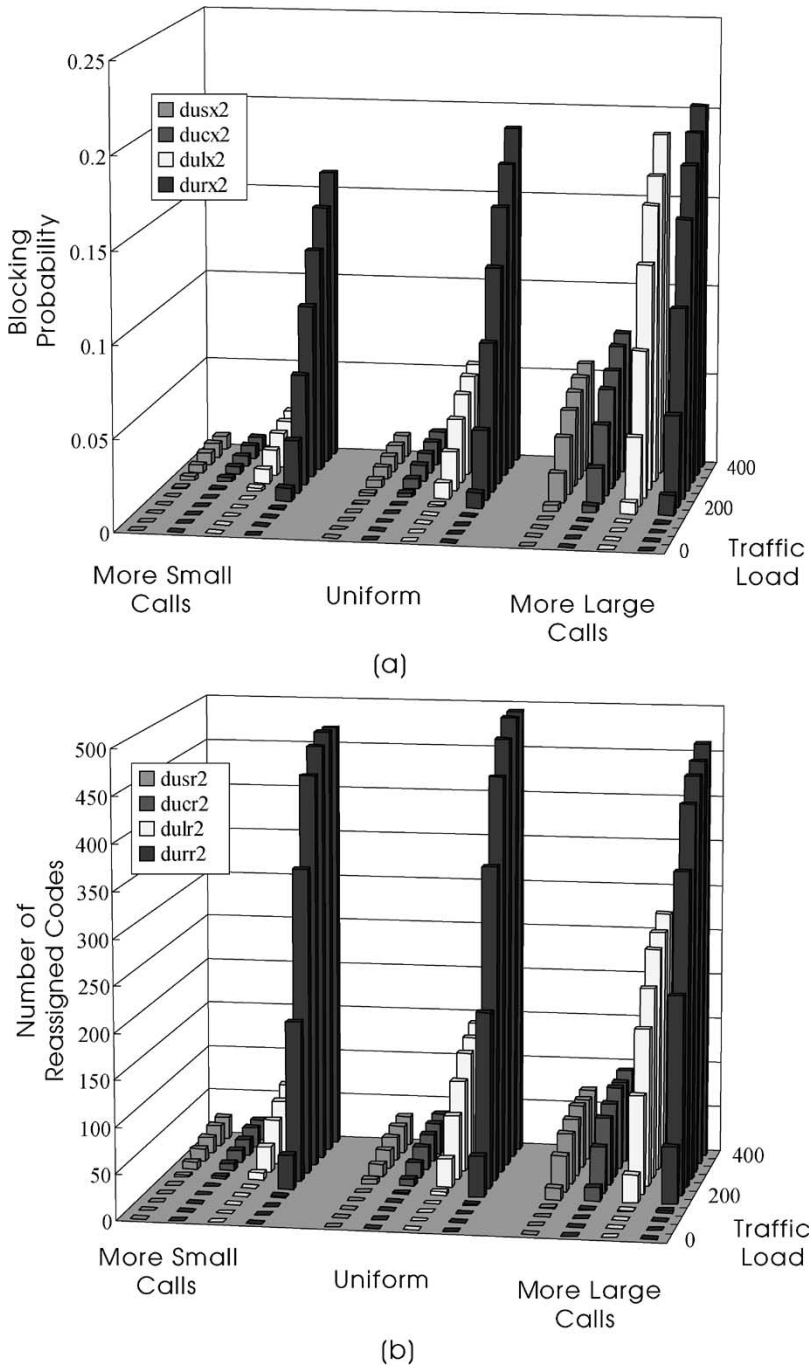

Fig. 7. The effects of call patterns on (a) blocking probability and (b) reassignment cost $(\mathrm{m} = 8)$.

more codes to support a request is more attractive in such cases. Fig. 8(b) shows the result in terms of migration costs. The trend is somewhat different: the numbers of reassignments all drop significantly when n is increased from 1 to 2.

