Code Placement and Replacement Schemes for WCDMA Rotated-OVSF Code Tree Management

Yuh-Shyan Chen, *Member*, *IEEE*, and Ting-Lung Lin

Abstract—Orthogonal variable spreading factor (OVSF) channelization codes are widely used to provide variable data rates for supporting different bandwidth requirements in wideband code division multiple access (WCDMA) systems. Many novel works in the literature have intensively investigated code placement and replacement schemes in OVSF code trees to reduce the code blocking probability and the code reassignment cost. In this paper, we introduce a new code tree structure, namely, a rotated-orthogonal variable spreading factor (ROVSF) code tree, whose code capacity is the same as that of the traditional OVSF code tree. This work addresses both code placement and replacement schemes in the ROVSF code tree system, where ROVSF codes can be used at the forward link of WCDMA systems. Some valuable properties of the developed ROVSF code tree are presented to develop code placement/replacement schemes with lower code blocking probability and less code reassignment cost. The main contribution of our scheme is to identify *linear-code chains* (LCCs) and *nonlinear-code trees* (NCTs) in the ROVSF code tree. This work exploits the *unsequence property* of *linear-code chains* to design a new code placement and replacement mechanism. Our code placement/replacement schemes initially attempt to allocate request codes on LCCs and, then, to allocate them to NCTs. Using LCCs with the *unsequence property* allows us to efficiently reduce the code blocking probability and the code reassignment cost. Extensive simulations are conducted to illustrate that our code placement/replacement results based on the ROVSF code tree actually improve the code blocking probability and the code reassignment cost.

Index Terms—Wireless communication, code assignment, code reassignment, OVSF, WCDMA.

1 Introduction

Wireless communication technology has recently and widely investigated significantly increasing mobile subscribers, traffic, and bandwidth-consuming applications, such as mobile learning [6], [7], mobile gaming, music down-loading, and wireless multimedia streaming. The capacity demand is supplied by the provision of new spectrum technology, namely, wideband code division multiple access or WCDMA, in order to create a global standard for real-time multimedia services and support international roaming. With the support of the International Telecommunication Union (ITU), a specific spectrum was allocated: 2 GHz for 3G telecom systems. The work was later taken over by the 3GPP (3rd Generation Partnership Project), which is now the WCDMA specification body with delegates from all over the world. With the rapid development of signal processing, it has become feasible to use this technology for wireless communication, also referred to as WCDMA and CDMA2000 [12], [16]. With CDMA2000, a 1.25-MHz-wide radio signal is multiplied by a spreading signal (which is a pseudonoise code sequence) with a higher rate than the data rate of the message. WCDMA has been selected for third-generation (3G) mobile telephone systems [1], [9], [13], while the universal mobile telecommunication system (UMTS) and WCDMA are often used as synonyms. First, UMTS is part of the ITU's IMT-2000 vision of a global family of 3G mobile communication systems. The 3GPP is developing technical specifications for IMT-2000, the ITU framework for 3G standards. The 3GPP is composed of ARIB (Japan), CWTS (China), ETSI (Europe), TI (USA), TTA (Korea), and TTC (Japan). In addition, WCDMA uses variable rate techniques in digital processing to achieve multirate transmissions.

In a WCDMA system, the spread spectrum is used to transmit multiple channels over a common bandwidth, and the capacity of the WCDMA system is limited by interference from other channels. Hence, in the 3G technical specifications, *orthogonal variable spreading factor* (OVSF) codes are usually selected to be the channelization codes. In 3GPP specifications [15], orthogonal codes (known as channelization codes) are used to preserve the orthogonality between users' physical channels. In total, 256 pieces of channelization codes are available [2], [11] and the spreading factor indicates how many bits of those codes are used in the connection. They are basically Walsh codes of different lengths that are able to preserve orthogonality between channels even when they are operating at different data rates. The OVSF codes are arranged in a tree structure for code allocation as described in Section 2. The OVSF channelization codes are widely used to provide variable data rates for supporting different bandwidth requirements in WCDMA systems. Much work in the literature [3], [4],

- *Y.-S. Chen is with the Department of Computer Science and Information Engineering, National Chung Cheng University, Chiayi, 621, Taiwan. E-mail: yschen@cs.ccu.edu.tw.*
- *T.-L. Lin is with Wistron Corporation, Taipei, 221, Taiwan. E-mail: peter_lin@wistron.com.tw.*

Manuscript received 30 Dec. 2003; revised 25 June 2004; accepted 14 Sept. 2004; published online 16 Jan. 2006.
For information on obtaining reprints of this article, please send e-mail to: tmc@computer.org, and reference IEEECS Log Number TMC-0225-1203.

[5], [8], [10], [14], [17], [18], [19] has intensively investigated *code placement* and *replacement* schemes with the objective of producing lower code blocking probability and less reassignment cost in the OVSF code tree.

In this paper, we initially introduce a new code tree structure, namely, the rotated-orthogonal variable spreading factor (ROVSF) code tree, whose code capacity is the same as that of the traditional OVSF code tree. This work addresses both code placement and replacement schemes in the ROVSF code tree system, where ROVSF codes are used at the forward link of the WCDMA system. Some new properties of the developed ROVSF code tree are presented to develop code placement and replacement schemes with lower code blocking probability and less code reassignment cost. The main idea of our scheme is to identify *linear-code chains* (LCCs) and *nonlinear-code trees* (NCTs) in the ROVSF code tree. This work exploits the *unsequence property* of LCCs to design a new code placement and replacement mechanism. The *unsequence property* is a very important result which offers the same allocation result for any different sequence of some new calls. Our code placement and replacement schemes initially attempt to allocate request codes on LCCs and, then, to allocate them to NCTs. Using LCCs with its *unsequence property* allows us to reduce the code blocking probability and the code reassignment cost. Finally, the simulation results illustrate that our code placement and replacement results based on the ROVSF code tree actually improve the code blocking probability and reduce the code reassignment cost.

The remainder of this paper is organized as follows: Section 2 reviews the background knowledge of this investigation. Section 3 introduces the definition and properties of the ROVSF code tree. The code placement and replacement schemes based on the ROVSF code tree are presented in Section 4. The performance analysis is given in Section 5 to demonstrate the performance improvement. Section 6 concludes this paper.

2 Background Knowledge

Since the number of orthogonal codes is limited in a WCDMA system, the code blocking problem was raised by Minn and Siu [14]. The code blocking problem is that a new call requesting a transmission rate cannot be supported because all available codes for the transmission rate are not orthogonal to other assigned codes, even if the system has sufficient remaining capacity. The high blocking probability is occurred if the new call requests higher transmission rate. More recently, several investigations of the management of channelization codes were carried out [3], [4], [5], [8], [10], [14], [17], [18], [19]. In this paper, a novel approach, namely, the ROVSF, is presented. Efforts are made in this investigation to reduce the code blocking probability and the code reassignment cost by developing the ROVSF-based scheme.

2.1 Related Work

Many existing results [3], [4], [5], [8], [10], [14], [17], [18], [19] are divided into OVSF-based and OVSF-like schemes. OVSF-based schemes have been thoroughly investigated as follows. Chao et al. proposed single-code and multicode placement and replacement schemes in WCDMA systems [3], [19]. The algorithm for single-code placement/replacement is simple in [19], but it possibly incurs many fragmented codes which produce a code blocking problem. The multi-OVSF code placement and replacement scheme [3] can actually reduce the code blocking problem by using a code-separation operation. Minn and Siu [14] developed a dynamic assignment of OVSF codes to provide an optimal dynamic code assignment (DCA) scheme, which reassigns codes with minimum costs. Cheng and Lin [8] proposed an OVSF code channel assignment for IMT-2000, whose objective is to provide the multirate service with less complexity. Chen et al. [5] partitioned codes into two groups based on the code capacity. When a code is released, the "right-most" or "left-most" request in the same layer will be rearranged to the "just-released" code. As a result, large unnecessary code reassignment costs may exist. In addition, the hybrid multicode (MC) strategies based on the concept of *tree partitioning* were investigated in [10]. Recently, Chen and Chen [4] proposed an efficient best-fit least recently used (BLRU) code assignment algorithm to efficiently reduce the code fragmentation problem [3], [19]. An alternative scheme was presented by Rouskas and Skoutas [17] for OVSF code assignment and reassignment schemes at the forward link of WCDMA 3G systems, which behave similar to *crowded-first* and *mostuser-first* schemes in [3], [19].

Tsaur and Lee [18] developed symbol rate adaptation and blind rate detection using a forest for OVSF-sequence-set-inducing lineage (FOSSIL). The rate information is implied between some codes without occupying extra bandwidth. A new code-tree was developed to provide code sequences with different lengths for different users who are communicating at different constant symbol rates. Observe that FOSSIL [18] has more usable codes than that of the OVSF code tree. It is worthwhile developing a new OVSF-like scheme which can efficiently improve system performance. In this work, we attempted to develop a completely new code tree.

2.2 OVSF Code Tree

In a WCDMA system [1], [2], [3], [19], two operations are applied to user data. The first one is channelization, which transforms every bit into a code sequence. The length of the code sequence per data bit is called the spreading factor (SF), which is typically a power of two. The second operation is scrambling, where a scrambling code is applied to the spread signal. Scrambling codes are used to separate the signal from different sources, while channelization codes are used to separate transmission from a single source.

The OVSF channelization codes preserve orthogonality between a user's different physical channels. The possible OVSF codes can be represented by a code tree as illustrated in Fig. 1. Channelization codes in the OVSF code tree have a unique description as $C_{SF,k}$, 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. Channelization codes of equal length and those which differ in length but none is the ancestor of the other are orthogonal

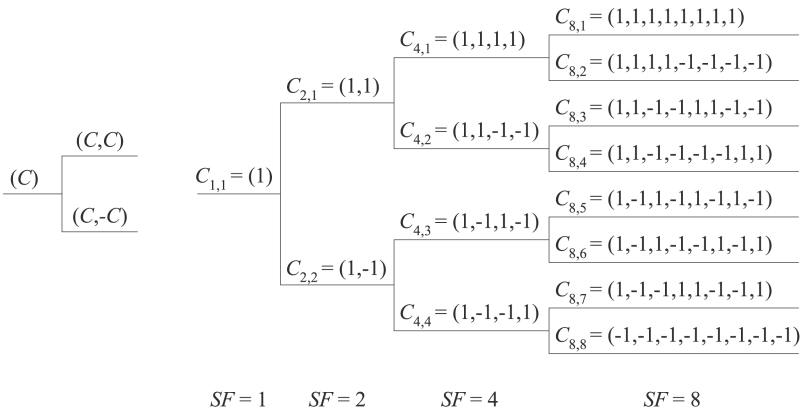

Fig. 1. An orthogonal variable spreading factor (OVSF) code-tree.

to each other. That is, none of the codes in the path from the target code to the root of the tree is already in use in a cell. As illustrated in Fig. 1, the cross-correlation is occurred if codes $C_{4,2}$ and $C_{8,3}$ are used. Codes $C_{4,1}$ and $C_{8,3}$ on different branches are uncorrelated.

3 Rotated-OVSF Code Tree

In this paper, we define a new code tree, namely, rotated OVSF (ROVSF) code tree, and propose code placement/replacement algorithms on the ROVSF code tree. Several novel properties of the ROVSF code tree are investigated in this paper. 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. In this work, the code $RC_{SF,k}$ is called as ROVSF code. Observe that 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. *Given a root node of the ROVSF code tree, two children codes, $RC_{2,1}$ and $RC_{2,2}$, of the root node are initially set to $(-1,-1)$ and $(-1,1)$, respectively. Assume that a pair of neighboring ROVSF codes $RC_{i,j}$ and $RC_{i,j-1}$, at the kth level of the ROVSF code tree, are A and B, where $1 \leq j-1 < i = 2^k$ and j is the even number. Two children codes of $RC_{i,j}$ at the $(k+1)$th level of the ROVSF code tree are $RC_{2i,2j-1} = (-B,-B)$ and $RC_{2i,2j} = (B,-B)$. Two children codes of $RC_{i,j-1}$ are $RC_{2i,2j-2} = (A,-A)$ and $RC_{2i,2j-3} = (-A,-A)$.*

Fig. 2 shows an ROVSF code tree. Two codes, (P,Q) and (R,S), are said to be brother codes if $Q = S$ and P is the complement of R, i.e., $P = -R$. The key feature of the ROVSF-based scheme is the existence of *linear-code chain* and *nonlinear-code tree*, as defined below.

