Multi-Code Placement and Replacement Schemes for W-CDMA Rotated-OVSF Code Tree

Yuh-Shyan Chen and Han-Chen Chang
Department of Computer Science and Information Engineering
National Chung Cheng University
Chiayi, 621, Taiwan, R.O.C.
E-mail: yschen@cs.ccu.edu.tw

Abstract— **Orthogonal variable spreading factor (OVSF) channelization codes are widely used to provide variable data rates for supporting different bandwidth requirements in WCDMA systems. Much work in the literature has intensively investigated code placement/replacement schemes in OVSF code trees to reduce the code blocking probability and reassignment code. This work introduces a new code tree structure, namely an ROVSF (rotated-OVSF) code tree, whose the code capability is the same as that of the traditional OVSF code tree. Using the single code placement and replacement schemes usually lead to the *internal* and *external* fragmentation problems in the traditional OVSF or ROVSF code trees. The internal and external fragmentation problems can be efficiently improved by using the multi-code placement and replacement schemes. Therefore, this work mainly investigates both multi-code placement and replacement problems in a ROVSF code tree system. In this paper, we developed new multi-code placement and replacement schemes in a ROVSF code tree to offer the results of the less code blocking probability and less reassignment cost. Finally, the simulation results illustrate that our multi-code placement/replacement results based on the ROVSF code tree can actually improve the code blocking probability and the code reassignment cost.**

I. Introduction

Wireless communication technology has been recently and widely investigated to significantly increase subscribers, traffic, and new bandwidth-consuming applications, such as mobile learning, mobile gaming, music down-loading and wireless video streaming. The capacity demand is supplied by provision of new spectrum technology, namely wideband CDMA or WCDMA (Wideband Code Division Multiple Access), in order to create a global standard for real-time multimedia services and support international roaming. Many existing results for code placement and replacement schemes are discussed in [1][2][3][4][5]. Tseng *et al.* proposed single-code placement and replacement schemes in WCDMA systems [5]. The algorithm for single-code placement/replacement is simple, but it possibly incurs many fragmental codes which produce a code blocking problem. Minn *et al.* [3] developed a dynamic assignment of OVSF codes to provide an optimal dynamic code assignment (DCA) scheme, which reassigns codes with minimum costs. To reduce internal and external fragmentations of OVSF codes, Chao *et al.* [1] recently developed the multi-code placement and replacement schemes in WCDMA systems. Moreover, the multi-OVSF code placement and replacement scheme [1] can reduce the code blocking problem by using the code-separation operation.

Two problems, *external* and *internal* fragmentation problems, are needed to be investigated in the WCDMA OVSF code tree management. External fragmentation problem happens when the capacity of the code tree is still enough to support the new arrival code, but the code tree is too fragmental to place the code. It is due to the limit for the capacity of WCDMA OVSF code tree and the number of available orthogonal codes. Internal fragmentation problem occurs when the request transmission rate is not power of two, but the data rates of WCDMA OVSF code tree are all power of two. It wastes the code tree capacity.

In this paper, we introduce a new code tree structure, namely an ROVSF (rotated-OVSF) code tree, whose the code capability is the same as that of the traditional OVSF code tree. In [2], Chen *et al.* developed single-code placement and replacement schemes in the ROVSF code tree to improve the code blocking probability and the code reassignment cost. Similarly, using the single code placement and replacement schemes lead to the *internal* and *external* fragmentation problems in the ROVSF code tree. To reduce the *internal* and *external* fragmentations of ROVSF codes, this work investigates the multi-code placement and replacement schemes in the ROVSF code tree. In this paper, we developed new multi-code placement and replacement schemes in a ROVSF code tree to efficiently reduce the *internal* and *external* fragmentation problems. Finally, the simulation results illustrate that our multi-code placement/replacement schemes have the better performance achievements.

The rest of this paper is organized as follows. Section II introduces the ROVSF code tree. The multi-code placement and replacement schemes based on the ROVSF code tree are presented in Section III and Section IV. In Section V, we discuss the simulation results to demonstrate the improvement in performance. Section VI concludes this paper.