VI. Conclusion

Due to the quick growth of PCS user populations, wireless bandwidth has become a critical resource to the success of next-generation wireless communication systems. In this paper, we have addressed both internal and external fragmentation problems when using WCDMA OVSF code trees. Based on the idea of using multiple codes to support a call request, several assignment and reassignment strategies are proposed and evaluated to relieve these problems. Simulation results show that the crowded-first-code or crowded-first-space strategy coupled with the united and decreasing strategies is the best choice among all in terms of blocking probability, code tree utilization, and code migration cost. Simulation results also suggest that using two or three codes will be sufficient to achieve high performance as well as effectively relieve both the internal and external fragmentation problems.

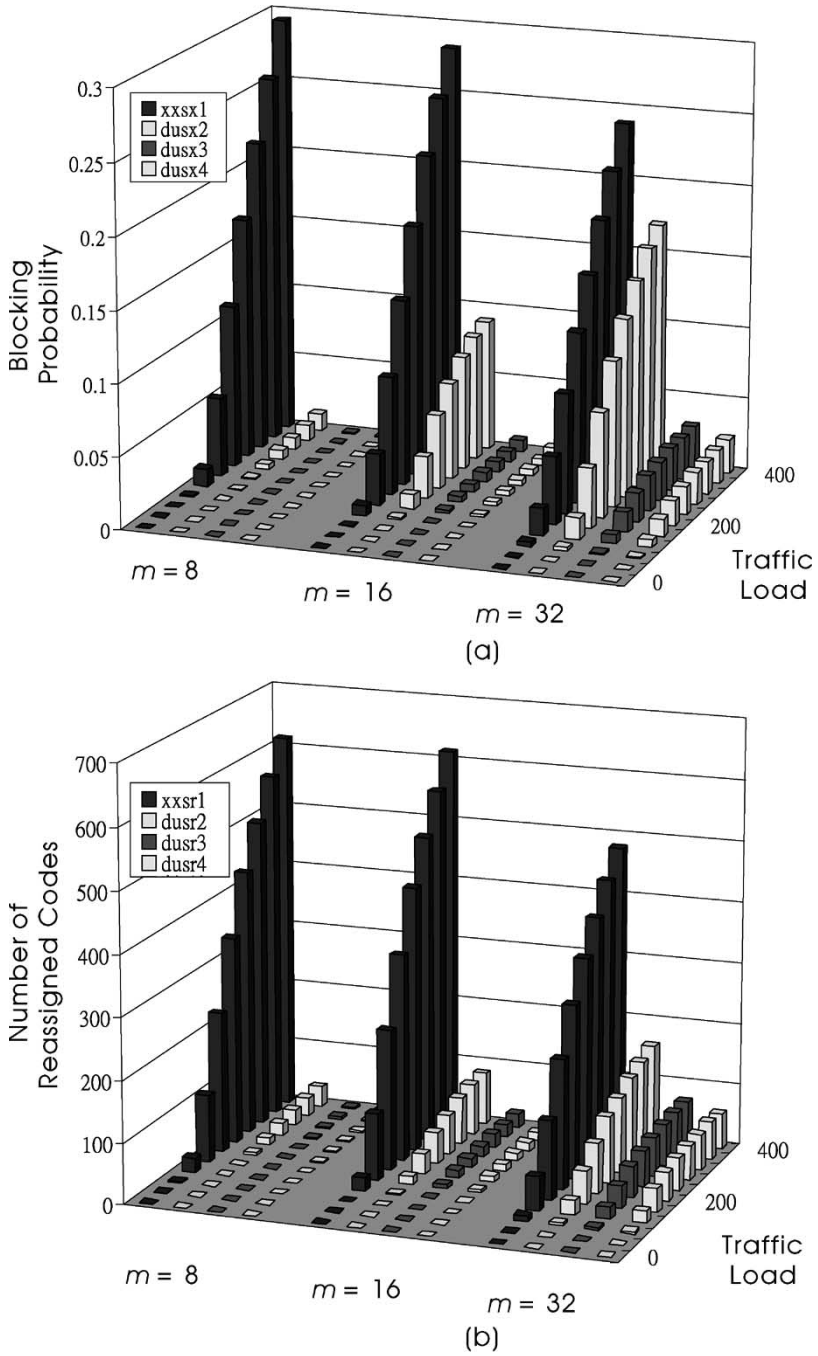

Fig. 8. The effects of n and m on (a) blocking probability and (b) reassignment cost.

