An OVSF Code Tree Partition Policy for WCDMA Systems Based on the Multi-Code Approach

Huei-Wen Ferng, Hao-Lun Chin, David Shiung†, and Ying-Tsung Chen

Department of Computer Science and Information Engineering
National Taiwan University of Science and Technology, Taipei 106, Taiwan
†MediaTek Inc., No. 1-2, Innovation Road I, Science-Based Industrial Park, Hsin-Chu 300, Taiwan
E-mail: hwferng@mail.ntust.edu.tw

***Abstract*— Orthogonal variable spreading factor (OVSF) codes for wide-band CDMA (WCDMA) systems have been proposed in the third generation (3G) mobile communications' standard to support services of variable rates. To assign OVSF codes, there are single-code and multi-code approaches. In this paper, the multi-code placement and replacement issue is addressed. We propose a tree partition policy for managing the OVSF code tree to reduce code fragments and the number of code reassignments. Besides, we adjust the multi-code rate according to a unit-based method, which is capable of decreasing the number of code fragments because fewer low-rate codes are utilized. Through extensive simulations, it turns out that the tree partition policy and the unit-based multi-code method jointly perform better than the left-most and crowded-first schemes.**

I. Introduction

The universal mobile telecommunication system (UMTS) employs WCDMA to enable high-rate data transmission and variable-bit-rate services for different users using OVSF codes as channelization codes for connections. Since a different *spreading factor* (SF) can provide a different rate service, variety of requested rates can be flexibly fulfilled. OVSF codes may be represented as a binary code tree [1], in which the spreading factor of the downstream level is a power of two of the current level, while the relationship of code rates for these two levels is reverse. Any two codes are mutually orthogonal if and only if one code is not the ancestor or descendant code of the other. Avoiding simultaneous use of ancestor and descendant codes, the orthogonality can be maintained.

In the literature, there are two assignment approaches for OVSF codes: single-code [2], [3], [4] and multi-code [5], [6], [7]. Single-code assignment allocates only one code to a user. It may cause rate wastage and a code fragmentation problem. Multi-code assignment can support multi-rate services using multiple codes with multiple rake combiners. It is able to reduce the code blocking probability because a small-SF OVSF code (a high-rate code) can be used to replace some large-SF OVSF codes (low-rate codes) to avoid code fragmentation. In [5], a multi-code assignment method is proposed. Among possible candidates, this method selects the best multi-code with less codes but more small-SF codes. Its main drawback is that it relies on the complicated computation. Although it indeed avoids using too many large-SF codes, it perhaps blocks small-SF ones. Moreover, the code placement and replacement are not taken into consideration. To prevent redundant computation, the scheme proposed in [6] uses the technique of dynamic programming scheme to build up a multi-code table in advance so that less time in selecting the best multi-code is required. In [7], the internal and external fragmentation problems are defined and solved by using multi-codes. To deal with the code placement and replacement, crowded-first and left-most schemes are proposed in [4]. The disadvantages of these two schemes are extra storage and code reassignments required.

What we have learned from the aforementioned work is that multi-codes have many merits but may cause side effects. When multi-codes are released, they may split resources, resulting in more fragments which may subsequently result in code blocking of high-rate codes. Moreover, the more code fragments we have, the more code reassignments are required. To eliminate the above-mentioned side effects, we propose an OVSF code tree partition policy called high rate right most (HRRM) and the unit-based multi-code (UBMC) method to solve the problem of code fragmentation and to reduce a large amount of code reassignments in this paper. Compared with the existing placement and replacement schemes, i.e., crowded-first and left-most, the advantages of the proposed schemes are demonstrated in the later section.

The rest of the paper is organized as follows. The system model of multi-code OVSF is introduced in Section II. In Section III, the proposed schemes, i.e., HRRM and UBMC, are described in detail. After that, the simulation model and numerical results are presented in Section IV. Finally, Section V concludes the paper.

II. System Model of Multi-Code OVSF

OVSF codes can be depicted by a binary code tree as shown in Fig. 1 in which $C_{l,b}$ denotes the bth code of the lth layer. Counting upwards, the code rate and SF of the hth layer code(s) are $2^{h-1}R$ and $SF_{max}/2^{h-1}$, respectively if the code rate and SF of the leaf code are set to R (unit rate) and SF_{max}, respectively. For ease of description, let us define some notation in the following. First, we denote $S = (S_1, S_2, S_3, S_4, S_5, S_6, S_7)$ to represent the system codeword, where S_i is the number of available codes of the ith layer. Considering a code tree with 9 layers and $SF_{max} = 256$, then

0-7803-9152-7/05/$20.00 (c) 2005 IEEE

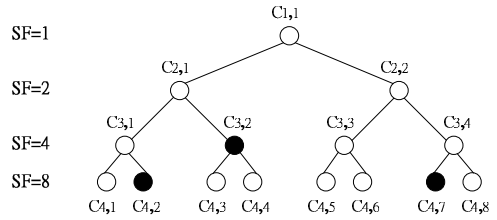

Fig. 1. An OVSF code tree (busy codes are in the black color).

$S = (4, 0, 0, 0, 0, 0, 0)$ initially since we can divide the entire code tree into four 7-layer subtrees. Furthermore, let $W(S)$ and $C = (C_1, C_2, C_3, C_4, C_5, C_6, C_7)$ denote the capacity of S and the requested multi-code codeword, respectively, where C_i is the number of required codes of the ith layer. Explicitly, $W(S) = \sum_{i=1}^{7} S_i 2^{7-i} R$ and $W(C)$ represents the requested rate. Finally, let $N(C) = \sum_{i=1}^{7} C_i$ denote the number of codes required by a multi-code C.

For a UE equipped with k rake combiners, the number of multi-codes required cannot exceeds k codes. Therefore, the larger k is, the more variety we have. Note that the limit on the number of rake combiners is mainly due to the hardware constraints. In the following, we denote $D(w, k)$ to be the multi-code set for a new call requesting rate w under k rake combiners. For example, the maximum set of $D(4R, 3) = \{(0, 0, 0, 0, 1, 0, 0), (0, 0, 0, 0, 0, 2, 0), (0, 0, 0, 0, 0, 1, 2)\}$. Among possible candidates in $D(w, k)$, one can be selected to serve the incoming call. In the following, we select the *best* multi-code based on the following principle: 1) the multi-code with more small-SF codes after allocation is selected; 2) if two and more multi-codes have the same number of small-SF codes after allocation, the multi-code with the least codes is chosen; 3) if no multi-code meets the constraint of rake combiners but there is enough capacity, we may allow a bit more rate wastage.

Now, let us describe the code fragmentation problem. The problem occurs when a multi-code can not be allocated because the OVSF code tree is too fragmented even if the system still has sufficient capacity. Then, the incoming call will be blocked if no code replacement/reassignment is involved. Thus, such a blocking probability is defined as the *code blocking probability* for this particular situation. Fig. 1 gives an example of code blocking. Assume a call requests a rate of $4R$ with 2 rake combiners. Thus, $D(4R, 2) = \{(0, 0, 0, 0, 1, 0, 0), (0, 0, 0, 0, 0, 2, 0)\}$. Although the current system codeword S is $(0, 0, 0, 0, 0, 1, 2)$ and the remaining capacity is sufficient, we can not find a suitable choice in the set $D(4R, 2)$ to satisfy the request. To solve the code fragmentation problem, it is inevitable to perform code reassignments. In the literature, researchers focus on the code placement and replacement issue by devising schemes with less fragments and reassignments. In this paper, we touch this issue, too.

III. Proposed schemes

We address the placement and replacement issue in this paper so that the OVSF code tree is well-managed. We first propose a tree partition policy, i.e., HRRM for the OVSF code tree. Then, we modify the multi-code rate using UBMC. Finally, UBMC is combined with HRRM. Let us describe these schemes in detail as follows.

A. High rate right most (HRRM)

Since both low-rate codes and high-rate codes may be requested at the same time, the purpose of the tree partition policy is to utilize codes of separated areas in the code tree for high-rate and low-rate codes. For low-rate codes, we assign codes from the left to the right in the code tree, while high-rate codes are assigned from the right to the left in the code tree. Thus, most of low-rate codes are put to the left-hand side of the code tree and this area forms a low-rate partition. Similarly, most of high-rate codes can form a high-rate partition on the right-hand side of the code tree. In the following, let us specifically outline different cases for HRRM.

- Case 1: If the requested multi-code rates range among R, $2R$, $4R$, $8R$, and $16R$, we find available codes accordingly from the left to the right in the code tree.
- Case 2: If the requested multi-code rates are $32R$ and $64R$, we find available codes accordingly from the right to the left in the code tree.

The benefit of HRRM is that a more complete code tree can be got when a multi-code is released. Since the high-rate (low-rate) codes are put together on the right-hand (left-hand) side of the code tree, no mixture of high-rate and low-rate codes occurs. This makes the high-rate partition will have larger capacity when codes are released, thus more chances to successfully get the high-rate codes in the future. That is the reason why we can reach a lower code blocking probability and assure a less number of code reassignments.

To further reduce code reassignments, we take a reactive policy which performs code reassignments only when code fragments happen under the condition that the remaining capacity of the code tree is sufficient. Furthermore, we adopt the dynamic code assignment (DCA) scheme proposed in [3] to find the least reassignment cost subtree (*minimum-cost subtree*). For each subtree, HRRM is applied.

B. Unit-based multi-code (UBMC)

In general, the multi-code can be used to solve the code fragmentation problem. However, the code tree will get more and more fragmented because too many dispersed low-rate codes in the code tree. This makes high-rate codes have a much higher probability of being blocked. For this reason, we propose a unit-based method to diminish the number of low-rate codes. We adjust the multi-code rate to a multiple of a designate rate unit, say U. For instance, a call requesting a rate of $3R$ will be assigned a rate of $4R$ if $U = 4R$. So, the multi-code is transformed from $(0, 0, 0, 0, 0, 1, 1)$ to $(0, 0, 0, 0, 1, 0, 0)$. Similarly, the assigned rate will be $8R$ for a call requesting a rate of $6R$. Thus, the multi-code is transformed from $(0, 0, 0, 0, 1, 1, 0)$ to $(0, 0, 0, 1, 0, 0, 0)$. The above demonstrates that the number of low-rate codes (i.e., $1R$ and $2R$) will be decreased. But bandwidth wastage is

TABLE I

A LOOKUP TABLE FOR UBMC ($U = 4R$, $k = 1 \sim 4$, AND $w = R \sim 32R$)

Rate	$k=1$	$k=2$	$k=3$	$k=4$
$4R$	(0,0,0,0,1,0,0)	(0,0,0,0,0,2,0)	(0,0,0,0,0,1,2)	(0,0,0,0,0,0,4)
$8R$	(0,0,0,1,0,0,0)	(0,0,0,0,2,0,0)	(0,0,0,0,1,2,0)	(0,0,0,0,0,4,0) (0,0,0,0,1,1,2)
$12R$	none	(0,0,0,0,1,1,0,0)	(0,0,0,0,3,0,0) (0,0,0,0,1,0,2,0)	(0,0,0,0,2,2,0) (0,0,0,0,1,0,1,2)
$16R$	(0,0,1,0,0,0,0)	(0,0,0,0,2,0,0,0)	(0,0,0,0,1,2,0,0)	(0,0,0,0,4,0,0) (0,0,0,0,1,1,2,0)
$20R$	none	(0,0,0,1,0,1,0,0)	(0,0,0,0,2,1,0,0) (0,0,0,1,0,0,2,0)	(0,0,0,0,1,3,0,0) (0,0,0,0,2,0,2,0) (0,0,0,1,0,0,1,2)
$24R$	none	(0,0,0,1,1,0,0,0)	(0,0,0,0,3,0,0,0) (0,0,0,1,0,2,0,0)	(0,0,0,0,2,2,0,0) (0,0,0,0,1,0,1,2,0)
$28R$	none	none	(0,0,0,1,1,1,0,0)	(0,0,0,0,3,1,0,0) (0,0,0,1,0,3,0,0) (0,0,0,1,1,0,2,0)
$32R$	(0,1,0,0,0,0,0)	(0,0,0,2,0,0,0,0)	(0,0,0,1,2,0,0,0)	(0,0,0,4,0,0,0) (0,0,0,1,1,2,0,0)

TABLE II

A LOOKUP TABLE FOR UBMC ($U = 16R$, $k = 1 \sim 4$, AND $w = 33R \sim 128R$)

Rate	$k=1$	$k=2$	$k=3$	$k=4$
$48R$	none	(0,1,1,0,0,0,0)	(0,1,0,2,0,0,0) (0,0,3,0,0,0,0)	(0,0,2,2,0,0,0) (0,1,0,1,2,0,0)
$64R$	(1,0,0,0,0,0,0)	(0,2,0,0,0,0,0)	(0,1,2,0,0,0,0)	(0,0,4,0,0,0,0) (0,1,1,2,0,0,0)
$80R$	none	(1,0,1,0,0,0,0)	(0,2,1,0,0,0,0) (1,0,0,2,0,0,0)	(0,1,3,0,0,0,0) (0,2,0,2,0,0,0) (1,0,0,1,2,0,0)
$96R$	none	(1,1,0,0,0,0,0)	(0,3,0,0,0,0,0) (1,0,2,0,0,0,0)	(0,2,2,0,0,0,0) (1,0,1,2,0,0,0)
$112R$	none	none	(1,1,1,0,0,0,0)	(0,3,1,0,0,0,0) (1,0,3,0,0,0,0) (1,1,0,2,0,0,0)
$128R$	none	(2,0,0,0,0,0,0)	(1,2,0,0,0,0,0)	(0,4,0,0,0,0,0) (1,1,2,0,0,0,0)

inevitable. Table I (II) shows all possible multi-codes when $U = 4R$ ($U = 16R$) for different number of rake combiners k and requested rate w. Since high-rate requests have a larger scale of code rate range, they will have more extremely high-rate and low-rate codes simultaneously. Therefore, we increase the rate unit for higher rate requests to decrease the number of low-rate codes.

Sacrificing some capacity by UBMC, we can increase the probability of serving new requests and reduce the number of code fragments. Furthermore, our code tree will become less fragmented because the number of the low-rate codes are diminished. In order to lessen the degree of using too much additional capacity, we set a threshold T. When the total amount of free capacity of the code tree is more than T, we use UBMC; otherwise, UBMC is not employed. In the following, we describe UBMC under $T = 64R$.

- Step 1: Check whether the total free capacity is greater than T. If yes, go to the next step; otherwise, use the original multi-code instead of UBMC and then quit.
- Step 2: If a call requests a rate not larger than $32R$, use UBMC with $U = 4R$; otherwise, go to step 3.
- Step 3: If a call requests a rate larger than $32R$, use UBMC with $U = 16R$.

Combining HRRM and UBMC, we may take advantages of both schemes so that the code blocking probability and number of code reassignments can be greatly reduced.

IV. SIMULATION RESULTS AND DISCUSSIONS

We have implemented a simulator run on an IBM compatible PC to have our proposed schemes evaluated. Let us

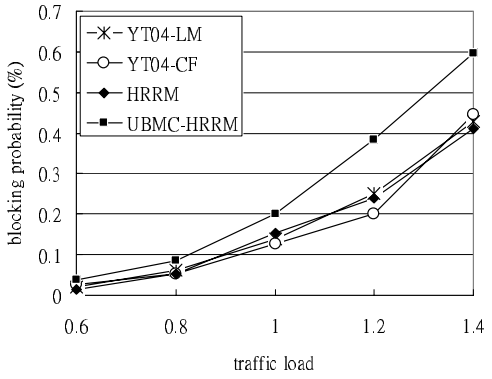

Fig. 2. Blocking probabilities for different schemes when $M = 64$.

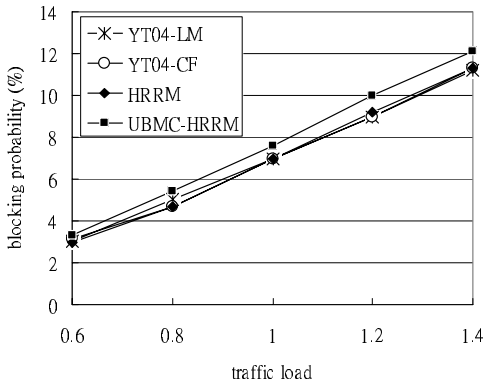

Fig. 3. Blocking probabilities for different schemes when $M = 128$.

first describe assumptions for system models. Assume that the OVSF code tree has 9 layers and $SF_{max} = 256$ with four 7-layer subtrees and $S = (4, 0, 0, 0, 0, 0, 0)$ initially. The capacity of the whole system is assumed to be $256R$. A poisson process is employed to model call arrivals. For each call, call duration is assumed to be exponentially distributed. In the following simulations, the requested rate w is generated randomly between R and MR. As for the number of rake combiners k, k is uniformly distributed between 1 (2) and 4 (4) when $M = 64$ ($M = 128$).

We now evaluate the performance of the proposed schemes against the multi-code assignment schemes presented in [6]. In the following, the performance of [6] is denoted by YT04-LM and YT04-CF, where LM=left-most and CF=crowded-first. Firstly, we measure the blocking probability for each scheme. The blocking probability is caused by two reasons: insufficient capacity of the code tree, which causes the so-called capacity blocking and code fragmentation which results in the code blocking probability. Consequently, the blocking probability is composed of the capacity blocking probability and the code blocking probability. Secondly, we count the number of code fragments and calculate the code blocking probability. Thirdly, we show the number of code reassignments. At last, we discuss how to design an appropriate rate unit U.

Figs. 2 and 3 show the blocking probabilities for different schemes when $M = 64$ and $M = 128$, respectively. In both

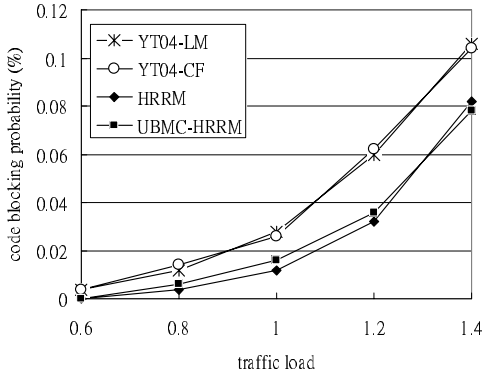

Fig. 4. Code blocking probabilities for different schemes when $M = 64$.

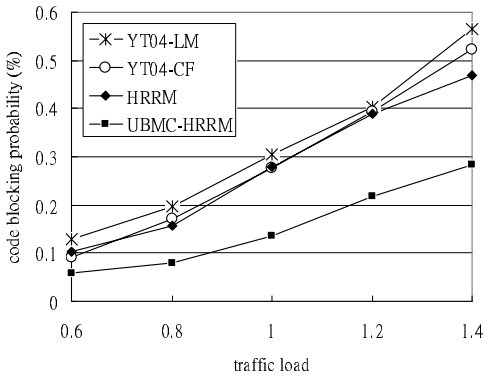

Fig. 5. Code blocking probabilities for different schemes when $M = 128$.

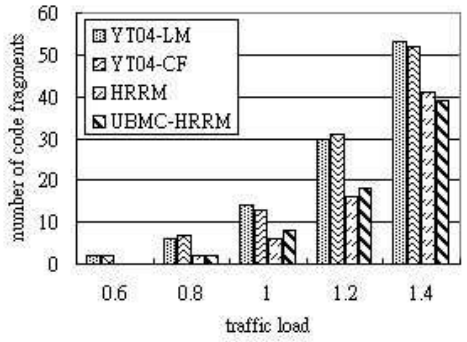

Fig. 6. Numbers of code fragments for different schemes when $M = 64$.

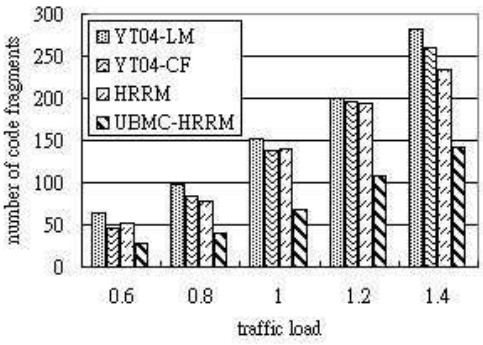

Fig. 7. Numbers of code fragments for different schemes when $M = 128$.

figures, we observe that YT04-LM, YT04-CF, and HRRM almost have the same blocking probability. However, UBMC-HRRM (a joint version of UBMC and HRRM) has a bit higher blocking probability because UBMC-HRRM will grant the request with a slightly higher rate, causing noticeable capacity wastage for $M = 64$ and slight capacity wastage for $M = 128$. Figs. 4 and 5 present code blocking probabilities for different schemes for $M = 64$ and $M = 128$, respectively. In Fig. 4, we observe that HRRM (UBMC-HRRM) outperforms YT04-LM, and YT04-CF by reducing 25% of code blocking probability when the traffic load is 1.4. Of course, UBMC seems to be not necessary for low-rate requests, i.e., $M = 64$ since no significant reduction in the code blocking probability. In Fig. 5, UBMC-HRRM obviously greatly improves the code blocking probability (its improvement is 50% over YT04-LM and YT04-CF when the traffic load is 1.4). Note that HRRM performs slightly better than YT04-LM and YT04-CF. Likewise, the same phenomena can be observed in Figs. 6 and 7. In Figs. 8 and 9, the number of code reassignments are counted. With the assistance of code reassignments, we can mitigate the code fragmentation problem by paying extra cost and spending more time. In both figures, we see that UBMC-HRRM has the least number of code reassignments. Compared with YT04-LM and YT04-CF, the number of code reassignments of UBMC-HRRM (HRRM) gets 51% (16%) of reduction for $M = 128$ and 36% (16%) of reduction for $M = 64$ when the traffic load is 1.4. In Figs. 10 and 11, we show the blocking probability and code blocking probability of different rate units when $M = 128$. What we have learned from these two figures are i) the larger U is, the lower the code blocking probability is; ii) a reverse phenomenon is found for the relationship between the blocking probability and the rate unit. Consequently, a tradeoff exists between the blocking probability and the code blocking probability for different values of U. As shown from these two figures, $U = 16R$ for high-rate requests, i.e., $M = 128$ is a good choice. That explains the reason why we choose $U = 16R$ when $M = 128$ in the previous parameter setting.

V. Conclusions

The side effects of the multi-code approach are code fragmentation and a large number of code reassignments. Therefore, we propose an OVSF code tree partition policy called HRRM and the unit-based multi-code method called UBMC in this paper to solve the problem of code fragmentation and reduce a large amount of code reassignments. Using HRRM, the code tree will become more complete with larger capacity when the multi-code is released and we can diminish the number of low-rate codes using UBMC which results in less fragments. Through simulations, we have successfully demonstrated that our proposed schemes perform better than the previously proposed schemes in [6] in terms of code blocking probability and number of code reassignments.

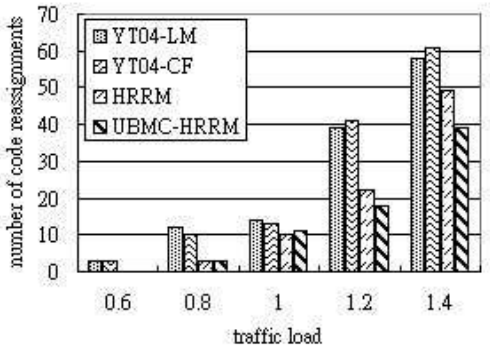

Fig. 8. Numbers of code reassignments for different schemes when $M = 64$.

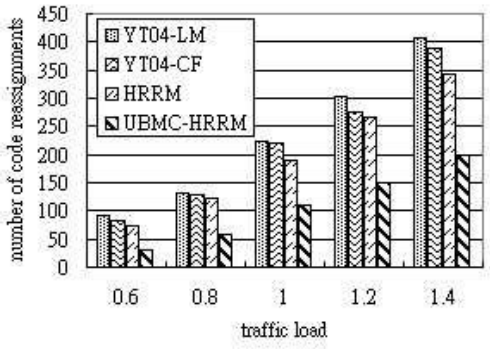

Fig. 9. Numbers of code reassignments for different schemes when $M = 128$.

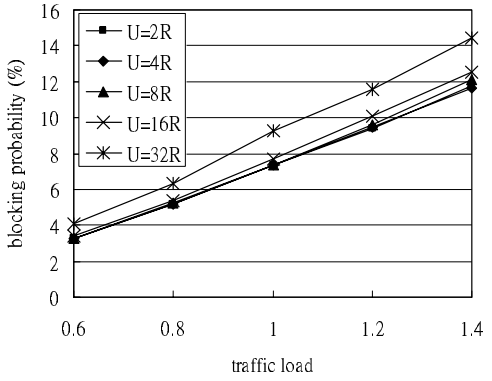

Fig. 10. Blocking probabilities of UBMC-HRRM when $M = 128$.

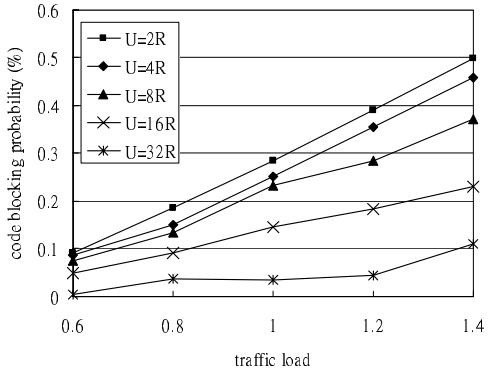

Fig. 11. Code blocking probabilities of UBMC-HRRM when $M = 128$.

ACKNOWLEDGMENT

The first author thanks the partial support in finance by the National Science Council, Taiwan under Contract NSC 93-2219-E-011-007.

REFERENCES

[1] 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," *Electronics Letters*, vol. 33, no. 1, pp. 27–28, Jan. 1997.

[2] W. T. Chen, Y. P. Wu, and H. C. Hsiao, "A novel code assignment scheme for W-CDMA systems," in *Proc. IEEE VTC '01*, Oct. 2001, pp. 1182–1186.

[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*, vol. 3, no. 8, pp. 1429–1440, Aug. 2000.

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

[5] R. G. Cheng and P. Lin, "OVSF code channel assignment for IMT-2000," in *Proc. IEEE VTC '00*, May 2000, pp. 2188–2192.

[6] L. H. Yen and M. C. Tsou, "An OVSF code assignment scheme utilizing multiple rake combiners for W-CDMA," *Computer Communications*, vol. 27, no. 16, pp. 1617–1623, Oct. 2004.

[7] C. M. Chao, Y. C. Tseng, and L. C. Wang, "Reducing internal and external fragmentation of OVSF codes in WCDMA systems with multiple codes," in *Proc. IEEE WCNC '03*, Mar. 2003, pp. 693–698.