II. The ROVSF Code Tree

In this paper, we initially define a new code tree, namely ROVSF (Rotated OVSF) code tree, and then propose a code placement/replacement algorithms on the ROVSF code tree. Several properties of the ROVSF code tree are further investigated in this article. Channelization codes in the ROVSF code tree have a unique description as $RC_{SF,k}$, where SF is the spreading factor of the code and k is the code number, $1 \leq k \leq SF$. Each level in the code tree defines channelization codes of length SF, corresponding to a spreading factor of SF. The code $RC_{SF,k}$ is denoted as a ROVSF code, observe that if any ROVSF code $RC_{X,Y}$ is orthogonal to its two children codes, $RC_{2X,2Y-1}$ and $RC_{2X,2Y}$. The ROVSF code tree is formally defined as follows.

Definition 1: **ROVSF Code Tree:** The ROVSF code of root node of ROVSF code tree is assumed as 1, and two children codes of the root node are initially set to be $(-1,-1)$ and $(-1,1)$, respectively. Consider a neighboring ROVSF codes $RC_{i,j} = (A)$ and $RC_{i,j-1=} (B)$ at k-th level, $i = 2^k$, of code tree, where A and B denote as the ROVSF codes of $RC_{i,j}$ and $RC_{i,j-1}$, respectively. Two children codes of $RC_{i,j}$ at $(k+1)$-th level of ROVSF code tree are $RC_{2i,2j-1} = (-B,-B)$ and

Yuh-Shyan Chen is with the Department of Computer Science and Information Engineering, National Chung Cheng University, Chiayi, Taiwan, R.O.C., E-mail: yschen@cs.ccu.edu.tw. This work was supported by the National Science Council of the Republic of China under grant #NSC92-2213-E-194-022.

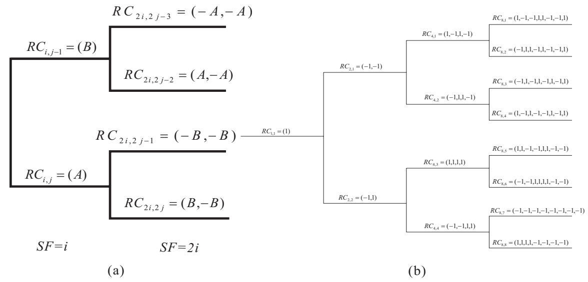

Fig. 1. Rotated OVSF (ROVSF) code-tree

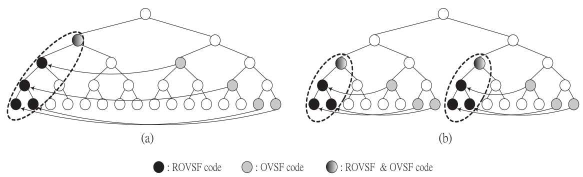

Fig. 2. Relationship of ROVSF and OVSF code trees

$RC_{2i,2j} = (B, -B)$. Similarly, two children codes of $RC_{i,j-1}$ are $RC_{2i,2j-2} = (A, -A)$ and $RC_{2i,2j-3} = (-A, -A)$. Two codes (P, Q) and (R, S) are said as brother codes if $Q = S$ and P is the complement of R, i.e., $P = -R$.

Fig. 1 shows an example of the ROVSF code tree. The key feature existing in the ROVSF-based scheme to reduce the code blocking probability is the *linear-code chain* as illustrated in Fig. 2. Observe that every node of an OVSF code tree is logically mapped to the corresponding node of a ROVSF code tree. These mapping nodes can form a code-chain, which is called as a *linear-code chain*. Observe that, one or more linear-code chains may exist in a ROVSF code tree. A linear-code chain is formally defined as follows.

Definition 2: **Linear-Code Chain:** Given a data rate, $R_{\max} = 2^{\log_2 R_{\max}}$ (or called a chain-max-code), we denote S as a linear-code chain as follows.