3.1 Linear-Code Chain (LCC) and Nonlinear-Code Tree (NCT)

Every node of the OVSF code tree is logically mapped to the corresponding node of the ROVSF code tree. An example is illustrated in Fig. 3. These mapping nodes form a code chain, which is called a *linear-code chain* (LCC). One or more LCCs may exist in the ROVSF code tree. LCCs can be divided into *complete* and *partial* LCCs. Consider the case where a new call arrives requesting a rate γR, where γ is power of 2. No free code of rate γR exists in the *complete* LCC, but it may allocate a free code of rate γR to the *partial* LCC. The *complete* LCC is defined as follows:

Definition 2. Complete Linear-Code Chain (Complete LCC). *Given a code set $S_k = [R_{\max}, \frac{R_{\max}}{2^1}, \frac{R_{\max}}{2^2}, \cdots, \frac{R_{\max}}{2^\alpha}, \cdots, \frac{R_{\max}}{2^k}]$, where $R_{\max} = 2^{\log_2 R_{\max}}$ (called the chain-max-code) and $0 \leq \alpha < k \leq \log_2(R_{\max})$. Let the complete LCC be denoted as $S = S_k \cup \{\frac{R_{\max}}{2^k}\} = [R_{\max}, \frac{R_{\max}}{2^1}, \frac{R_{\max}}{2^2}, \cdots, \frac{R_{\max}}{2^\alpha}, \cdots, \frac{R_{\max}}{2^k}, \frac{R_{\max}}{2^k}]$, where a pair of brother codes, $\frac{R_{\max}}{2^k}$ and $\frac{R_{\max}}{2^k}$, is on the same level of the ROVSF code tree.*

For example, given a five-layer ROVSF code tree as shown in Fig. 4, $[8R, 4R, 2R, 1R, 1R]$, $[8R, 4R, 2R, 2R]$, and $[8R, 4R, 4R]$ are complete LCCs as illustrated in Fig. 4b, Fig. 4d, and Fig. 4f, respectively. No free code exists in $[8R, 4R, 2R, 1R, 1R]$, $[8R, 4R, 2R, 2R]$, or $[8R, 4R, 4R]$. Next, the *partial linear-code chain* or *partial* LCC is defined herein.

Defnition 3. Partial Linear-Code Chain (Partial LCC). *Given a complete LCC, S, a partial linear-code chain, S', is approximately equal to S, but at least one code exists in S which does not exist in S'.*

Let $S' = [R_{\max}, \frac{R_{\max}}{2^1}, \frac{R_{\max}}{2^2}, \cdots, \frac{R_{\max}}{2^\alpha}, \cdots, \frac{R_{\max}}{2^k}]$ be a partial LCC, where $0 \leq \alpha \leq k \leq \log_2(R_{\max})$. When a new call arrives requesting a code of rate $\frac{R_{\max}}{2^k}$, the code of rate $\frac{R_{\max}}{2^k}$ can be allocated to the partial LCC, S'. For instance, $[8R, 4R, 2R, 1R]$, $[8R, 4R, 2R]$, $[8R, 4R]$, and $[8R]$ are partial LCCs as illustrated in Fig. 4a, Fig. 4c, and Fig. 4e, respectively. A requesting code of rate $1R$ can be allocated to $[8R, 4R, 2R, 1R]$, $[8R, 4R, 2R]$, $[8R, 4R]$, or $[8R]$. For another example, $[R_{\max}, \frac{R_{\max}}{2^2}, \frac{R_{\max}}{2^5}]$, $[8R, 4R, 1R]$, and $[8R, 2R, 2R]$ are partial LCCs.

Suppose a new call arriving requests a rate γR, the main idea of the ROVSF code placement and replacement scheme is to allocate a free code of rate γR to partial LCCs. To provide a simple code assignment mechanism, we maintain a bit-word mechanism to keep the free code information for

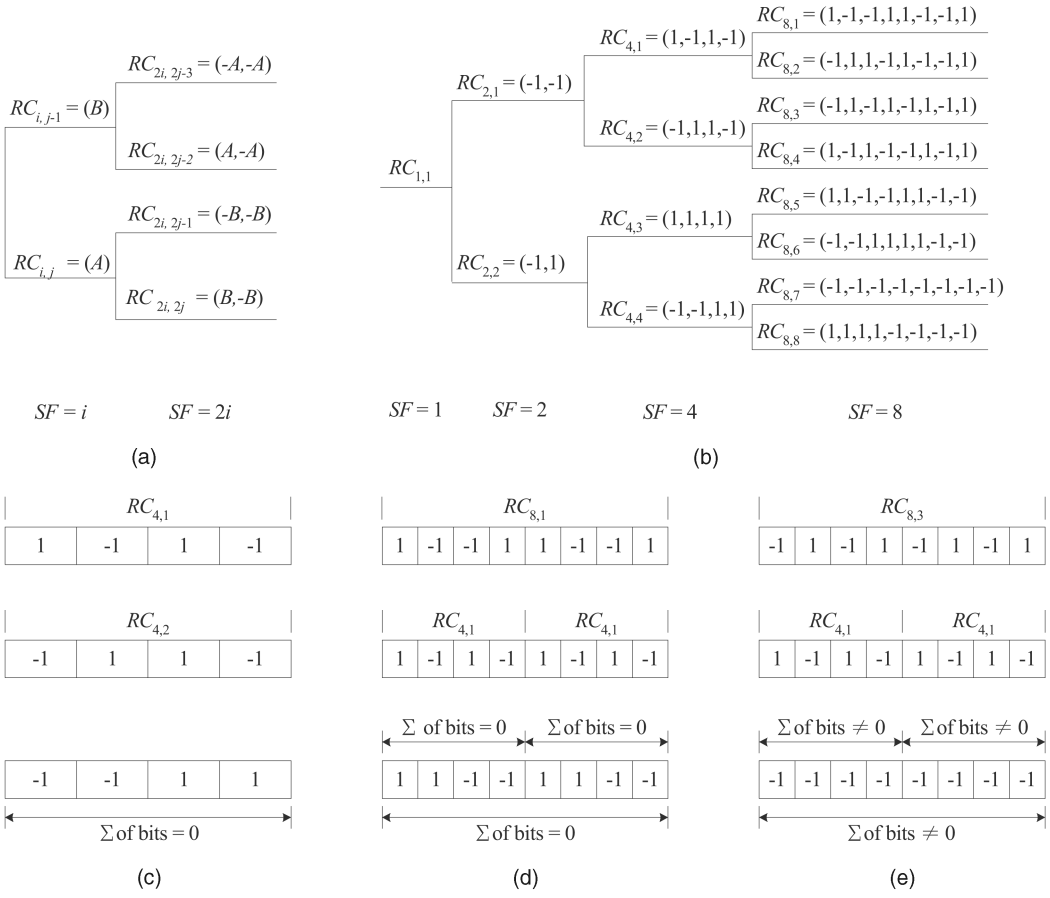

Fig. 2. A rotated OVSF (ROVSF) code-tree.

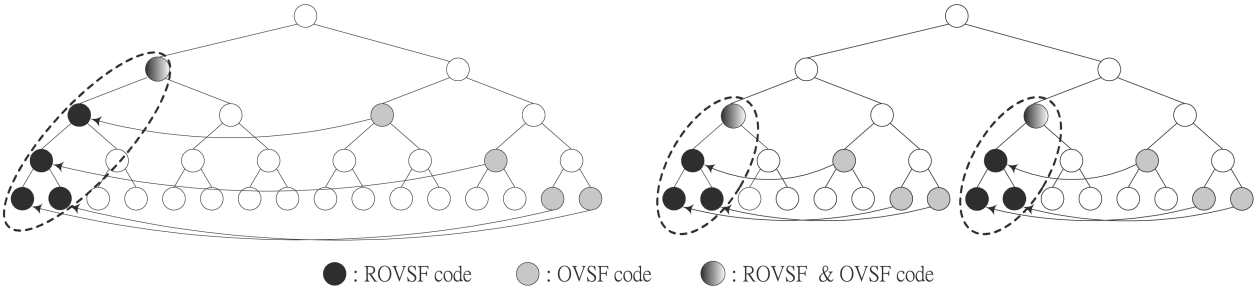

Fig. 3. Relationship of ROVSF and OVSF code trees.

LCCs. Each complete or partial LCC maintains one bit-word. It can easily check the information on the allocation status from these bit-words. Given a complete LCC, $S = [R_{\max}, \frac{R_{\max}}{2^1}, \frac{R_{\max}}{2^2}, \cdots, \frac{R_{\max}}{2^j}, \frac{R_{\max}}{2^j}]$, the bit-word $BW = (b_k, b_{k-1}, b_{k-2}, \cdots, b_i, \cdots, (b_j, b_j))_{k+2}$ is used to represent the complete linear-code chain, S, where $b_i = b_j = 1$ and $0 \leq j < i \leq k = \log_2(R_{\max})$. This bit-word has at most $k+2$ bits, where $k = \log_2(R_{\max})$. For instance, $BW = (1, 1, (1, 1))_5$ is used for the complete LCC $= [8R, 4R, 2R, 2R]$. For a partial LCC, $BW = (b_k, b_{k-1}, b_{k-2}, \cdots, b_i, \cdots, (b_j, b_j))_{k+2}$ is used, where b_i and $b_j = 0$ or 1, and $0 \leq j < i \leq k = \log_2(R_{\max})$. That is, if $b_i = 1$ or $b_j = 1$, then $\frac{R_{\max}}{2^i}$ or $\frac{R_{\max}}{2^j}$ exists, and, if $b_i = 0$ or $b_j = 0$, then $\frac{R_{\max}}{2^i}$ or $\frac{R_{\max}}{2^j}$ does not exist. If a new call arrives requesting a rate $\frac{R_{\max}}{2^i}$ or $\frac{R_{\max}}{2^j}$, we can allocate free code of rate $\frac{R_{\max}}{2^i}$ or $\frac{R_{\max}}{2^j}$ to the partial LCC if $b_i = 0$ or $b_j = 0$, respectively. For instance, $BW = (1, 1, 0, (1, 0))_5$ and $BW = (1, 0, (1, 1))_5$ are used for $[8R, 4R, 1R]$ and $[8R, 2R, 2R]$, respectively.

Suppose a new call arriving requests a rate γR; if there is no free code of rate γR in the LCCs, we introduce the *nonlinear-code tree* (NCT) to allocate a free code for the new call.

Definition 4. Nonlinear-Code Tree (NCT). *Given a complete* LCC $= [R_{\max}, \frac{R_{\max}}{2^1}, \frac{R_{\max}}{2^2}, \cdots, \frac{R_{\max}}{2^i}, \cdots, \frac{R_{\max}}{2^j}, \frac{R_{\max}}{2^j}]$, *where* $0 \leq i < j$, *if* $R = \frac{R_{\max}}{2^i}$ *and the brother code of* R *is* R', *a subtree rooted at* R' *is denoted a nonlinear-code tree (NCT).*

Examples of NCTs are given in Fig. 5. Fig. 5a shows a complete LCC; no NCT is needed since there is no

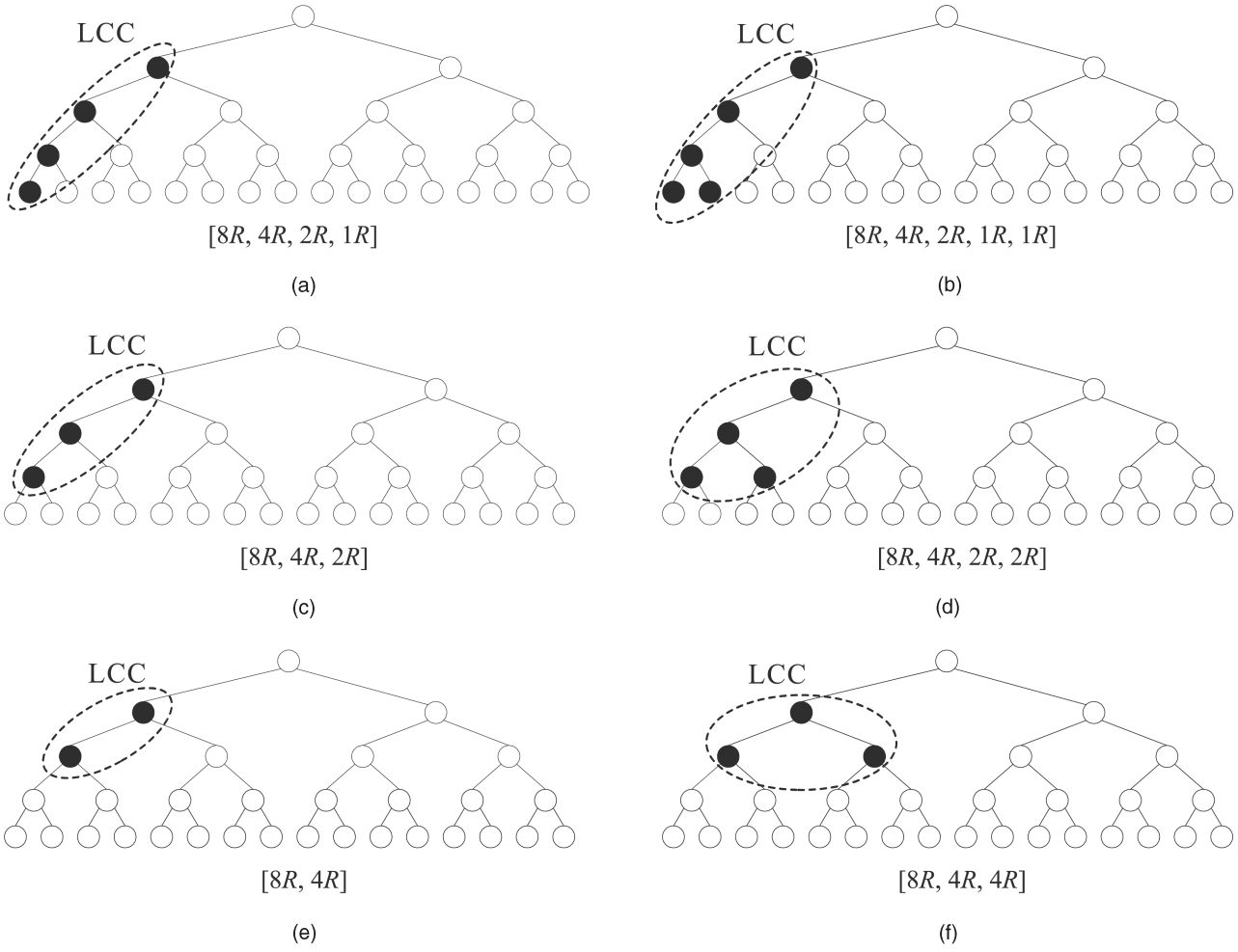

Fig. 4. Examples of partial and complete LCCs.

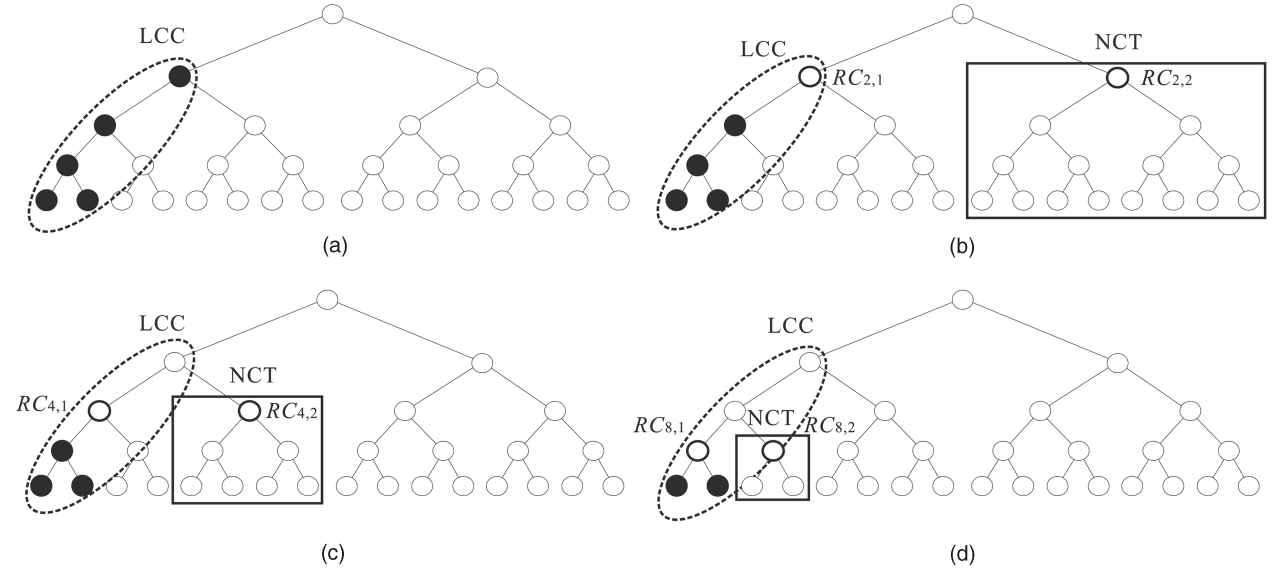

Fig. 5. Examples of NCTs.

remaining capacity of the ROVSF code tree. Given three partial LCCs as shown in Fig. 5b, Fig. 5c, and Fig. 5d, three NCTs whose root nodes are $RC_{2,2}$, $RC_{4,2}$, and $RC_{8,2}$ are illustrated in Fig. 5b, Fig. 5c, and Fig. 5d, respectively. If we cannot allocate any free code to the partial LCCs, then an attempt is made to allocate a free code to NCTs. The detailed placement operations are described in Section 4.

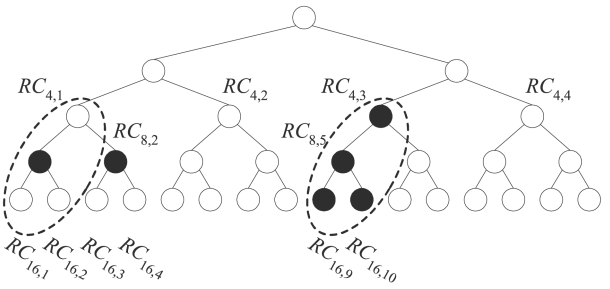

Fig. 6. Properties of an ROVSF code tree.

3.2 ROVSF Properties

Some important properties of ROVSF code trees are introduced as follows:

P1. The ROVSF code tree and the OVSF code tree have the same code capabilities.

P2. The maximum data rate in an n-layer ROVSF code tree is $2^{n-1}R$.

P3. Any pair of two different ROVSF codes, $RC_{i,j}$ and $RC_{i,j'}$, at the kth level of the ROVSF code tree, is orthogonal, where $i = 2^{k-1}$ and $j \neq j'$. For instance, as shown in Fig. 2c, $RC_{4,1}$ is orthogonal to the same level code $RC_{4,2}$ and $RC_{4,3} = (1, 1, 1, 1)$ is orthogonal to $RC_{4,4} = (-1, -1, 1, 1)$. Similarly, $RC_{4,1}$ is orthogonal to $RC_{4,3}$ or $RC_{4,4}$ and $RC_{4,2}$ is orthogonal to $RC_{4,3}$ or $RC_{4,4}$.

P4. An ROVSF code, $RC_{i,j}$, is cyclically orthogonal to its two children codes, $RC_{2i,2j}$ and $RC_{2i,2j-1}$. For instance as shown in Fig. 2d, $RC_{4,1}$ is cyclically orthogonal to $RC_{8,1}$. Similarly, $RC_{4,1} = (1, -1, 1, -1)$ is cyclically orthogonal to $RC_{8,2} = (-1, 1, 1, -1, 1, -1, -1, 1)$.

P5. An ROVSF code, $RC_{i,j}$, is cyclically orthogonal to any descendant codes of $RC_{i,j}$. This property is obtained from Property 4 according to the transitive property.

P6. Given a pair of brother codes, $RC_{i,j} = (-A, -A)$ and $RC_{i,j-1} = (A, -A)$, $RC_{i,j}$ (or $RC_{i,j-1}$) is not cyclically orthogonal to any descendant codes of $RC_{i,j-1}$ (or $RC_{i,j}$). Observe that this property of the ROVSF code tree should be referred to as the blocked ancestor codes as well. For instance, as shown in Fig. 2e, the cross-correlation is occurred between codes $RC_{4,1}$ and $RC_{8,3}$.

P7. For an n-layer ROVSF code tree, $2^{n-\alpha-1}$ LCCs exist, where *chain-max-code* $R_{\max} = 2^{\alpha-1}R$ and the initial value of α is usually assumed to be $\lceil \frac{n}{2} \rceil$. The effects of system performance with different values of α are discussed in the performance analysis.

For example, consider a five-layer ROVSF code tree as illustrated in Fig. 6. By **P7**, there exists $2^{n-\alpha-1} = 2^{5-3-1} = 2$ LCCs, where *chain-max-code* is $R_{\max} = 2^{\alpha-1}R = 2^{3-1}R = 4R$ and $\alpha = \lceil \frac{5}{2} \rceil$. By **P3**, each of $RC_{4,1}$, $RC_{4,2}$, $RC_{4,3}$, and $RC_{4,4}$ is orthogonal to the other. By **P4** and **P5**, codes $RC_{4,3}$, $RC_{8,5}$, $RC_{16,9}$, and $RC_{16,10}$ are orthogonal since $RC_{4,3}$ is cyclically orthogonal to its children codes and $RC_{8,5}$ is cyclically orthogonal to its children codes. By **P6**, $RC_{16,1}$, $RC_{16,2}$, $RC_{16,3}$, and $RC_{16,4}$ cannot be allocated since $RC_{8,1}$ and $RC_{8,2}$ are already occupied. All descendant codes of $RC_{4,2}$ can be used since $RC_{4,1}$ is not occupied. But, none of the codes in the entire subtree of $RC_{4,4}$ can be allocated since $RC_{4,3}$ is already occupied.

3.3 The Unsequence Allocation Property of LCCs

To reduce the code blocking probability and the code reassignment cost, an LCC, which is a collection of mutually exclusive ROVSF codes, is exploited. Two main differences between OVSF-based and our ROVSF-based schemes are 1) the restriction of the fixed data rate and 2) the unsequence allocation property. First, the restriction of the fixed data rate of our approach limits the fixed length of LCCs, where the length is denoted α. Any new call with a transmission request, γR, where $\gamma > R_{\max} = 2^{\alpha-1}$, is possibly blocked although the remaining capacity is sufficient. For example, as shown in Fig. 7a, it cannot allocate a free code of $8R$ to LCCs since $R_{\max} = 2^{\alpha-1} = 2^{3-1} = 4$. With different lengths of LCCs, we can allocate a free code $RC_{2,1}$ of $8R$ to an LCC as shown in Fig. 7b since $R_{\max} = 2^{\alpha-1} = 2^{4-1} = 8$. To reduce the restriction of the fixed data rate, a dynamic operation of adjusting the length of LCCs is investigated in this work.

Observe that, if data rate of all new calls $< R_{\max} = 2^{\alpha-1}$, our code placement scheme results in lower code blocking probability than the scheme in [19], which is clarified in Section 5. The reason is that our new ROVSF-based scheme offers the *unsequence allocation* property of LCCs, which is the main contribution of this work. Efforts will be made to drastically utilize the unsequence property to reduce the code blocking probability.

As illustrated in Fig. 3, allocations of $8R$, $4R$, $2R$, $1R$, and $1R$ in OVSF and ROVSF code trees are given. In this example, observe that any permutation of the data request sequence $[8R, 4R, 2R, 1R, 1R]$ must be allocated to one of the LCCs. However, it is observed that any permutation of the data request sequence $[8R, 4R, 2R, 1R, 1R]$ may be improperly allocated to the OVSF code tree. A code blocking problem may occur, which is very dependent on the order of incoming data requests, even if one very carefully uses the code placement scheme on the OVSF code tree. If only one LCC exists, then we may call any permutation of $[8R, 4R, 2R, 1R, 1R]$ the *perfect allocation pattern*. If there are two LCCs, then the *perfect allocation pattern* is any permutation of $[4R, 4R, 2R, 2R, 1R, 1R, 1R, 1R]$. Any perfect allocation pattern can properly be assigned to LCCs. Therefore, this property is called the *unsequence allocation property* of LCCs. If a data request sequence possesses a perfect allocation pattern which can be fit into LCCs, then the code blocking probability can obviously be reduced. For example, as shown in Fig. 7c, $[2R, 1R, 1R]$, $[1R, 1R, 2R]$, and $[1R, 2R, 1R]$ are possibly assigned into different codes in the OVSF code tree. To utilize the unsequence property, $[2R, 1R, 1R]$, $[1R, 1R, 2R]$, or $[1R, 2R, 1R]$ can be allocated into a unique LCC as illustrated in Fig. 7d. Following the same example, as shown in Fig. 7e, $[2R, 2R]$ cannot be allocated to the OVSF code tree, but $[2R, 2R]$ can be allocated to our ROVSF code tree as shown in Fig. 7f. The second data request with $2R$ is blocked on the OVSF code tree, but it can be fitted into our ROVSF code tree.

Note that not all new calls can be assigned to LCCs and not all new calls possess a perfect allocation pattern. However, an attempt will be made to assign any incoming

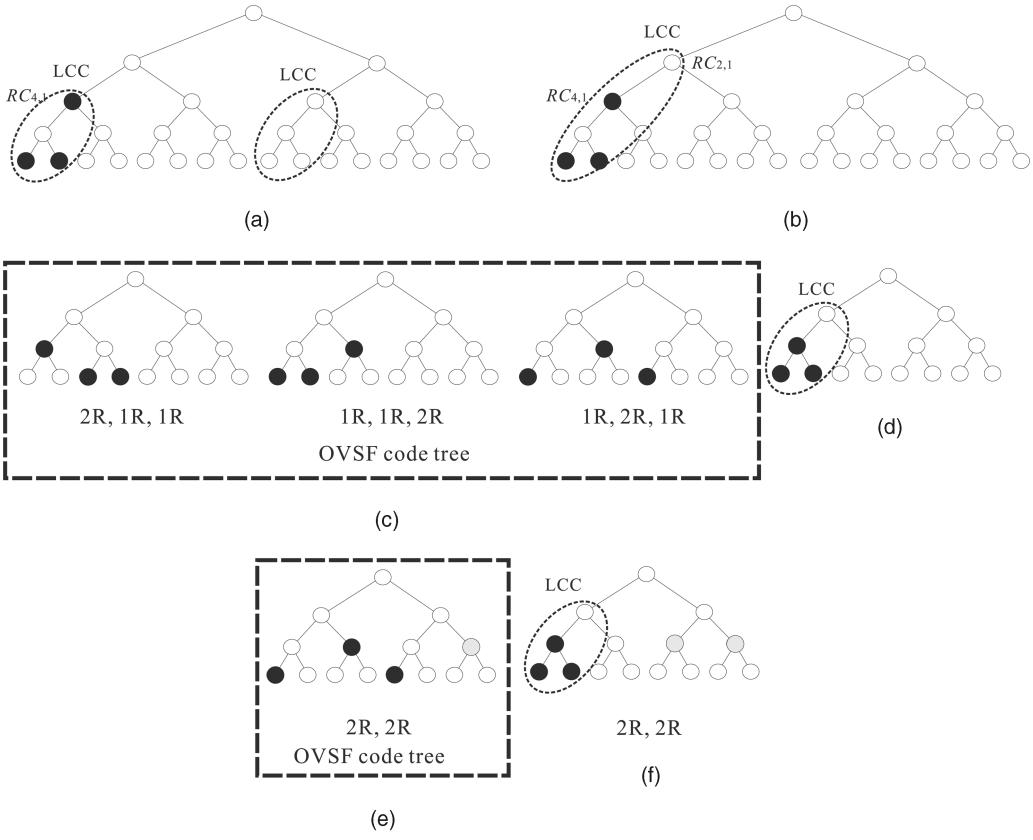

Fig. 7. The key difference between ROVSF and OVSF schemes.

data request, which does not posses a perfect allocation pattern to NCTs. It is still possible to take other codes from the ROVSF code tree. In addition, we discuss the impact of code blocking rate versus the utilization of allocation of LCCs in Section 5.

4 Code Placement and Replacement Strategies of ROVSF Code Trees

In this section, we present the code placement and replacement schemes of the ROVSF code tree. The code placement scheme is divided into two phases: linear-code chain (LCC) placement and nonlinear-code tree (NCT) placement phases. The code placement scheme with lower code blocking probability was investigated with the limitation of the fixed lengths of LCCs. The code replacement scheme, based on a modified DCA algorithm [14], was developed to eliminate the limitation of the fixed lengths of LCCs. A dynamic adjustment operation is presented in the replacement scheme to dynamically adjust the length of the LCC, which aims to support any rate of incoming requests.

4.1 Code Placement Scheme

Suppose a new call arriving requests a rate γR, where γ is a power of 2. The main idea of the ROVSF code placement scheme is to initially assign a free code of rate γR to LCCs. If we fail to find any feasible code in the LCCs, we then try to find a possible free code from the NCTs. The main purpose of using the LCC is to reduce the code blocking probability. Observe that utilization of the LCC significantly impacts the code blocking probability. Basically, the more the LCC is utilized, the lower the code blocking probability will be. The code placement scheme is divided into LCC and NCT phases as follows.

4.1.1 Linear-Code Chain (LCC) Placement Phase

Consider an n-layer ROVSF code tree, for which $2^{n-\alpha-1}$ LCCs exist, where the chain-max-code $R_{\max} = 2^{\alpha-1}R$, where $\alpha = \lceil \frac{n}{2} \rceil$. If a new call arriving requests a rate γR, the LCC placement phase attempts to allocate a free code of a rate γR to one of the $2^{n-\alpha-1}$ LCCs from left to right. Since each linear-code chain has its own bit-word, BW, this work can be achieved by checking the bit-word sequence $[BW_1, BW_2, \cdots, BW_{2^{n-\alpha-1}}]$. A *leftmost-assignment* strategy is performed as follows. We initially try to allocate a free code of rate γR to BW_1. If this fails, we continue by trying BW_2. We repeatedly execute the above operation until a free code of rate γR can be allocated to BW_j, where $j \leq 2^{n-\alpha-1}$. If we still cannot allocate a free code of rate γR to $BW_{2^{n-\alpha-1}}$, then we perform the NCT placement phase, which is described later.

Now, we further describe how to allocate a free code of rate γR to an LCC with BW, where γ is power of two, $BW = (b_k, b_{k-1}, b_{k-2}, \cdots, b_i \cdots, (b_j, b_j))_{k+2}$, b_i and $b_j = 0$ or 1, $0 \leq j < i \leq k = \log_2(R_{\max})$, and BW is one among $BW_1, BW_2, \cdots,$ and $BW_{2^{n-\alpha-1}}$. Let $\beta = \log_2 \gamma$, where γR is the requesting data rate. Observe that, if the LCC keeps $BW = (b_k, b_{k-1}, b_{k-2}, \cdots, (b_j, b_j))_{k+2}$, where $b_j = 1$, then we cannot allocate a free code of rate γR to the LCC if $0 \leq \beta < j$, where $\beta = \log_2 \gamma$. But, we can allocate a free

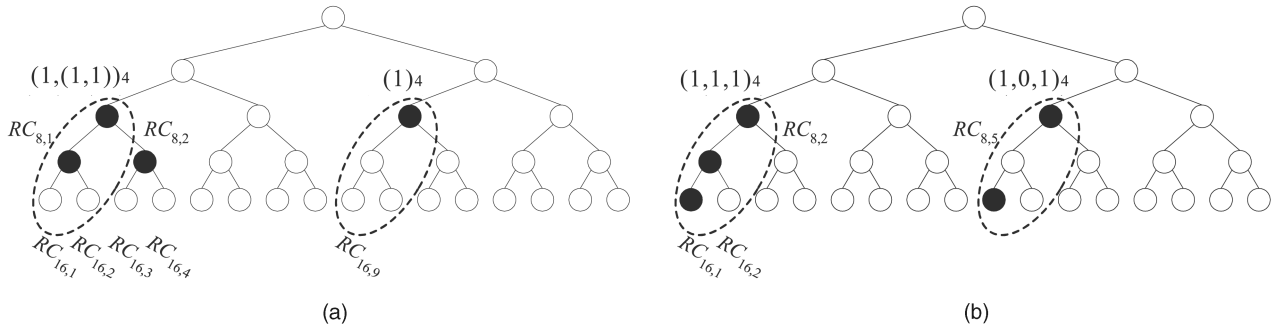

Fig. 8. Example of the LCC placement phase.

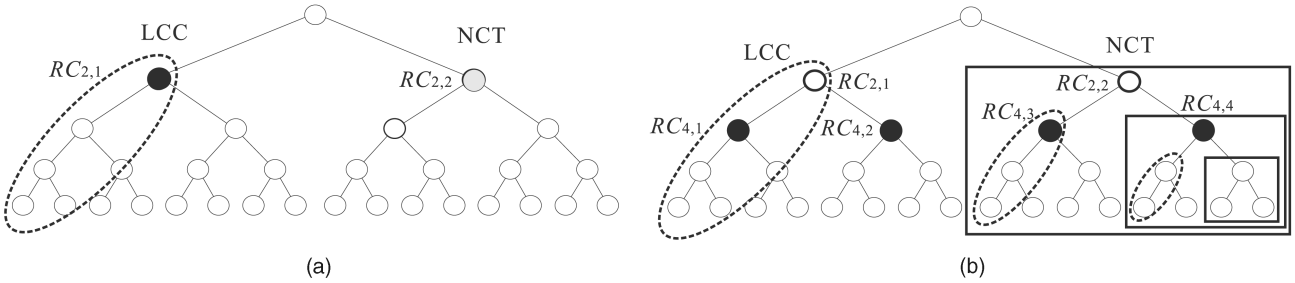

Fig. 9. Examples of NCTs.

code of rate γR to the LCC if $k \geq \beta \geq j$ and $b_\beta = 0$. A variable ROVSF_REMAINING_CAPACITY denotes the remaining capacity of the ROVSF code tree. The formal algorithm of the LCC placement operation is given in the following steps:

1. If $\gamma R >$ ROVSF_REMAINING_CAPACITY, then exit the LCC placement algorithm.
2. Repeatedly perform the following operations to allocate a free code of rate γR to the ith LCC with bit-word BW_i, where $1 \leq i \leq 2^{n-\alpha-1}$ and $\beta = \log_2 \gamma$.
   a. Try to assign a free code of rate γR to the ith LCC by changing BW_i from $(b_k, b_{k-1}, b_{k-2}, \cdots, (b_j = 1, b_j = 0))_{k+2}$ or $(b_k, b_{k-1}, b_{k-2}, \cdots, (b_j = 0, b_j = 1))_{k+2}$ to $(b_k, b_{k-1}, b_{k-2}, \cdots, (b_j = 1, b_j = 1))_{k+2}$ if $\beta = j$, where $k = \log_2(R_{\max})$. If the assignment is successful, then ROVSF_REMAINING_CAPACITY = ROVSF_REMAINING_CAPACITY $-xR,$, and exit the LCC placement algorithm.
   b. If $(b_k, b_{k-1}, b_{k-2}, \cdots, (b_j = 1, b_j = 1))_{k+2}$ exists and $\beta < j,$ the assignment fails; then go to **Step 2** for assigning a free code of rate γR to the $(i+1)$th LCC. If $\beta > j$ and $b_\beta = 0,$ then the assignment is successful, let ROVSF_REMAINING_CAPACITY = ROVSF_REMAINING_CAPACITY $- xR,$ and exit the LCC placement algorithm.
3. If we still cannot allocate a free code of rate γR to the last LCC with bit-word $BW_{2^{n-\alpha-1}}$, then execute the NCT placement phase.

For example, consider the ROVSF code tree in Fig. 8a. Assume that there is a new call with a rate requirement $= 1R$. We cannot allocate $RC_{16,1}$ for rate $1R$ on the left LCC with bit-word $(1,(1,1))_4$ since $RC_{8,2}$ is already allocated; thus, $RC_{16,9}$ is used in the right LCC with bit-word $(1)_4$. Another example, shown in Fig. 8b, is of a new incoming call with date rate $2R$, for which $RC_{8,5}$ is allocated, since the right LCC has bit-word $(1,0,1)_4$.

4.1.2 Nonlinear-Code Tree (NCT) Placement Phase

If we cannot allocate any free code of rate γR to all LCCs, then an attempt will be made to assign a free code of rate γR to NCTs. An NCT can be partitioned into a pair comprised of a small-sized LCC and a small-sized NCT. An NCT partition shown in Fig. 9b, the *chain-max-code* of the small-sized LCC is $4R$, and the root of the small-sized NCT is $RC_{4,4}$. If we want to allocate a free code of rate γR to an NCT, then we use the LCC code placement strategy on the small-sized LCC. If we cannot allocate a free code of rate γR to the small-sized LCC, then we recursively try the small-sized NCT.

It should be noted that an LCC has one or more NCTs. To simplify the description, we first consider that each LCC has one NCT. Consider a pair comprised of an LCC and an NCT, let the *chain-max-code* R of the LCC be $R_{\max}$ and the root node of the NCT be R', and R and R' be a pair of brother codes. Therefore, there are $2^{n-\alpha-1}$ LCCs and $2^{n-\alpha-1}$ NCTs. Before introducing how to allocate a free code of rate γR to $2^{n-\alpha-1}$ NCTs, some properties of NCTs are stated.

R1. By **P6**, if R is in use, then there is no free code, except for R', that can be assigned to NCT. Code R' in the NCT can be allocated if the bit-word of LCC is $(1)_{\log_2(R_{\max})+2}$.
R2. If R and R' are free, then the NCT is recursively split into a pair comprised of a small-sized LCC and a small-sized NCT, where *chain-max-code* $R'_{\max}$ of the small-sized LCC is $\frac{R}{2}$. We attempt to allocate a free

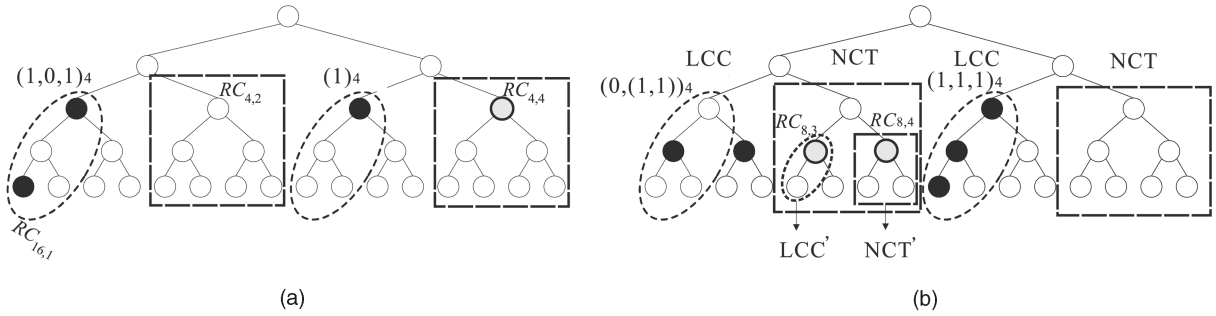

Fig. 10. Example of the NCT placement phase.

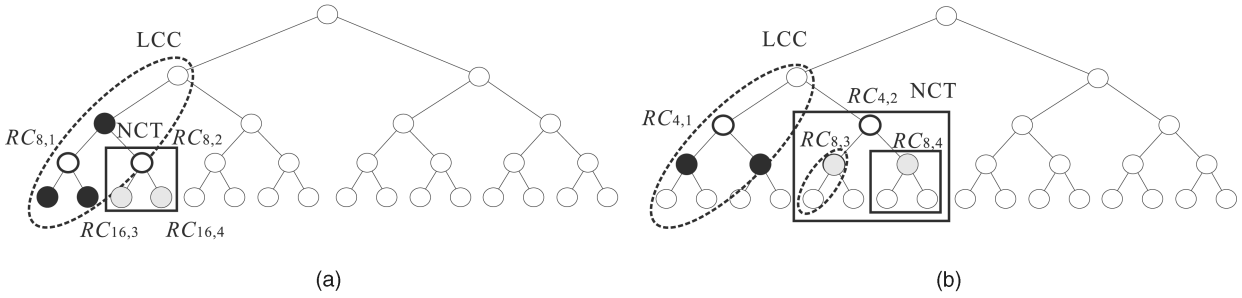

Fig. 11. Examples of NCTs.

code of rate γR to the small-sized LCC. If it fails, we continue to try to allocate a free code of rate γR to the small-sized NCT by recursively performing **R1** and **R2**.

For instance, as illustrated in Fig. 9a, since $RC_{2,1}$ in the LCC is used, none of the codes of the NCT, except for $RC_{2,2}$, can be used, and $RC_{2,2}$ in the NCT can be assigned since the bit-word of the LCC is $(1)_5$. Fig. 9b shows an NCT partition such that $RC_{4,1}$ and $RC_{4,2}$ can be assigned to the LCC and $RC_{4,3}$ and $RC_{4,3}$ can be assigned to the NCT.

If it fails to find a free code of rate γR in the LCC placement phase, then the NCT placement phase is performed to allocate a free code of rate γR to one of the NCTs. Given $2^{n-\alpha-1}$ LCCs and $2^{n-\alpha-1}$ NCTs, the formal algorithm of the NCT placement operation is given in the following steps:

1. If $\gamma R >$ ROVSF_REMAINING_CAPACITY, then exit the NCT placement algorithm.
2. Repeatedly perform the following procedure to allocate a free code of rate γR to the ith NCT, where the ith LCC's bit-word

$$BW_i = (b_k, b_{k-1}, b_{k-2}, \cdots, b_l, \cdots, (b_j, b_j))_{k+2,}$$

where $1 \le i \le 2^{n-\alpha-1}$, $\beta = \log_2 \gamma$, and $k = \log_2(R_{\max})$.

   a. If the ith LCC's bit-word BW_i is $(1)_{k+2}$, then it can assign γR to the ith NCT and γR is the root of the NCT.
   b. If the ith LCC's bit-word BW_i is

$$(b_k, b_{k-1}, b_{k-2}, \cdots, b_l, \cdots, (b_j, b_j))_{k+2},$$

   then do the following operations:

   - If $b_k = 1$ and $b_l = 1$ or $b_j = 1$, then increase the value of i and go to **Step 2**.
   - If $b_k = 0$, then split the ith NCT into a pair comprised of a small-sized LCC and a small-sized NCT. We find a free code of rate γR into the small-sized LCC using the same rule of the LCC placement algorithm. If it fails, we attempt to find a free code of rate γR into the small-sized NCT by recursively executing similar operations of **Step 2a** and **Step 2b**. After the recursive execution, if we cannot find a free code of rate γR to the ith NCT, then increase the value of i and go to **Step 2**.

3. If we still cannot find a free code of rate γR to the NCT of the last LCC, then go to **Step 4** and **Step 5**.

Consider an ROVSF code tree as shown in Fig. 10a. A new call with a rate requirement of $4R$ arrives; the NCT code placement operation fails for the first NCT, since the left LCC's bit-word is $(1,0,1)_4$. Therefore, $4R$ is assigned to $RC_{4,4}$, since the right LCC's bit-word is $(1)_4$. Another example is given in Fig. 10b, in which the left LCC has bit-word $(0,(1,1))_4$. A new call with rate requirement, $2R$, is coming, so $RC_{8,3}$ is assigned to a small-sized LCC. Then, a new call with rate requirement, $2R$, is coming again, then $RC_{8,4}$ is assigned to the root of a small-sized NCT.

It is worth discussing the case when we cannot find free code of rate γR to $2^{n-\alpha-1}$ LCCs and $2^{n-\alpha-1}$ NCTs. Then, we may try to allocate a free code of rate γR to other NCTs of $2^{n-\alpha-1}$ LCCs. To simplify the discussion, we consider one LCC among $2^{n-\alpha-1}$ LCCs, where the LCC has bit-word $(b_k, b_{k-1}, b_{k-2}, \cdots, b_l, \cdots, (b_j, b_j))_{k+2}$, for $j < l < k$. In addition, there are at most $k - j - 1$ NCTs for the LCC. For instance, as shown in Fig. 11, two NCTs exist. If we cannot find a free code of rate γR to $2^{n-\alpha-1}$ LCCs and $2^{n-\alpha-1}$ NCTs, then we allocate a free code of rate γR to $2^{n-\alpha-1} \times (k - j - 1)$ NCTs as follows.

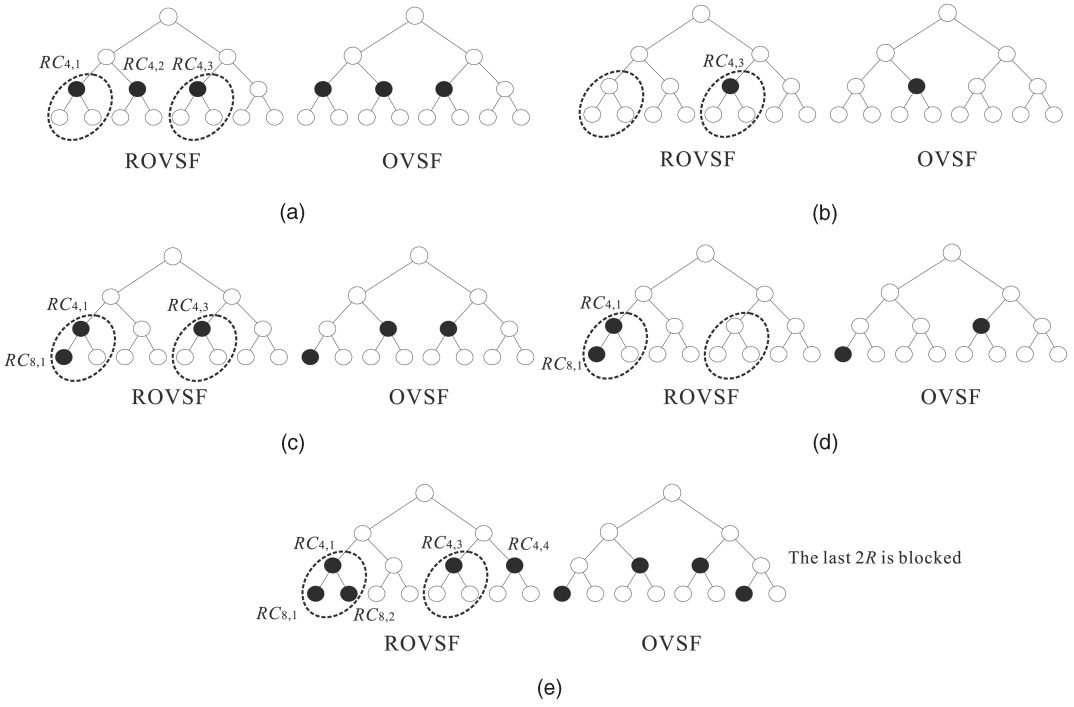

Fig. 12. Example of code placement using our scheme and the OVSF+crowded-first scheme. (a) $2R$, $2R$, and $2R$ are coming. (b) The first and third $2Rs$ are released. (c) $1R$ and $2R$ are coming. (d) The second 2R is released. (e) $1R$, $2R$, and $2R$ are coming.

4. Repeatedly perform the following procedure to allocate a free code of rate γR to $k-j-1$ NCTs of the ith LCC's bit-word

$$BW_i = (b_k, b_{k-1}, b_{k-2}, \cdots, b_l, \cdots, (b_j, b_j))_{k+2},$$

where $1 \le i \le 2^{n-\alpha-1}$, $\beta = \log_2 \gamma$, and $k = \log_2(R_{\max})$.
5. Assume that l ranges from $j+1$ to $k-1$, if $b_l = 0$, let $R = \frac{R_{\max}}{2^{k-l}}$ and R' be a pair of brother codes, and an NCT rooted at R' is formed. The similar NCT placement operation is applied to the NCT.

For example, as shown in Fig. 11a, $1R$ and $1R$ are respectively allocated to $RC_{16,3}$ and $RC_{16,4}$, Fig. 11b shows that $2R$ and $2R$ are allocated to $RC_{8,3}$ and $RC_{8,4}$, respectively. Finally, we give an example in Fig. 12 to illustrate a case in which an OVSF+crowded-first scheme [19] causes a code blocking, but our ROVSF scheme does not.

4.2 Code Replacement Scheme

To completely eliminate the code blocking problem, our replacement scheme on the ROVSF code tree was investigated with less code reassignment cost. The *code fragmentation* problem [3], [19] always occurs in a code tree after a number of variable data codes have been allocated and released. In Section 4.1, the allocation policy of a code placement scheme was already investigated to allocate the free code of rate γR, where γ is a power of two. However, when no such free code exists but the remaining capacity of the ROVSF code tree is sufficient, a code blocking problem occurs. To completely eliminate code blocking with less code reassignment cost, a replacement scheme was developed to relocate some codes in the ROVSF code tree to "squeeze" a free space for the new call. One other purpose of the replacement scheme is to overcome the drawback of the fixed lengths of LCCs; therefore, a dynamic adjustment operation for the LCC and NCT was designed.

To minimize the number of reassigned codes, Minn and Siu [14] 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 code replacement scheme aims to reduce the code reassignment cost for ROVSF code tree management. Our proposed code replacement scheme on the ROVSF code tree was modified from the DCA algorithm [14] as follows.

Our ROVSF-based DCA-modified algorithm is given by providing a different rule of the *minimum-cost branch* [14] for the ROVSF code tree. Consider that the remaining capacity of an ROVSF code tree is sufficient, assuming that there are $2^{n-\alpha-1}$ nodes, $RC_{2^{n-\alpha},i}$, where $1 \le i \le 2^{n-\alpha}$. Node $RC_{2^{n-\alpha},i}$, denoted a *branch node,* maintains a *branch cost*, where the *branch cost* is a count variable to estimate the number of assigned codes. The *branch cost* also denotes the number of assigned codes which need to be reassigned. The greater the number of the count variable is, the higher the reassignment cost will be. Each node, $RC_{2^{n-\alpha},i}$, forms a subtree, for which $RC_{2^{n-\alpha},i}$ is viewed as its root. Each node, $RC_{2^{n-\alpha},i}$, connects to an LCC or NCT, depending on whether the value of i is odd or even. For each $RC_{2^{n-\alpha},i}$, the branch cost, denoted $C_i = C(RC_{2^{n-\alpha},j})$, is determined by the total number of assigned nodes in the neighboring subtree or branch. Assuming that $RC_{2^{n-\alpha},i}$ and $RC_{2^{n-\alpha},j}$ are a pair of neighboring nodes, a minimum-cost branch is determined below. A branch node is said to be a minimum-cost branch if $\underset{1 \le i \le 2^{n-\alpha}}{Min}$

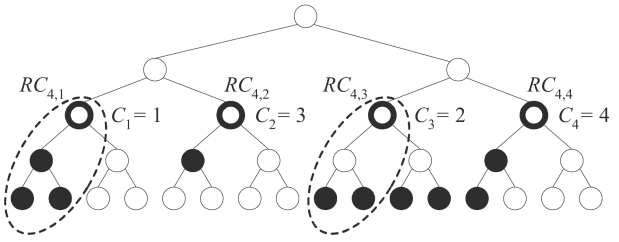

Fig. 13. Branch costs for our ROVSF-based DCA algorithm.

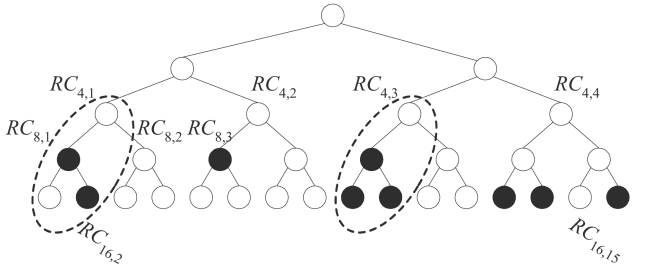

Fig. 14. Example of code replacement.

C_i. To reorganize the ROVSF code tree structure in order to obtain a code with higher data rate, all assigned codes in the neighboring subtree are released and reallocated to other subtrees. The reallocating operation occurs by sequentially performing our code placement algorithm for each of the assigned codes in the neighboring subtrees. For example, as shown in Fig. 13, the maximum transmission rate in the ROVSF code tree is $4R$. Therefore, C_1, C_2, C_3, and C_4 of $RC_{4,1}$, $RC_{4,2}$, $RC_{4,3}$, and $RC_{4,4}$ are 1, 3, 2, and 4, respectively; eventually, $RC_{4,1}$ is the minimum-cost branch.

To eliminate the code blocking problem in the ROVSF code tree with less code reassignment cost, the code replacement algorithm is described in the following steps:

1. Check to see if the current ROVSF code tree has sufficient remaining capacity to offer a new call with the data rate requirement γR. If it fails, reject the request.
2. Once the request is allowed, an ROVSF-based DCA-modified algorithm is performed; a *minimum-cost branch* is initially found which accommodates with a free code of data rate, γR.
3. If $\log_2(\gamma R) > \log_2(R_{\max})$, where $R_{\max}$ is the *chain-max-code* of the LCC, then go to **Step 4** to execute the dynamic adjustment operation. Otherwise, go to **Step 5**.
4. If there exists a $BW = (b_k, b_{k-1}, b_{k-2}, \cdots, (b_j, b_j))_{k+2}$, where $k = \log_2(R_{\max})$, the data rate of the new call is $2^{k+t}R$, and $1 \le t \le n-k$, then the LCC is enlarged to $(b_{k+t}, \cdots, b_k, b_{k-1}, b_{k-2}, \cdots, (b_j, b_j))_{k+t+2}$. Observe that, if all $b_{k+t}, b_{k+t-1}, \cdots,$ and b_{k+1} of all LCCs become zero, then the LCC is restored to $(b_k, b_{k-1}, b_{k-2}, \cdots, (b_j, b_j))_{k+2}$.
5. Sequentially execute the code placement algorithm to relocate all assigned codes in the neighboring subtrees for the *minimum-cost branch*.

For example, as shown in Fig. 14, the remaining capacity of the ROVSF code tree is $4R$. A new call is requested with a rate requirement of $4R$. This request is blocked by the code placement algorithm, although there is sufficient remaining capacity to accept this request. Therefore, code replacement is applied. The minimum-cost branch code is $RC_{4,1}$ to reallocate code $RC_{8,3}$. We attempt to move $RC_{8,3}$ to $RC_{8,2}$. However, $RC_{16,2}$ is in the subtree of $RC_{8,1}$, therefore, $RC_{8,3}$ cannot directly move to $RC_{8,2}$. We should first move $RC_{16,2}$ out. By repeatedly performing the ROVSF-based DCA-modified algorithm, $RC_{8,2}$ is again selected as the minimum-cost branch to move $RC_{16,2}$ to $RC_{16,15}$ so that $RC_{8,3}$ can successfully move to $RC_{8,2}$. Finally, the new call with $4R$ is successfully allocated to code $C_{4,1}$.

We further describe the dynamic adjustment operation as illustrated in Fig. 15. Four LCCs exist with a *chain-max-code* of $2R$, where $2R$ is the maximum transmission rate as illustrated in Fig. 15a. If a new call arrives with a rate requirement of $4R$, the remaining capacity is $5R$ and the request should be accepted; however, $4R >$ *chain-max-code* $= 2R$. The LCCs will be enlarged from the *chain-max-code* $= 2R$ to the *chain-max-code* $= 4R$. In such an example, $RC_{8,3}$ is reallocated to $RC_{8,6}$ based on the code placement strategy. The four LCCs with *chain-max-code* $= 2R$ are merged to form two LCCs with the *chain-max-code* $= 4R$. After the operation, we can directly support the new call with the rate requirement of $4R$ and the resultant LCCs as shown in Fig. 15b. In addition, the length of the LCCs can be restored to the smaller-sized LCCs. Fig. 15c shows the scenario in which both $RC_{8,1}$ and $RC_{16,14}$ depart from the system after $RC_{4,1}$ is allocated. If there are two new calls with data requirements of $1R$ and $2R$, these are assigned to $RC_{16,3}$ and $RC_{8,8}$, respectively. For a period of service time, t, $RC_{4,1}$ may release this data service after time t. To possibly maintain smaller-sized LCCs and maintain a larger number of LCCs, we may reduce the length of LCCs without the need of code reassignment operations. Resultant LCCs are shown in Fig. 15d.

5 Performance Analysis

We have developed a discrete event simulator to evaluate the performance achievement. To examine the effectiveness of our scheme, it is mainly compared with existing code placement and replacement schemes in the OVSF code tree [19]. To make a fair comparison, we provide ROVSF-version *leftmost-first*, *crowded-first*, and *mostuser-first* code placement strategies based on *leftmost-first*, *crowded-first*, and *mostuser-first* code placement strategies developed for the OVSF code tree [19]. To illustrate the performance achievement of our code placement and replacement schemes, we mainly analyzed the code blocking probability and the total number of code reassignments under different call patterns.

In the following simulator, we used OR, OL, OC, OM, RL, RC, and RM to denote different types of code trees and code placement strategies, where the first letter indicates the type of code tree (O = *OVSF-based* schemes and R = *ROVSF-based* schemes), and the second letter indicates the code placement strategy (R = *random*, L = *leftmost-first*, C = *crowded-first*, and M = *mostuser-first*). Each curve was obtained by using about 1,000 simulation runs, where each run contained at least 4,000 accepted calls. The system parameters in our simulator were as follows:

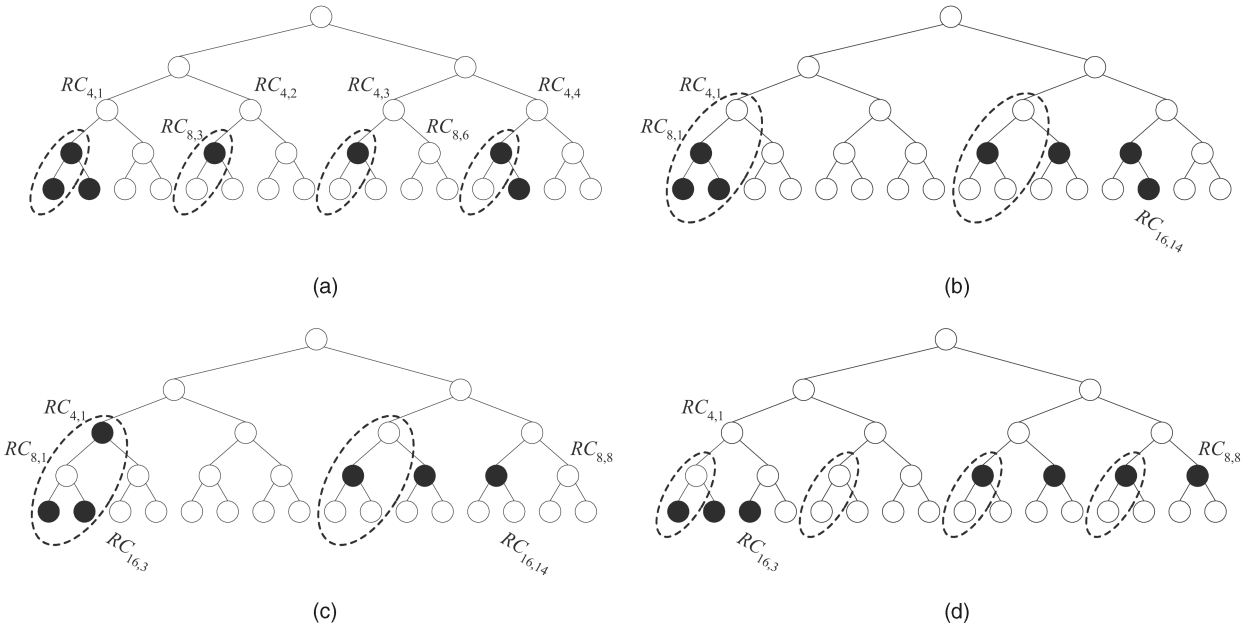

Fig. 15. Example of the dynamic adjustment operation.

- Capacity test: code-limited.
- Maximum spreading factors: 64 and 256 to reflect values following IS-95 and WCDMA standards [9], [16].
- Call arrival process: Poisson distribution with a mean arrival rate $\lambda = 1$ to 16 calls/unit time ($SF = 64$), and $\lambda = 4$ to 64 calls/unit time ($SF = 256$).
- Mean call duration time: Exponentially distributed with a mean value of four time units.
- Possible transmission rates: $1R$, $2R$, $4R$, and $8R$.

When a new request enters a WCDMA system, the system searches for an available code. The performance metrics observed in this study are defined below.

1. *Blocking probability*: The probability that a new call cannot be accepted, although the system still has sufficient remaining capacity.
2. *Utilization of LCCs*: The number of incoming requests assigned to LCCs divided by the total number of accepted requests.
3. *Number of reassigned codes*: The total number of code reassignments of all occupied codes for supporting a new call.

It is worth mentioning that efficient code placement and replacement schemes are achieved by having lower code blocking probability and less code reassignment cost. In the following, we illustrate our simulation results from various perspectives.

5.1 Impact of Code Placement

Different code placement strategies were adopted to observe the code blocking probability of a new call in the OVSF or ROVSF code tree even if the code tree had sufficient code capacity. The traffic pattern used was $1R : 2R : 4R : 8R = 1 : 1 : 1 : 1$ to indicate that the probabilities of incoming requests with rates of $1R$, $2R$, $4R$, and $8R$ were uniformly distributed. Fig. 16a shows the code blocking probability for eight schemes if only code placement was implemented, under a fixed mean arrival rate, if $SF = 256$. Observe that it is expected that crowded-first and mostuser-first schemes, which are meant to be used with an OVSF code tree, cannot perform well on an ROVSF code tree. Furthermore, we expected that the OR and OL schemes would have worse performances than the OM and OC schemes. There was no point of presenting the performances of the RC, RM, OR, and OL schemes. The results will be clearer if the RL scheme is compared only with the OC and OM schemes. Therefore, Fig. 17a is provided to illustrate the code blocking probability when $SF = 64$.

The ROVSF-based scheme with the leftmost strategy (RL) has the lowest code blocking probability. This is because code blocking always occurs for new calls with high transmission rates. Recalling that the LCC has the unsequence allocation property as described in Section 3.3, therefore, the lower code blocking probability is obtained because our scheme tries to allocate most code requests to LLCs. Using RL, our scheme can allocate more codes into LCCs. Only a few code requests, which are allocated outside LCCs, may experience code blocking problems. For example, when the mean arrival rate is equal to 64 with $SF = 256$, the code blocking probabilities are 15 percent, 18 percent, 18 percent, 22 percent, 22 percent, 23 percent, and 27 percent for the RL, OC, OM, RC, RM, and OR schemes, respectively.

To understand the effect of utilization of LCCs on the code blocking probability, the utilization of LCCs for RL, RC, and RM are given in Fig. 16b and Fig. 17b for $SF = 256$ and 64. Basically, the higher the utilization of LCCs is, the lower the code blocking probability that is obtained. For instance, the utilizations of LCC of RL are 1 and 0.85, and the code blocking probabilities are 0 and 0.15, respectively. Similar results for the RC and RM schemes were obtained in Fig. 16b and Fig. 17b. Observe that our scheme attempts to

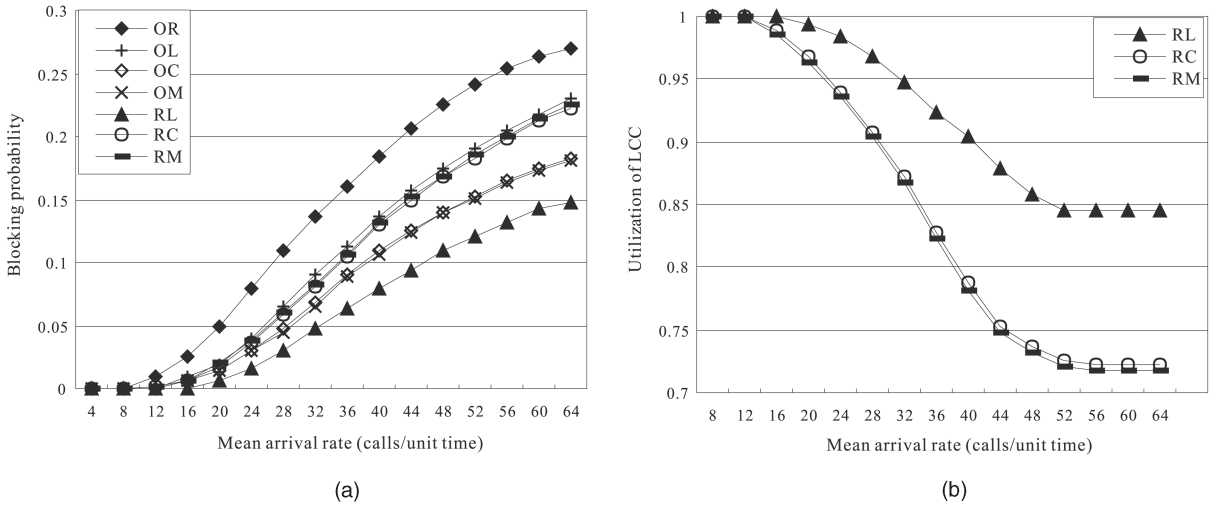

Fig. 16. $SF = 256$, $\alpha = 5$: (a) blocking probability at different traffic loads and (b) utilization of linear-code chains.

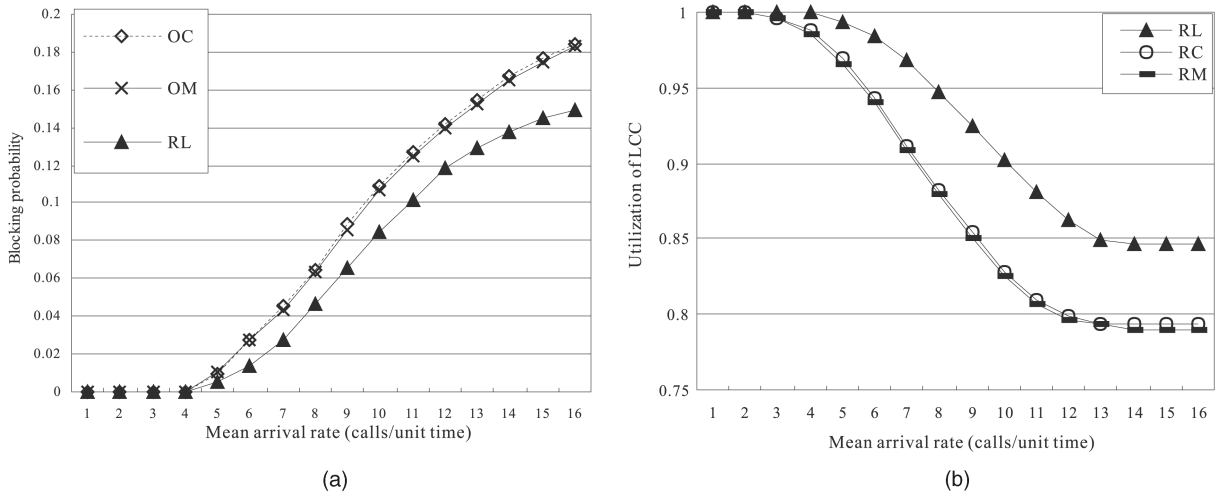

Fig. 17. $SF = 64$, $\alpha = 5$: (a) blocking probability at different traffic loads and (b) utilization of linear-code chains.

properly assign most codes to LCCs, thus lower code blocking probabilities than other schemes were obtained.

5.2 Impact of Code Replacement

The code replacement scheme is designed to completely eliminate the code blocking problem since there is sufficient remaining capacity. To see the efficiency of our code replacement scheme, observe the total number of code reassignments. In this simulation, we first considered the traffic pattern to be $1R : R : 4R : 8R = 1 : 1 : 1 : 1$. This indicates that we first considered equal probabilities for all incoming requests with data rates of $1R$, $2R$, $4R$, and $8R$. Various traffic patterns are investigated later. Considering the code replacement, therefore there is no code blocking problem. Therefore, we only observed the number of code reassignments. To compare with eight schemes, Fig. 18a shows the results of the number of code reassignments under a fixed mean arrival rate, λ, and $SF = 256$. Only comparing the RL, OC, and OM schemes, Fig. 18b shows the results of the number of code reassignments, when $SF = 64$.

This result shows that the RL scheme still has the lowest reassignment cost when compared to all other schemes. This is possibly because the RL scheme better utilizes LCCs, thus, it can reduce the total number of code reassignments to accommodate new calls with high data rates. For example, as shown in Fig. 18a for $SF = 256$, if the mean arrival rate is 64, there are 530, 660, 669, 784, 793, 810, and 1160 code reassignments for the RL, OM, OC, RC, RM, OL, and OR schemes, respectively. Our ROVSF-based scheme has an improved number of code reassignments. Similar results can be found in Fig. 18b for $SF = 64$.

5.3 Impact of the Length of LCCs

To observe the influence on the code blocking probability and code reassignment cost, different lengths of LCCs were considered. Fig. 19 shows the simulation results of different lengths of LCCs, which reflect the effect of the code blocking probability on the mean arrival rate, λ. The curves of RL2, RL3, RL4, RL5, and RL6 denote that the default lengths of the LCCs are 2, 3, 4, 5, and 6, respectively. Note that the traffic pattern used here is $1R : 2R : 4R : 8R = 1 : 1 : 1 : 1$. Hence, the code blocking probability of RL2 is stable at

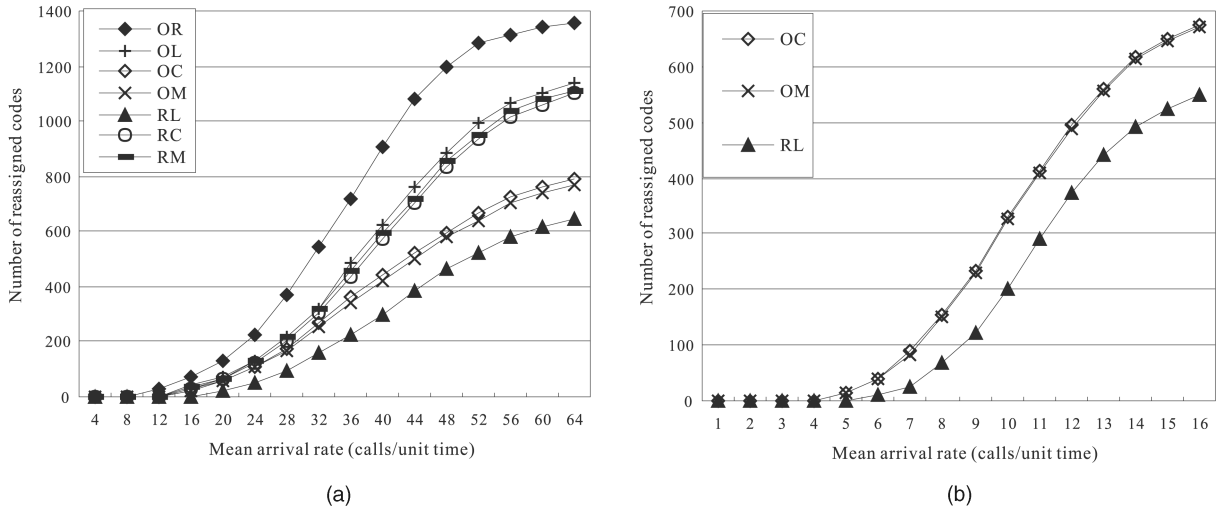

Fig. 18. Number of code reassignments to remove code-blocking: (a) $SF = 256$, $\alpha = 5$ and (b) $SF = 64$, $\alpha = 5$.

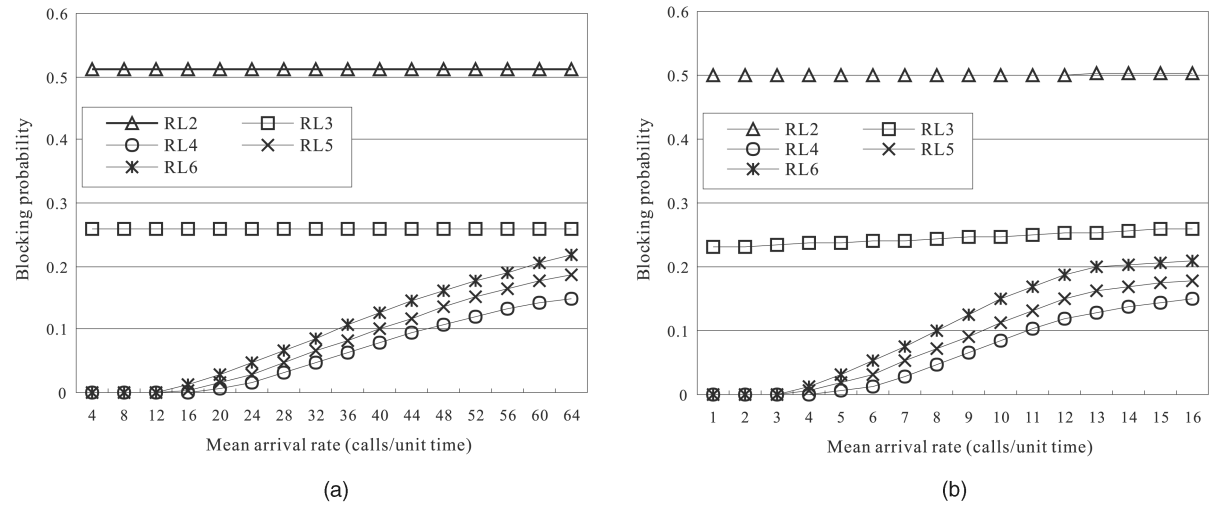

Fig. 19. Impacts of the length of the linear-code chain on the blocking probability: (a) $SF = 256$ and (b) $SF = 64$.

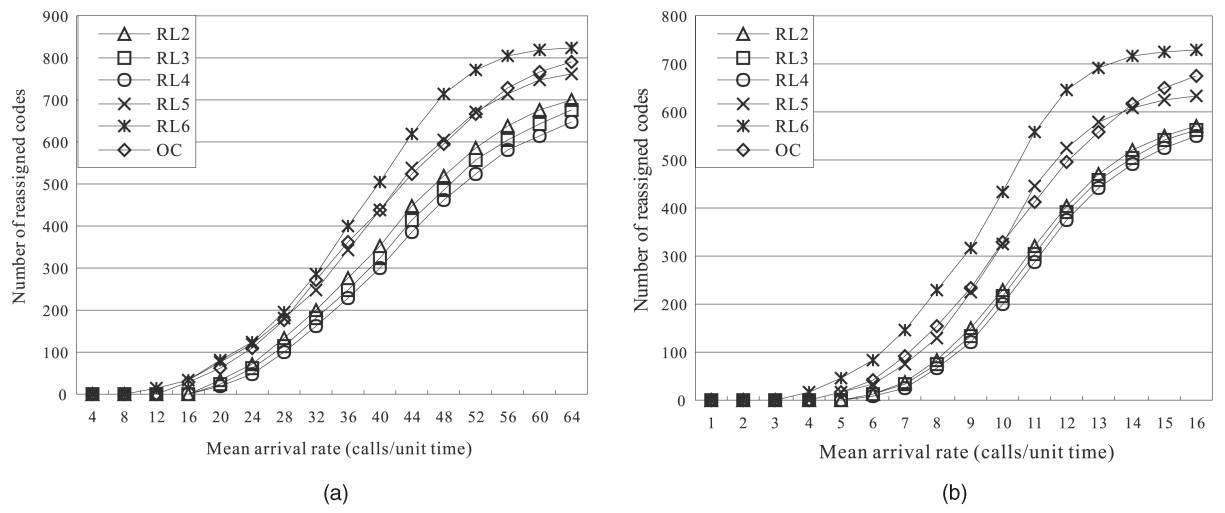

Fig. 20. Impacts of the length of the linear-code chain on the reassignment cost: (a) $SF = 256$ and (b) $SF = 64$.

about 51 percent, because most data requests with $8R$ and $4R$ are blocked. Similarly, most requests with a data rate of $8R$ are blocked by RL3 and the code blocking probability is stable at about 27 percent. In general, the lengths of RL5 and RL6 are too large, so that total numbers of assigned codes in the LCCs are fewer than for RL4; therefore, the code blocking probabilities of RL5 and RL6 are higher than that of RL4. Fig. 20 shows the simulation results of the

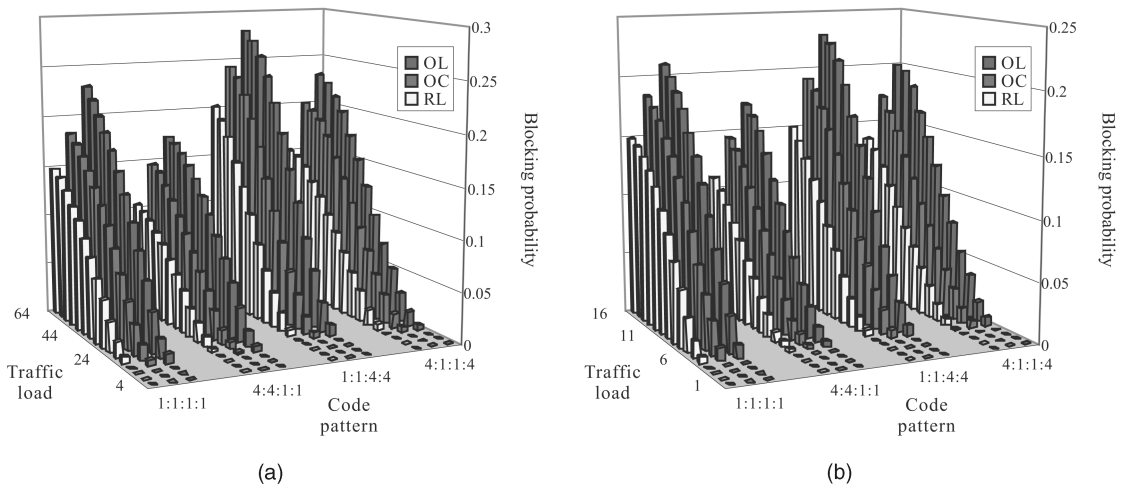

Fig. 21. Effects of different call patterns on code blocking probabilities: (a) $SF = 256$, $\alpha = 5$ and (b) $SF = 64$, $\alpha = 5$.

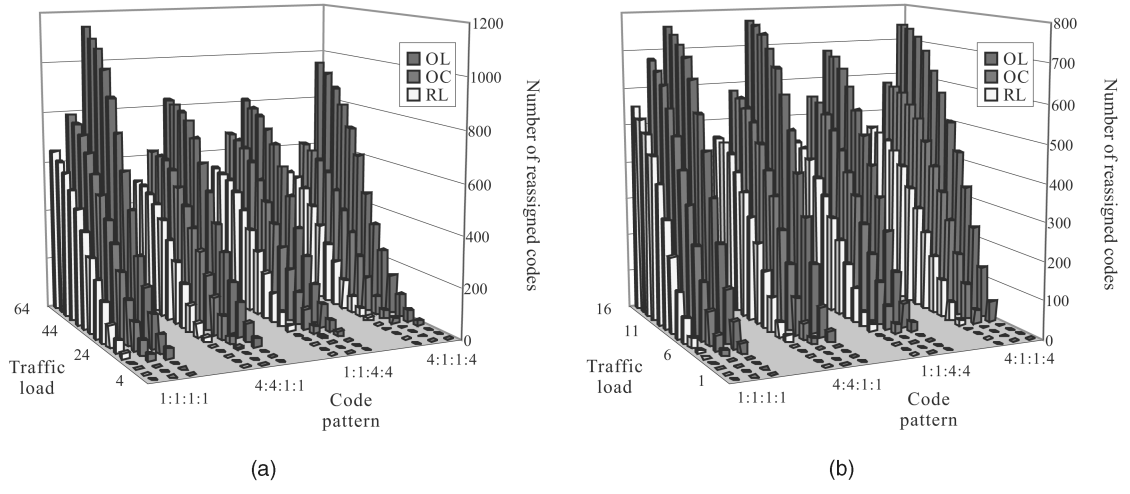

Fig. 22. Effect of different call patterns on code reassignment costs: (a) $SF = 256$, $\alpha = 5$ and (b) $SF = 64$, $\alpha = 5$.

effect of the number of code reassignments on various default lengths of LCCs. The dynamic adjustment operation of the linear-code chain was performed for our code replacement scheme. Observe that data requests with data rates of $4R$ and $8R$ can finally be accepted by RL2 and RL3 even if $\log_2(kR) > \log_2(R_{\max})$. In our simulation, RL4 had the best performance since its LCC length mostly met our traffic pattern. On the other hand, the lengths of LCCs of R5 and R6 were too large, so they incurred greater numbers of code reassignments compared to R4, R3, or R2. This implies that the best performance is obtained if the default length of the LCC accommodates the call pattern.

5.4 Impact of Call Patterns

To illustrate the effects of call patterns, various traffic patterns of $1R : 2R : 4R : 8R = 4 : 4 : 1 : 1$, $1 : 1 : 4 : 4$, and $4 : 1 : 1 : 4$ were considered in order to observe the influences on the code blocking probability and the number of code reassignments. Fig. 21 and Fig. 22 illustrate the simulation results of the code blocking probability and the number of code reassignments. Better performance was obtained for the call patterns of $1R : 2R : 4R : 8R = 1 : 1 : 4 : 4$ and $4 : 1 : 1 : 4$. These results are expected because our ROVSF-based scheme can reduce the code blocking for high-rate transmission requests. Our scheme does not show obvious improvements if the call pattern is $4 : 4 : 1 : 1$ because data requests with $1R$ are always accepted if the ROVSF code tree is not full. Our RL scheme is always better than the OC scheme since LLCs possess the unsequence property. We have discussed the utilization of LCCs versus the mean arrival rate in Fig. 16b and Fig. 17b to illustrate that most incoming requests will be assigned codes from LCCs. The more the LCC is utilized, the lower the code blocking probability will be. Otherwise, if an incoming code cannot be assigned to LCCs, we can still assign this code to NCTs. Consequently, our RL scheme is always better than the OC scheme since LLCs possess the unsequence property.

6 Conclusions

This work mainly investigates the code placement and replacement schemes on the ROVSF (rotated-orthogonal variable spreading factor) code tree. The main idea of our schemes is to identify linear-code chains (LCCs) and nonlinear-code trees (NCTs) in the ROVSF code tree. LCCs

possess the *unsequence property*, which provides a new code placement and replacement mechanism. Our scheme initially attempts to allocate request codes to LCCs and, then, tries to allocate them to NCTs. Using linear-code chains with the unsequence property allows us to reduce the code blocking probability and code reassignment cost. Finally, the simulation results illustrate that our code placement and replacement results based on the ROVSF code tree can actually improve the code blocking probability and reduce the code reassignment cost. Future work will investigate multicode placement and replacement schemes to eliminate the internal fragmentation problem in the ROVSF code tree.

ACKNOWLEDGMENTS

We would like to thank the editors and anonymous referees whose insightful comments helped us improve the presentation of the paper. This work was supported in part by the National Science Council of the Republic of China, under grants NSC-92-2213-E-194-022 and NSC-93-2213-E-194-028.

REFERENCES

[1] F. Adachi, M. Sawahashi, and H. Suda, "Wideband DS-CDMA for Next-Generation Mobile Communication Systems," *IEEE Comm. Magazine,* vol. 36, no. 9, pp. 56-69, Sept. 1998.

[2] A. Baier, U.C. Fiebig, W. Granzow, W. Koch, P. Teder, and J. Thielecke, "Design Study for a CDMA-Based Third-Generation Mobile Radio System," *IEEE J. Selected Areas in Comm.,* vol. 12, no. 4, pp. 733-743, May 1994.

[3] C.M. Chao, Y.C. Tseng, and L.C. Wang, "Reducing Internal and External Fragmentations of OVSF Codes in WCDMA Systems with Multiple Codes," *Proc. IEEE Wireless Comm. and Networking Conf.,* vol. 1, pp. 693-698, 2003.

[4] J.C. Chen and W.S. Chen, "Implementation of an Efficient Channelization Code Assignment Algorithm in 3G WCDMA," *Proc. Nat'l Computer Symp.,* pp. E237-E244, 2001.

[5] W.T. Chen, Y.P. Wu, and H.C. Hsiao, "A Novel Code Assignment Scheme for W-CDMA Systems," *Proc. IEEE Vehicular Technology Conf.,* vol. 2, pp. 1182-1186, Fall 2001.

[6] Y.-S. Chen, T.-C. Kao, and J.-P. Sheu, "A Mobile Learning System for Scaffolding Bird Watching Learning," *J. Computer Assisted Learning,* vol. 19, no. 3, pp. 347-359, Sept. 2003.

[7] Y.-S. Chen, T.-C. Kao, G.-J. Yu, and J.-P. Sheu, "A Mobile Butterfly-Watching Learning System for Supporting Independent Learning," *Proc. IEEE Int'l Workshop Wireless and Mobile Technologies in Education (IEEE WMTE 2004),* pp. 11-18, Mar. 2004.

[8] R.G. Cheng and P. Lin, "OVSF Code Channel Assignment for IMT-2000," *Proc. IEEE Vehicular Technology Conf.,* vol. 3, pp. 2188-2192, Spring 2000.

[9] E. Dahlman, B. Gudmundson, M. Nilsson, and J. Skold, "UMTS/IMT-2000 Based on Wideband CDMA," *IEEE Comm. Magazine,* vol. 36, pp. 70-80, Sept. 1998.

[10] M. Dell'Amico, M.L. Merani, and F. Maffioli, "Efficient Algorithms for the Assignment of OVSF Codes in Wideband CDMA," *Proc. IEEE Int'l Conf. Comm. (ICC'02),* vol. 5, pp. 3055-3060, Apr. 2002.

[11] R. Fantacci and S. Nannicini, "Multiple Access Protocol for Integration of Variable Bit Rate Multimedia Traffic in UMTS/IMT-2000 Based on Wideband CDMA," *IEEE J. Selected Areas in Comm.,* vol. 18, no. 8, pp. 1441-1454, Aug. 2000.

[12] V.K. Garg, *IS-95 CDMA and cdma2000: Cellular/PCS Systems Implementation.* New Jersey: Prentice Hall, Dec. 1999

[13] H. Holma and A. Toskala, *WCDMA for UMTS: Radio Access for Third Generation Mobile Comm.,* second ed. New York: John Wiley & Sons, Sept. 2002.

[14] T. Minn and K.Y. Siu, "Dynamic Assignment of Orthogonal Variable-Spreading-Factor Codes in W-CDMA," *IEEE J. Selected Areas in Comm.,* vol. 18, no. 8, pp. 1429-1440, Aug. 2000.

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

[16] T.S. Rappaport, *Wireless Communications: Principles and Practice,* second ed. Prentice Hall, Dec. 2001.

[17] A.N. Rouskas and D.N. Skoutas, "OVSF Codes Assignment and Reassignment at the Forward Link of W-CDMA 3G Systems," *Proc. 13th IEEE Int'l Symp. Personal, Indoor, and Mobile Radio Comm. (PIMRC'02),* vol. 5, pp. 2404-2408, Sept. 2002.

[18] L.F. Tsaur and D.C. Lee, "Symbol Rate Adaptation and Blind Rate Detection Using FOSSIL (Forest for OVSF-Sequence-Set-Inducing Lineages)," *Proc. IEEE Int'l Conf. Comm.,* vol. 6, pp. 1754-1759, 2001.

[19] 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.

Yuh-Shyan Chen received the BS degree in computer science from Tamkang University, Taiwan, Republic of China, in June 1988 and the MS and PhD degrees in computer science and information engineering from the National Central University, Taiwan, Republic of China, in June 1991 and January 1996, respectively. He joined the Department of Computer Science and Information Engineering at Chung-Hua University, Taiwan, Republic of China, as an associate professor in February 1996. He joined the Department of Statistic, National Taipei University, in August 2000, and then joined the Department of Computer Science and Information Engineering, National Chung Cheng University, in August 2002. Dr. Chen served as editor-in-chief of the *International Journal of Ad Hoc and Ubiquitous Computing* (IJAHUC), editorial board member of *Telecommunication Systems*, the *International Journal of Internet Protocol Technology* (IJIPT), and *The Journal of Information, Technology and Society* (JITAS), guest editor of *Telecommunication Systems*, special issue on wireless sensor networks (2004), and guest editor of the *Journal of Internet Technology*, special issue on wireless Internet applications and systems (2002) and special issue on wireless ad hoc network and sensor networks (2004). He was a workshop cochair of the 2001 Mobile Computing Workshop, a member of the IASTED Technical Committee on Telecommunications for 2002 and 2005, program committee member of IEEE ICPP '2003, IEEE ICDCS '2004, IEEE ICPADS '2001, IEEE ICCCN '2001 and 2004, IASTED CCN '2002 and 2004, IASTED CSA '2004, IASTED NCS '2005, and MSEAT '2003 and 2004. His paper won the 2001 IEEE 15th ICOIN-15 Best Paper Award. His recent research topics include mobile ad hoc networks, wireless sensor networks, Bluetooth WPANs, mobile computing, mobile learning systems, and mobile P2P communication. Dr. Chen is a member of the IEEE, the IEICE Society, and the Phi Tau Phi Society.

Ting-Lung Lin received the BS degree in information management from the ChaoYang University of Technology, Taiwan, Republic of China, in June 2001 and the MS degree in information management from National Taipei University, Taiwan, Republic of China, in July 2003. His research interests include wireless network and WCDMA systems.