REFERENCES

[1] (1999). "Spreading and modulation (FDD), Third Generation Partnership Project; Technical Specification Group Radio Access Network," [Online]. Available: http://www.3gpp.org

[2] 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," *Electron. Lett.*, vol. 33, no. 1, pp. 27–28, Jan. 1997.

[3] 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.

[4] R. Assarut, K. Kawanishi, U. Yamamoto, Y. Onozato, and M. Matsushita, "Region division assignment of orthogonal variable-spreading-factor codes in W-CDMA," in *Proc. IEEE Vehicular Technology Conf. (VTC)*, Atlantic City, NJ, 2001, vol. 3, pp. 1884–1888.

[5] W.-T. Chen, Y.-P. Wu, and H.-C. Hsiao, "A novel code assignment scheme for W-CDMA systems," in *Proc. IEEE Vehicular Technology Conf. (VTC)*, Atlantic City, NJ, 2001, vol. 2, pp. 1182–1186.

[6] R.-G. Cheng, "A code management mechanism for WCDMA mobile communication networks," in *Proc. Int. Workshop Mobile Communications*, Crete, Greece, 1999, pp. 348–353.

[7] R.-G. Cheng and P. Lin, "OVSF code channel assignment for IMT-2000," in *Proc. IEEE Vehicular Technology Conf. (VTC)*, Tokyo, Japan, 2001, vol. 3, pp. 2188–2192.

[8] E. Dahlman, B. Gudmundson, M. Nilsson, and J. Skold, "UMTS/IMT-2000 based on wideband CDMA," *IEEE Commun. Mag.*, vol. 36, no. 9, pp. 70–80, Sep. 1998.

[9] 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. Sel. Areas Commun.*, vol. 18, no. 8, pp. 1441–1454, Aug. 2000.

[10] V. K. Garg, *IS-95 CDMA and CDMA2000*. Upper Saddle River, NJ: Prentice-Hall, 2000.

[11] H. Holma and A. Toskala, *WCDMA for UMTS*. New York: Wiley, 2000.

[12] C.-L. I *et al.*, "IS-95 enhancements for multimedia services," *Bell Labs Tech. J.*, vol. 1, no. 2, pp. 60–87, Aug. 1996.
[13] 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.
[14] J. G. Proakis, *Digital Communications*. New York: McGraw-Hill, 1995.
[15] F. Shueh, Z.-E. P. Liu, and W.-S. E. Chen, "A fair, efficient, and exchangeable channelization code assignment scheme for IMT-2000," in *Proc. IEEE Int. Conf. Personal Wireless Communications (ICPWC)*, Hyderabad, India, 2000, pp. 429–433.
[16] A. S. Tanenbaum, *Modern Operating Systems*. Englewood Cliffs, NJ: Prentice-Hall, 1992.
[17] 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.
[18] Y. Yang and T.-S. P. Yum, "Nonrearrangeable compact assignment of orthogonal variable-spreading-factor codes for multi-rate traffic," in *IEEE Vehicular Technology Conf. (VTC)*, Atlantic City, NJ, 2001, vol. 2, pp. 938–942.

Chih-Min Chao received the B.S. degree in computer science from Fu Jen Catholic University, Taiwan, R.O.C., in 1992, the M.S. degree in computer science from National Tsing-Hua University, Taiwan, R.O.C., in 1996, and the Ph.D. degree in computer science and information engineering from National Central University, Taiwan, R.O.C., in 2004.