1) Let linear-code chain S be a subset of $S_k = [R_{\max}, \frac{R_{\max}}{2^1}, \frac{R_{\max}}{2^2}, \frac{R_{\max}}{2^3}, \cdots, \frac{R_{\max}}{2^k}]$, where $0 \leq k \leq \log_2(R_{\max})$,

or

2) Let linear-code chain $S = S_k \cup \left\{\frac{R_{\max}}{2^k}\right\} = [R_{\max}, \frac{R_{\max}}{2^1}, \frac{R_{\max}}{2^2}, \frac{R_{\max}}{2^3}, \cdots, \frac{R_{\max}}{2^k}, \frac{R_{\max}}{2^k}]$, where $\frac{R_{\max}}{2^k}$ and $\frac{R_{\max}}{2^k}$ are on the same level of the ROVSF code tree, and $0 \leq k \leq \log_2(R_{\max})$.

III. Multi-Code Placement Scheme on ROVSF Code Tree

To reduce the internal fragmentation and external fragmentation problems, multi-code placement scheme is developed herein. Three phases are performed in the multi-code placement scheme; (1) *multi-code separation* phase, (2) *multi-code LCC placement* phase, and (3) *multi-code NCC placement* phase. In the *multi-code separation* phase, multiple codes are calculated and determined if a new call with a rate requirement is not the power of two. The use of multi-codes provides high probability for high data-rate requirements.. We initially assign the request data-rate codes to the LCCs in the *multi-code LCC placement* phase. If we fail to find feasible codes in the LCCs, then we may try to find possible codes from NCCs in the *multi-code NCC placement* phase. From [2], we have the important result that linear-code chain is designed to reduce the code blocking probability in the placement scheme due to the code locality property. The utilization of LCCs significantly impacts the code blocking probability. Basically, the higher utilization of LCCs is, the lower the code blocking probability will be. Considering a *n*-layer ROVSF code tree, there exist $2^{n-\alpha-1}$ linear-code chains, where the chain-max-code is $R_{\max} = 2^{\alpha-1}$, the height of LCCs is assumed to be α, and k codes are used.

A. Phase 1: Multi-Code Separation

First, the incoming call is separated into multiple codes as follows. The request call is recursively derived to fit ROVSF code tree by the code separation equation, $f^k(i) = \begin{cases} 2^{\lceil \lg i \rceil} & n = 1, \\ 2^{\lfloor \lg i \rfloor} + f^{k-1}(i - 2^{\lfloor \lg i \rfloor})\ f^k(i) \leqslant 2^{\alpha-1}, n \geqslant 2, \\ 2^{\alpha-1} + f^{k-1}(i - 2^{\alpha-1}) \quad f^k(i) \geqslant 2^{\alpha-1}, n \geqslant 2. \end{cases}$

Therefore, multiple codes are extracted from equation $f^k(x)R = x_1R + x_2R + ... + x_kR$ by the descending order. For instance, assumed that $R_{\max} = 4$ and two codes are used. If an incoming call is $6R$, then $4R$ and $2R$ are obtained, where $f^2(6)R = 4R + 2R$.

B. Phase 2: Multi-Code LCC Placement

¿From multi-code separation phase, $f^k(x)R = x_1R + x_2R + ... + x_kR$, where x_i is the power of 2, $1 \leqslant i \leqslant k$. The multi-code LCC placement operation is to sequentially assigned x_i into LCCs. Consider an n-layer ROVSF code tree, for which $2^{n-\alpha-1}$ LCCs exists, where the chain-max-code $R_{\max} = 2^{\alpha-1}$.

The LCC placement phase will try to assign x_i, $1 \leqslant i \leqslant k$, to each of LCCs from left to right. Each linear-code chain has its own bit-word BW, this work can be achieved by checking by a bit-word sequence $[BW_1, BW_2, \cdots, BW_{2^{n-\alpha-1}}]$. In our scheme, a *leftmost-like* strategy is used as follows. We initially try to assign the incoming data rate x_iR to BW_1. If this fails, we continue by trying BW_2. We repeatedly execute the above operation until x_iR can be assigned to BW_j, where $j \leq 2^{n-\alpha-1}$. If we still cannot assign x_iR to $BW_{2^{n-\alpha-1}}$, then we perform the multi-code NCC placement operation, which is described later.

Now we describe how to assign x_i in $f^k(x)R = x_1R + x_2R + ... + x_kR$, x_i where $1 \leqslant i \leqslant k$, to an LCC with $BW = (b_m, b_{m-1}, b_{m-2}, \cdots, b_1, b_0)$, $b_p = 1, 0$, or $(1, 1)$, and $0 \leq p \leq m$, and BW is one of $BW_1, BW_2, \cdots, BW_{2^{n-\alpha-1}}$, . Let $\beta = \log_2 x_i$, where $f^k(x)R = x_1R + x_2R + ... + x_kR$ is the request data rate. Observe that if LCC kept $BW = (b_m, b_{m-1}, b_{m-2}, \cdots, (b_j, b_j), 0, \cdots, 0)$, where $b_j = 1$, then we cannot assign any data rate x_iR into the LCC if $\beta < j$. But we still can assign data rate x_iR into the LCC if $\beta > j$ and $b_\beta = 0$. Consider $f^k(x)R = x_1R + x_2R + ... + x_kR$, the formal algorithm of multi-code LCC placement operation is given.

Step 1: Repeatedly perform the following assignment to assign the x_iR, $1 \leqslant i \leqslant k$, into *l*-th LCC with a bit-word BW_l, where $1 \leq l \leq 2^{n-\alpha-1}$ and $\beta = \log_2 x_i$.

(a) Try to assign x_iR into *l*-th LCC to change the bit-word BW_l from $(b_m, b_{m-1}, b_{m-2}, \cdots, (1, 0)_j,\ 0, \cdots, 0)$ to $(b_m, b_{m-1}, b_{m-2}, \cdots, (1, 1)_j, 0,\ \cdots, 0)$ if $b_{j=\beta} = 1$ and all $b_\gamma = 0$, where $r < \beta = j$. If the assignment is successful, then go to **Step 2**.

(b) If a $(b_m, b_{m-1}, b_{m-2}, \cdots, (b_j, b_j), 0, \cdots, 0)$ exists and $\beta < j$,and $b_j = 1$, then the assignment fails even if $b_\beta = 0$. If $\beta > j$ and $b_\beta = 0$, then the assignment is successful; go to **Step 2**.

(c) If $b_\beta = 1$ and one value of $b_\gamma = 1$, where $r < \beta$, then the assignment fails; go to **Step 1**.

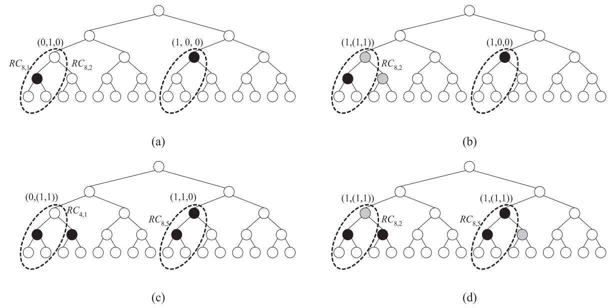

Fig. 3. Example of LCC placement phase

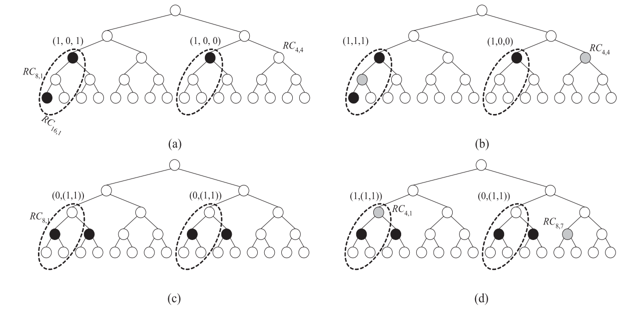

Fig. 4. Example of NCC placement phase

Step 2: If the incoming data rate x_iR, $1 \leqslant i \leqslant k$, cannot be assigned to the last LCC with bit-word $BW_{2^{n-\alpha-1}}$, then execute the multi-code NCC placement phase.

For instance, consider the ROVSF code tree in Fig. 3(a). Assume that there is a new call with a rate requirement $f^2(6)R = 4R + 2R$. Data rate $4R$ is assigned to $RC_{4,1}$ on the first LCC with bit-word $(0, 1, 0)$ and $2R$ is assigned to $RC_{8,2}$ on the first LCC with bit-word $(1, 1, 0)$. Another example, shown in Fig. 3(c), is of a new call with $6R = 4R + 2R$, thus $4R$ and $2R$ are assigned to $RC_{4,1}$ and $RC_{8,6}$, respectively.

C. Phase 3: Multi-Code NCC Placement

If a data rate, x_i, where incoming data rate $f^k(x)R = x_1R + x_2R + ... + x_kR$, fails to be assigned in the multi-code LCC placement phase, then multi-code NCC placement phase is performed to assign x_i into NCCs, where $\beta = \log_2 x_i$. Observe that an NCC is a sub-tree which contains no LCC. We give the formal algorithm of NCC placement operation.

Step 1: Repeatedly perform the following procedure to assign data rates, x_i, to nodes of a neighboring sub-tree of the l-th LCC with a bit-word BW_l, where $1 \leq l \leq 2^{n-\alpha-1}$ and $\beta = \log_2 x_i$.

(a) If a LCC with a bit-word $BW_l = (b_\beta = 1, 0, 0, 0, \cdots, 0)$, indicates that a node α in LCC is already assigned the same data rate x_iR; then we may assign x_iR to a neighboring node of node α of the LCC on the level β of the ROVSF code tree.

(b) If an LCC with a bit-word $BW_l = (0, 0, \cdots, 0, (1, 1)_j, 0, \cdots, 0)$ is existed if $\beta = j$, then we may assign x_iR to nodes on the same level j of neighboring sub-tree of the LCC.

Step 2: If data rate x_iR eventually cannot be assigned to neighboring codes of any LCCs, then x_iR is blocked.

For example, consider a ROVSF code tree as shown in Fig. 4(a). A new call with a rate requirement $f^2(6)R = 4R+2R$ exists, however the multi-code LCC code placement operation assigns $2R$ to $RC_{8,1}$ but fails to be assigned for $4R$, therefore $4R$ is assigned to $RC_{4,4}$ in the multi-code NCC placement phase as shown in Fig. 4(b). Another example is given as shown in Fig. 4(c) Fig. 4(d).

IV. Multi-Code Replacement Scheme on ROVSF Code Tree

The code (internal/external) fragment problem always occurs in a code tree after a number of variable data codes have been allocated and released. To completely eliminate code blocking with less reassignment cost, a multi-code replacement scheme is developed to relocate some codes in the ROVSF code tree to "squeeze" a free space for the new call. To

To minimize the number of reassigned codes, Minn and Siu [3] developed a dynamic assignment of OVSF codes to provide a dynamic code assignment (DCA) scheme. The DCA algorithm is guaranteed to eliminate the code blocking problem. The use of linear-code chains can reduce the code blocking probability during the code placement operation. Our multi-code replacement scheme aims to reduce the reassignment cost for ROVSF code tree management. Our proposed multi-code replacement scheme on the ROVSF code tree is modified from the DCA algorithm [3] as follow.

Our ROVSF-based DCA-modified algorithm is given We modify the DCA algorithm to be a ROVSF-based DCA algorithm by providing a different rule of the *minimum-cost branch* [3] for the ROVSF code tree. Consider that a ROVSF code tree has sufficient code capacity, assumed that there are $2^{n-\alpha-1}$ nodes $RC_{2^{n-\alpha},l}$, where $1 \leq l \leq 2^{n-\alpha}$. Each node $RC_{2^{n-\alpha},l}$, denoted as a *branch node* is used to maintain a *branch cost*. The *branch cost* is a count variable used to estimate the number of assigned codes. The *branch cost* also denotes the number of assigned codes which needs to be reassigned.. The greater the number of the count variable is, the higher reassignment cost will be. Each node, $RC_{2^{n-\alpha},l}$, forms a sub-tree, for which $RC_{2^{n-\alpha},l}$ is viewed as its root. Each node, $RC_{2^{n-\alpha},l}$, connects to an LCC or NCC, depending on whether the value of l is odd or even. For each $RC_{2^{n-\alpha},l}$, the *branch cost*, denoted $C_i = C(RC_{2^{n-\alpha},j})$, is determined by the total number of assigned nodes in the neighboring sub-tree or branch. Assuming that $RC_{2^{n-\alpha},l}$ and $RC_{2^{n-\alpha},j}$ are a pair of neighboring nodes, a minimum-cost branch is determined as follows. A branch node is said to be a minimum-cost branch, which is determined by $\underset{1 \leq i \leq 2^{n-\alpha}}{Min} C_i$. To reorganize the ROVSF code tree structure in order to obtain the higher date rate, all assigned codes in the neighboring sub-tree are released and reallocated to other sub-trees.. The reallocating operation occurs by sequentially performing our code placement algorithm for each of the assigned codes in the neighboring sub-tree.

To eliminate the code blocking problem in ROVSF code tree with less reassignment cost, the code replacement algorithm is formally given. Consider a incoming rate $f^k(x)R = x_1R + x_2R + ... + x_kR$, where x_i is the power of 2, $1 \leqslant i \leqslant k$, each of x_i is executed the following operations.

Step 1: Check to see if the ROVSF code tree has sufficient code capacity to offer a new call with the data rate requirement x_iR. If it is failed, reject this request.

Step 2: Once the request is allowed, a ROVSF-based DCA algorithm is performed; a *minimum-cost branch* is initially found which accommodate the date rate, x_iR.

Step 3: Sequentially execute the multi-code LCC or NCC placement operations to relocate all assigned codes in the neighboring sub-tree for the *minimum-cost branch*.

For example as shown in Fig. 5, the total free capacity of the ROVSF code tree is $6R$. A new call is requested a data rate $f^2(6)R = 4R + 2R$ exists, however it is failed for both the multi-code LCC and NCC placement phases. The multi-code replacement operation is applied as follows; $RC_{16,5}$, $RC_{16,7}$

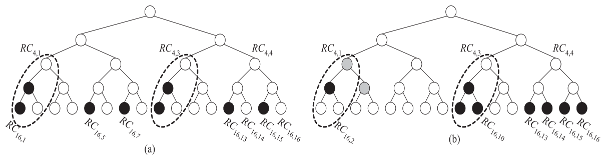

Fig. 5. Example of code replacement

are moved to $RC_{16,10}$ and $RC_{16,14}$, and $4R$ is assigned to $RC_{4,1}$, and then $RC_{16,1}$ is moved to $RC_{16,16}$, and $2R$ is finally assigned to $RC_{8,2}$.

V. Performance Analysis

We have developed a discrete event simulator to evaluate the performance achievement. To examine the effectiveness of our scheme, it mainly compared with OVSF-based multi-code placement/replacement schemes in [1]. In the following simulator, we used OR2, OL2, OC2, OM2, RL2, RC2, and RM2, to denote different type of code trees and code placement strategies, where the first letter indicates the type of code tree (O =*OVSF* and R =*ROVSF*), the second letter indicates the code placement strategy (R = *Random*, L = *Leftmost*, C = *Crowded-first*, and M = *Mostuser-first*) and the third letter indicates the number of codes can be used in a call (2 codes are used). Each curve is obtained by about 1000 simulation runs, where each run contains at least 4000 accepted calls. The system parameters in our simulator are given as follows.

- Capacity test: code-limited.
- Maximum spreading factors: 256 to reflect values following IS-95 and WCDMA standards.
- Call arrival process: Poisson distribution with a mean arrival rate $\lambda = 4$ to 64 calls/unit time ($SF = 256$).
- Mean call duration time: Exponentially distributed with a mean value of four time units.
- Possible transmission rates: $1, 2, 3, 4, 5, 6, 7$, or $8R$.

In the following, we illustrate our simulation results from various perspectives.

A. Impact of Multi-Code Placement

Fig. 6(a) shows the code blocking probability if only code placement is implemented, under fixed mean arrival rate and $SF = 256$. The ROVSF-based scheme with leftmost strategy (RL2) has the lowest code blocking probability. This is because that code blocking is always occurred for the new calls with high transmission rate. Recalled that the LCC has the *unsequence allocation* property [2]. Therefore, the lower code blocking probability is obtained because that our scheme tries to allocate most multi-code requests onto the LLCs. Using RL2, our scheme can allocate more codes into LCCs. Only a few code requests, which are allocated outside the LCCs, may experience code blocking problems. To understand the effect of utilization of LCCs vs. code blocking probability, the utilization of LCCs for RL2, RC2, and RM2 are given in Fig. 6(b), where $SF = 256$. Basically, the higher the utilization of LCCs is, the lower the code blocking probability that is obtained. For instance, the utilizations of LCC of RL2 are 1 and 0.85, and the code blocking probabilities are 0 and 0.15, respectively. Similar results for RC and RM schemes can be obtained in Fig. 6(b). Observe that our scheme tries to properly assign most multi-codes into LCCs, thus the lower code blocking probability than other schemes can be obtained.

B. Impact of Multi-Code Replacement

The code replacement scheme is designed to completely eliminate the code blocking problem since that there is suffi-

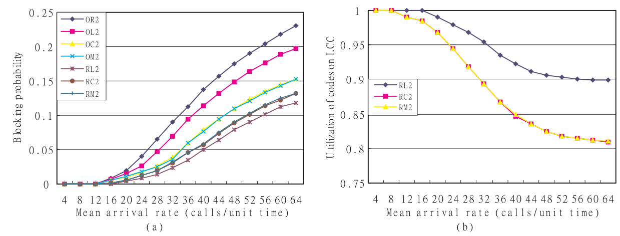

Fig. 6. $SF = 256$: (a) blocking probability at different traffic load and (b) utilization of linear-code chains

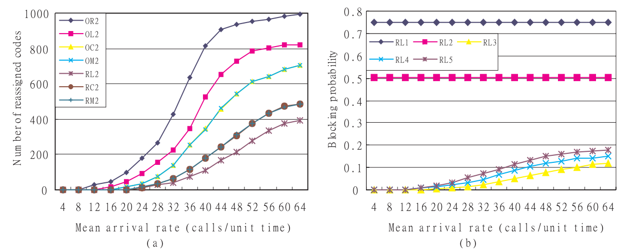

Fig. 7. $SF = 256$ (a) number of code reassignments to remove code-blocking (b) the impact of the length of LCC on blocking probability

cient free capacity. To see the efficiency of our multi-code replacement scheme, observe that the total number of code reassignments. Fig. 7 shows that RL2 has the lowest reassignment cost, when compared to all other schemes. This is possibly because that the RL2 has the better utilization of LCCs, thus it can reduce the total number of code reassignments to accommodate the new call with high data rate. Consequently, our ROVSF-based scheme has an improved number of code reassignments.

VI. Conclusions

In this paper, we introduce a new code tree structure, namely an ROVSF (rotated-OVSF) code tree, whose the code capability is the same as that of the traditional OVSF code tree. To reduce the *internal* and *external* fragmentation problems in the ROVSF code trees, we mainly investigates multi-code placement and replacement problems in a ROVSF code tree system to offer the results of the less code blocking probability and less reassignment cost. Finally, the simulation results illustrate that our multi-code placement/replacement results based on the ROVSF code tree can actually improve the code blocking probability and the code reassignment cost.

References

[1] C. M. Chao, Y. C. Tseng, and L. C. Wang. Reducing Internal and External Fragmentations of OVSF Codes in WCDMA Systems with Multiple Codes. In *Proceedings of IEEE Wireless Communications and Networking Conference*, volume 1, pages 693–698, 2003.

[2] Y.-S. Chen and T.-L. Lin. Code Placement and Replacement Schemes for W-CDMA Rotated-OVSF Code Tree Management. In *Proceedings of The International Conference on Information Networking (ICOIN 2004)*, Busan, Korea, February 2004.

[3] T. Minn and K. Y. Siu. Dynamic Assignment of Orthogonal Variable-Spreading-Factor Codes in W-CDMA. *IEEE Journal on Selected Areas in Communications*, 18(8):1429–1440, August 2000.

[4] Third Generation Partnership Project. Technical Specification Group Radio Access Network. Technical report, Spreading and Modulation, http://www.3gpp.org, 1999.

[5] Y. C. Tseng and C. M. Chao. Code Placement and Replacement Strategies for Wideband CDMA OVSF Code Tree Management. *IEEE Transactions on Mobile Computing*, 1(4):293–302, October-December 2002.