He was with SENAO International, Taiwan, R.O.C., in 1996. Since August 2004, he has been an Assistant Professor at the Department of Information Management, Tamkang University, Taiwan, R.O.C. His research interests include mobile computing and wireless communication, with a current focus on resource management in WCDMA systems.

Yu-Chee Tseng (S'91–M'95–SM'03) received the B.S. degree in computer science from National Taiwan University, Taiwan, R.O.C., in 1985, the M.S. degree in computer science from National Tsing-Hua University, Taiwan, R.O.C., in 1987, and the Ph.D. degree in computer and information science from The Ohio State University, Columbus, in 1994.

He was an Engineer with D-LINK Inc., Fountain Valley, CA, in 1990. He was an Associate Professor at Chung-Hua University, Taiwan, R.O.C., from 1994 to 1996 and at National Central University, Taiwan, R.O.C., from 1996 to 1999, and a Full Professor at National Central University from 1999 to 2000. Since 2000, he has been a Full Professor in the Department of Computer Science and Information Engineering, National Chiao-Tung University, Hsin-Chu Taiwan, R.O.C. His research interests include mobile computing, wireless communication, network security, and parallel and distributed computing.

Dr. Tseng served as a Program Chair for the Wireless Networks and Mobile Computing Workshop in 2000 and 2001, as a Vice Program Chair for the International Conference on Distributed Computing Systems in 2004, as a Vice Program Chair for the IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS) in 2004, as an Associate Editor for *The Computer Journal*, as a Guest Editor for *ACM Wireless Networks* Special Issue on "Advances in Mobile and Wireless Systems," as a Guest Editor for the IEEE TRANSACTIONS ON COMPUTERS Special Issue on "Wireless Internet," as a Guest Editor for Journal of Internet Technology special issue on "Wireless Internet: Applications and Systems," as a Guest Editor for *Wireless Communications and Mobile Computing* Special Issue on "Research in Ad Hoc Networking, Smart Sensing, and Pervasive Computing," as an Editor for the *Journal of Information Science and Engineering*, as a Guest Editor for the *Telecommunication Systems* Special Issue on "Wireless Sensor Networks," and as a Guest Editor for the *Journal of Information Science and Engineering* Special Issue on "Mobile Computing." He is a two-time recipient of the Outstanding Research Award, National Science Council, R.O.C., in 2001–2002 and 2003–2005, and a recipient of the Best Paper Award in the International Conference on Parallel Processing, 2003. Several of his papers have been chosen as Selected/Distinguished Papers in conferences and have been included for publication in journals. He has guided students to participate in several national programming contests and received several awards. Dr. Tseng is a Member of the Association for Computing Machinery (ACM).

Li-Chun Wang (S'92–M'96) received the B.S. degree from National Chiao-Tung University, Taiwan, R.O.C., in 1986, the M.S. degree from National Taiwan University, Taiwan, R.O.C., in 1988, and the M.Sc. and Ph.D. degrees in electrical engineering from Georgia Institute of Technology, Atlanta, in 1995 and 1996, respectively.

From 1990 to 1992, he was with the Telecommunications Laboratories of the Ministry of Transportations and Communications, Taiwan (currently the Telecom Labs of Chunghwa Telecom Company). In 1995, he was affiliated with Bell Northern Research of Northern Telecom, Inc., Richardson, TX. From 1996 to 2000, he was with AT&T Laboratories, where he was a Senior Technical Staff Member in the Wireless Communications Research Department. Since August 2000, he has been an Associate Professor in the Department of Communication Engineering, National Chiao-Tung University. His current research interests are in the areas of cellular architectures, radio network resource management, and cross-layer optimization for high-speed wireless networks. He is the holder of three U.S. patents with one more pending.

Dr. Wang was a corecipient of the Jack Neubauer Memorial Award in 1997 recognizing the best systems paper published in the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY. Currently, he is the Editor of the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS.