OVSF Code Channel Assignment With Dynamic Code Set and Buffering Adjustment for UMTS

Phone Lin, *Member, IEEE*, Chai-Hien Gan, and Ching-Chi Hsu

***Abstract*—The Universal Mobile Telecommunications System (UMTS) provides users variable data rate services, which adopts wide-band code-division multiple-access (WCDMA) as the radio access technology. In WCDMA, orthogonal variable spreading factor (OVSF) codes are assigned to different users to preserve the orthogonality between users' physical channels. The data rate supported by an OVSF code depends on its spreading factor (SF). An OVSF code with smaller SF can support higher data rate services than that with larger SFs. Randomly assigning the OVSF code with a large SF to a user may preclude a larger number of OVSF codes with small SFs, which may cause lots of high data rate call requests to be blocked. Therefore the OVSF code assignment affects the performance of the UMTS network significantly. In this paper, we propose two OVSF code assignment schemes CADPB1 and CADPB2 for UMTS. Both schemes are simple and with low system overhead. The simulation experiments are conducted to evaluate the performances for our schemes. Our study indicates that our proposed schemes outperform previously proposed schemes in terms of the weighted blocking probability and fairness index. Our schemes improve the call acceptance rate by slightly introducing call waiting time.**

***Index Terms*—Code channel assignment, dynamic adjustment, orthogonal variable spreading factor (OVSF), Universal Mobile Telecommunications System (UMTS), wide-band code-division multiple-access (WCDMA).**

I. Introduction

THE *Universal Mobile Telecommunications System* (UMTS) [4], [9] provides variable data rate services to users. Fig. 1 illustrates the network architecture of UMTS, which consists of the *Terrestrial Radio Access Network* (UTRAN) and the core network. A *user equipment* (UE) communicates with UTRAN through the air interface Uu [1] where *wide-band code-division multiple-access* (WCDMA) is used as the radio access technology. In WCDMA, *orthogonal variable spreading factor* (OVSF) codes are assigned to users' calls to preserve the orthogonality between users' physical channels. The data rate supported by an OVSF code depends on its *spreading factor* (SF). An OVSF code with smaller SF can support higher data rate services than one with larger SF. The OVSF codes are generated by using a complete binary tree. Depending on the height of the position in the tree where an OVSF code locates, the SF of the OVSF code is determined. An OVSF code can be assigned to a user's call if and only if the code, all of its ancestor OVSF codes, and descendant OVSF codes in the tree have not been assigned to other users' calls. Within the coverage area of a Node B, the total data rate that can be assigned to users is fixed. In this paper, the coverage area of a Node B is denoted as a "cell." The details of the criteria for the OVSF code assignment are given in Section II. Based on the OVSF code assignment criteria, randomly assigning the OVSF code with a large SF to a user may preclude a larger number of OVSF codes with small SFs, which may cause many high data rate call requests to be blocked. Thus the OVSF code assignment affects the quality of service (QoS) for the UMTS network significantly. For the above issue, 3GPP 25.213 [2] and 25.223 [3] proposed the multicode and single-code transmission mechanisms for variable data rate services.

In the multicode mechanism, multi-OVSF codes are used to serve a user's call, and in the single-code mechanism, single OVSF code is used to serve a user's call. Reference [13] and our previous study [6] proposed the OVSF code assignment schemes for the multicode mechanism. The multicode mechanism requires the complex transceiver, which may increase the complexity of a UE. To reduce the complexity of a UE, the single-code mechanism is preferred for variable data rate services. Several OVSF code assignment schemes [5], [7], [8], [12], [14] are proposed for the single-code mechanism, which can be classified into two categories: "rearrangeable" and "nonrearrangeable." The rearrangeable scheme may reassign some calls (being severed with low data rate) other OVSF codes to hold a satisfied OVSF code for the incoming call request. Reference [12] has shown that the rearrangeable scheme can reduce the call blocking probability. However, in the rearrangeable scheme, there are drawbacks for rearrangement operations: Extra computation is required to "find" the served calls to be reassigned OVSF codes, which increases the computation overhead of the system. Furthermore, when rearrangement operations are exercised, some of the served calls have to be "reassigned" OVSF codes, which introduces extra message exchange, may cause a served call to be forced to terminate, and thus decreases the QoS of the network. Therefore, the most considered design issue in the rearrangeable schemes is to reduce the number of reassignment of the codes serving calls when accepting a new request. The nonrearrangeable schemes [5], [14] are considered simple and with low system overhead for the OVSF code assignment.

Manuscript received October 26, 2002; revised September 14, 2004. The work of P. Lin was supported in part by the National Science Council, Taiwan, R.O.C., under Contract NSC-93-2213-E-002-095; in part by the Institute for Information Industry, Taiwan, R.O.C.; in part by the Chung-Shan Institute of Science and Technology, Taiwan, R.O.C.; in part by CCL/ITRI, Taiwan, R.O.C.; and in part by Microsoft. The review of this paper was coordinated by Prof. M. Juntti.

P. Lin and C.-H. Gan are with the Department of Computer Science and Information Engineering, National Taiwan University, Taipei 106, Taiwan, R.O.C. (e-mail: plin@csie.ntu.edu.tw; chyen@pcs.csie.ntu.edu.tw).

C.-C. Hsu is with the Institute for Information Industry, Taiwan, R.O.C. (e-mail: cchsu@iii.org.tw).

Digital Object Identifier 10.1109/TVT.2004.842449

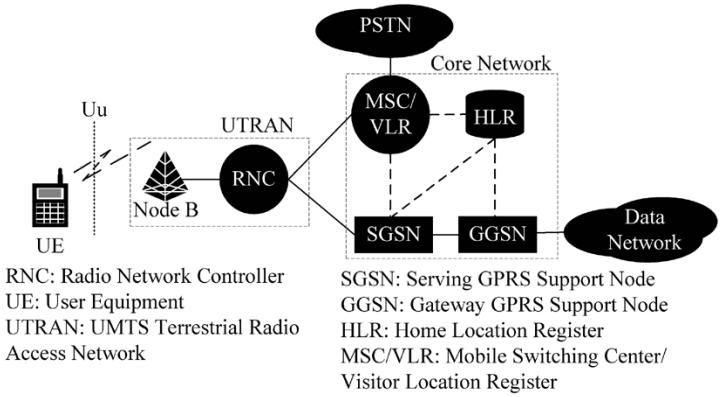

Fig. 1. The UMTS network architecture.

In this paper, we propose two nonrearrangeable OVSF code assignment schemes: code assignment with dynamic partition and buffering 1 (CADPB1) and CADPB2 with buffering mechanisms. Based on the measured traffic load, our schemes dynamically partition the OVSF code tree into subgroups to serve requests with different data rates. The buffering mechanism is adopted to queue the call requests that cannot be served (due to no satisfied OVSF code) instead of blocking them immediately. The buffering mechanism is considered simple and cost-effective to reduce the call blocking probability, which can have the same effect of the reassignment operation (i.e., reducing the call blocking probability). Our schemes also dynamically adjust the sizes of the buffers according to the call waiting times. In [7] and [8], authors also adopted the tree partition concept for OVSF code assignment, where the tree partitioning (TP) scheme is proposed. The similitude between TP and our schemes is that the OVSF codes are partitioned into groups to serve different types of call requests, and the differences are listed as follows.

1) TP is a rearrangeable scheme, but our schemes are nonrearrangeable schemes.
2) In our schemes, the dimension of each code tree partition is "dynamically" updated based on the traffic loads of different types of calls.
3) One more difference between the TP scheme and the CADPB2 scheme is that in CADPB2, the code sets to support high data rate call requests also contain that to support the lower data rate call requests so that ancestors of the codes in the code set (supporting lower data rate calls) can also be used to support high data rate requests.

In the end of this paper, we compare the performance of our schemes and the rearrangeable schemes to see the effects of the buffering mechanism and reassignment operation. This comparison was not seen in the other previous studies. This study focuses on the media access control for different types of calls. The consideration of other physical layer limitation factors (e.g., signal-to-noise ratio, bit error rate, total transmission power, etc.) is not treated in this paper. This is because when the network enters the phase "allocating an OVSF code to a call request," the other physical layer limitations for the request should have been satisfied [9]. For the details of other physical layer limitation factors, readers may refer to [9].

This paper is organized as follows. In Section II, we introduce the criteria for the OVSF code assignment. Section III presents the two schemes. In Section IV, we evaluate the performances for the CADPB1 and CADPB2 schemes. The conclusion is given in Section V.

II. Criteria for OVSF Code Assignment

The rules for the generation of OVSF codes are based on a complete binary tree with $K+1$ layers (from Layer 0 to Layer K) where $K \geq 2$. Fig. 2 depicts the binary tree. An OVSF code is corresponding to a node in the tree, which is denoted as $C_{k,n}$, where k is the layer number $(0 \leq k \leq K)$ and n is the position number in Layer $k (1 \leq n \leq 2^k)$. The node positions in Layer k are sequentially labeled (i.e., $1, 2, \ldots, 2^k$) from the left to the right. An OVSF code in Layer k has the spreading factor SF$= 2^k$. The data rate that can be supported by an OVSF code is determined by its SF. An OVSF code with the smaller SF can support higher data rate services than that with the larger SF. Suppose that the OVSF code $C_{K,n}$ can support data rate r bits/s. Then the data rate $R(C_{k,n})$ supported by $C_{k,n}$ is

$$R(C_{k,n}) = 2^{K-k} r \text{ bits/s}. \tag{1}$$

Three statuses free, busy, and assignable are used to indicate if an OVSF code can be allocated to a user. An OVSF code is free if the code has not been assigned to a user. An OVSF code is busy if the code is assigned to a user. A code is assignable if and only if the code and all of its ancestor and descendant OVSF codes in the tree are free. Only the OVSF codes in the assignable status can be assigned to users' calls, which preserve the orthogonality between users' physical channels. Accordingly, the remaining data rate that can be assigned to users within a cell is depending on the statuses of the all OVSF codes in the tree, which can be calculated by summing the data rates supported by the assignable OVSF codes in Layer K. Let R_A denote the remaining data rates within a cell. Then

$$R_A = \sum_{n=1}^{2^K} I_A(C_{K,n}) R(C_{K,n}) \tag{2}$$

where $R(C_{K,n})$ is obtained from (1) and

$$I_A(C_{K,n}) = \begin{cases} 1, & \text{if } C_{K,n} \text{ is assignable} \\ 0, & \text{otherwise.} \end{cases}$$

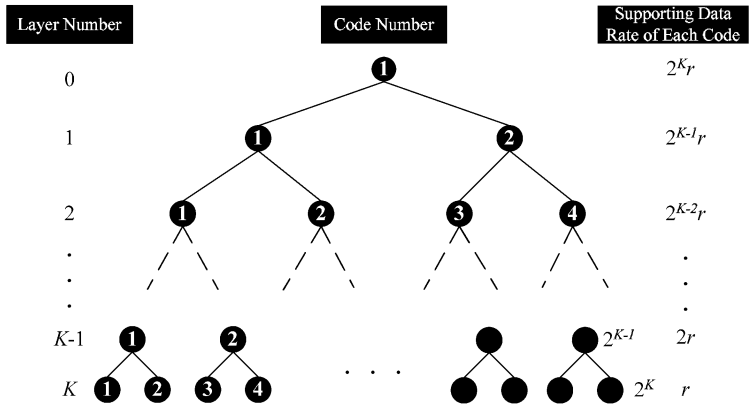

Fig. 2. A $K+1$-layer OVSF code tree.

The total data rate R_T supported by a Node-B can be calculated by summing the the total data rates supported by all codes in Layer K. From (2), we have

$$R_T = \sum_{n=1}^{2^K} R(C_{K,n}) = 2^K \, r \text{bits/s}.$$

III. The Code Assignment Schemes With Dynamic Partition and Buffering Mechanisms

This section proposes two OVSF code assignment schemes CADPB1 and CADPB2 for different types of calls. The call requests are classified into m types: type$_1$, type$_2$, …, type$_m$. Suppose that for $1 \le i \le m$, the data rate requested by the type$_i$ calls is $a_i r$ bits/s, where a_i is power of two. Without loss of generality, we assume that $a_1 < a_2 < \cdots < a_m$.

A. The CADPB1 Scheme

In CADPB1, we partition the OVSF codes in the code tree into m sets: $S_1, S_2, \ldots, S_m$. The OVSF codes in S_i are used to serve the type$_i$ calls, where $i \in \{1, 2, \ldots, m\}$. Let the number of codes in S_i be N_i. For the type$_i$ call requests, we maintain the first-in first-out buffer B_i to queue the type$_i$ call requests that cannot be served due to no assignable code in S_i. Let the size of B_i be Q_i.

In CADPB1, we dynamically partition the code tree and adjust the buffer sizes according to the measured traffic loads and the average call waiting times, respectively. The following system parameters are used in CADPB1 for the dynamic adjustment functionality.

1) Δ is a threshold value used to trigger the dynamic adjustment functionality for both the code sets and buffer sizes. After every Δ calls (including m types) have been served, the network dynamically partitions the code sets and adjusts the buffer sizes.
2) $CR_{A,i}$ and $CR_{D,i}$ are counters that count the numbers of the type$_i$ call arrivals and of the served type$_i$ calls during the time period when Δ calls (including m types) have been served, respectively.
3) $T_{W,i}, T_{A,i}$, and $T_{S,i}$ accumulate the time spent by the type$_i$ calls waiting in B_i, the interarrival time of the type$_i$ calls, and the service time spent by the type$_i$ calls when Δ calls (including m types) have been served, respectively.
4) W_i, A_i, D_i, and L_i store the average call waiting time, average arrival rate, average service time, and average traffic load for the type$_i$ calls during the time period when Δ calls (including m types) have been served, respectively.
5) $\delta^-(\delta^+)$ is a threshold values used to trigger the buffer size Q_i to be discounted (increased) by one when the average waiting time of the type$_i$ calls is larger (smaller) than $\delta^-/D_i(\delta^+/D_i)$.

The CADPB1 scheme consists of four procedures: Initial, Arrival, Departure, and Adjustment. "Initial" initiates the system parameters when the system starts. "Arrival" allocates the OVSF code to a call request. "Departure" is invoked when a call completes. "Adjustment" repartitions the code sets and readjusts the buffer sizes for each type of calls, which is triggered when every Δ calls have been served (i.e., $\sum_{1 \le i \le m} CR_{D,i} = \Delta$). The details of the four procedures are described as follows.

1) *Initial:* When the system starts up, for each type of call, N_i and Q_i are set to

$$N_i \leftarrow \frac{R_T}{m a_i r} \quad \text{and} \quad Q_i \leftarrow \theta N_i \tag{3}$$

where θ is a constant number. Initially, we allocate the same data rate to each code set. The buffer size Q_i is initially set to θN_i. Based on N_i, for $i \in \{1, 2, \ldots, m\}$, we generate the code set S_i by the following rule:

$$S_i = \left\{ C_{K-\log_2 a_i, l} \,\middle|\, \frac{R_T - \sum_{i<j\le m} a_j r N_j}{a_i r} - N_i + 1 \le l \le \frac{R_T - \sum_{i<j\le m} a_j r N_j}{a_i r} \right\}. \tag{4}$$

Fig. 3 shows an example of a partition result for four code sets where $R_T = 32r$ bits/s. In the figure, the nodes in the four rectangles are the codes in S_i, where $i \in \{1, 2, 3, 4\}$. For $i \in \{1, 2, \ldots, m\}, CR_{A,i}, CR_{D,i}, T_{A,i}, T_{S,i}$, and $T_{W,i}$ are set to zero, and all the codes in S_i are marked as assignable.

2) *Arrival:* For a type$_i$ call request where $i \in \{1, 2, \ldots, m\}$, the network finds an assignable code $C_{K-\log_2 a_i, j}$ in S_i. If such a code exists, the code is allocated to the request

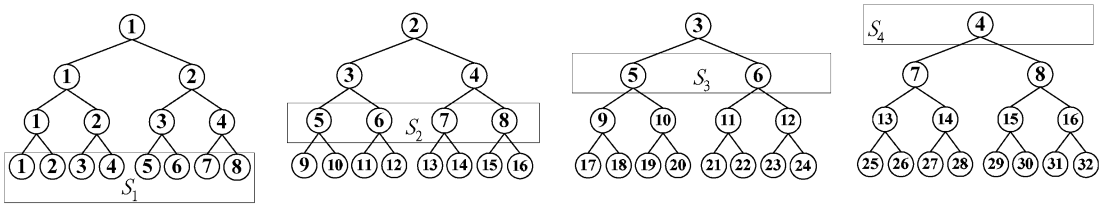

Fig. 3. An example for the partition result of Initial procedure in CADPB1.

and marked as busy. Otherwise (i.e., there is no *assignable* code in S_i), the network checks B_i. If B_i is full, the call request is blocked. Otherwise (i.e., B_i is not full), the call request is queued into B_i, $CR_{A,i}$ is increased by one, and the time period between the call and last type$_i$ call (i.e., intercall arrival time) is accumulated into $T_{A,i}$. If tie-breaking occurs (i.e., there are call requests arriving at the same system time), they can be processed in any order.

3) *Departure:* When a type$_i$ call (served with $C_{K-\log_2 a_i,j}$) completes, the code is marked as assignable, $CR_{D,i}$ is increased by one, the service time of this type$_i$ call is accumulated into $T_{S,i}$, and the waiting time of this type$_i$ call in B_i is accumulated into $T_{W,i}$. Then B_i is checked. If there are requests queued in B_i, the released code is allocated to the queued request.

   Note that if $C_{K-\log_2 a_i,j} \in S_i$ is allocated before the Adjustment procedure exercises, and released after the Adjustment procedure exercises, the code may not belong to S_i. If so, the code is marked as free. After this action, some codes in other code sets may become assignable by checking their ancestor and descendant codes. Suppose that the code that becomes assignable is in S_k. Then B_k is checked. If B_k is not empty, then the *assignable* code is allocated to the requests buffered in B_k.

4) *Adjustment:* When every Δ calls have been served (i.e., $\sum_{1\le i\le m} CR_{D,i} = \Delta$), the code tree is repartitioned, and the buffer sizes are readjusted as follows.

   To repartition the code tree, we first compute the traffic load of each type of calls as follows:

$$L_i = a_i A_i D_i \tag{5}$$

where

$$A_i = \frac{CR_{A,i}}{T_{A,i}} \quad \text{and} \quad D_i = \frac{T_{D,i}}{CR_{D,i}}. \tag{6}$$

Based on (5), for $i = m, m-1, \ldots, 2, 1$, we recursively determine the size of each code set (i.e., N_i) by the following equation:

$$N_i = \text{round}\left(\frac{L_i}{\sum_{1<j\le i} L_j} \times \frac{R_T - \sum_{i<j\le m} a_j r N_j}{a_i r}\right). \tag{7}$$

By applying (7) into (4), we obtain the new code set S_i for the type$_i$ calls.

The size of the buffer for each type of calls is readjusted based on the average call waiting time W_i, which is obtained by

$$W_i = \frac{T_{W,i}}{CR_{D,i}}.$$

If $W_i > \delta^- D_i$, where D_i is obtained from (6), the buffer size Q_i is discounted by one. If $W_i < \delta^+ D_i$ and $Q_i < \theta N_i$, the buffer size Q_i is increased by one.

After repartitioning the code tree and readjusting the buffer sizes, the statuses of the codes are updated by checking their ancestor and descendant codes. Then $CR_{A,i}, CR_{D,i}, T_{A,i}, T_{S,i}$, and $T_{W,i}$ are reset to zero.

B. The CADPB2 Scheme

In CADPB1, a situation may occur, that is, for $i \neq j$, though there are assignable codes in S_j, a type$_i$ request cannot be served (due to no *assignable* code in S_i). For improvement, we propose CADPB2 by slightly modifying CADPB1. The system parameters in CADPB2 are the same as that in CADPB1. The four procedures are modified as follows.

1) *Initial:* Similar to CADPB1, for $i \in \{1, 2, \ldots, m\}$, N_i is initially determined by using (3). The generation of the code set S_i follows (8), which is different from the one used in CADPB1 (i.e., (4))

$$S_i = \left\{ C_{K-\log_2 a_i,l} \,\middle|\, 1 \le l \le \frac{R_T - \sum_{i<j\le m} a_j r N_j}{a_i r} \right\}. \tag{8}$$

   From (8), the ancestors of the codes in the set supporting lower data rate services can also serve higher data rate call requests. Fig. 4 shows an example for the result after this procedure exercises. The nodes in the four rectangles represent the codes in S_i where $i \in \{1, 2, 3, 4\}$.

2) *Arrival:* This procedure is similar to that in CADPB1, except that, when a type$_i$ call request arrives, the network finds an assignable code in S_i by checking the statuses of the codes in S_i from "the right to the left." If an *assignable* code is found, it is assigned to the request. The code is marked as busy, and its ancestor and descendant codes are set to free.

3) *Departure:* When the code $C_{K-\log_2 a_i,j}$ serving the type$_i$ call is released, it is marked as free. The network updates the statuses of the codes in $S_1, S_2, \ldots, S_m$ by checking their ancestor and descendant codes. After this action, for $i = 1, 2, \ldots, m$, the network checks B_i sequentially. If B_i is not empty, the network checks if there are assignable codes in S_i. If such codes exist, the network allocates the codes to the requests queued in B_i, then marks the statuses

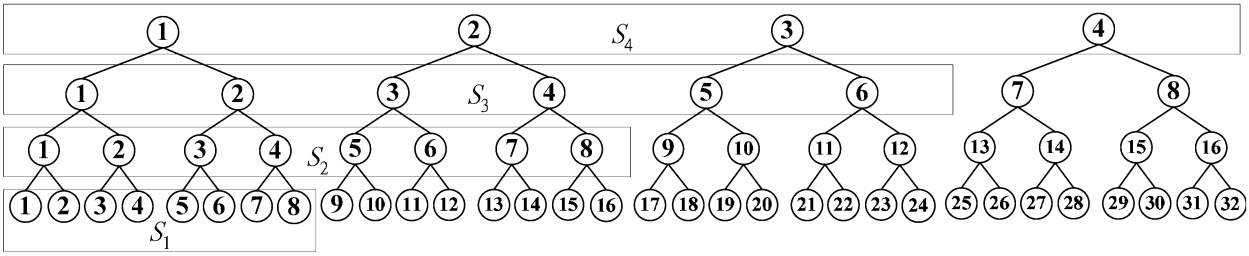

Fig. 4. An example for the result of Initial procedure in CADPB2.

of the allocated codes as busy and modifies the statuses of their ancestor and descendant codes as free.

4) *Adjustment* is similar to that in CADPB1. The buffer sizes are readjusted following the rule used in CADPB1. The sizes of code sets (i.e., N_i) are calculated by using (7). Then N_i is applied to (8) to obtain S_i.

IV. Performance Evaluation

This section compares the performances between our schemes and the previously proposed schemes based on the simulation experiments. We build up the simulation model according to the behavior of our schemes. We adopt the discrete event-driven approach [6], [10]–[11]. The details are not presented here.

We define $P_{b,i}$ for type$_i$ calls as the probability that a type$_i$ call request cannot be served by the network. In our schemes, when a type$_i$ call request arrives, the call request is blocked when there is no assignable OVSF code in S_i, and the B_i buffer is full. In the previous studies [7], [8], [12], the authors used the "code blocking probability" as the performance indicator, which is defined as the probability that "under the condition when the system has excess capacity to satisfy the requested data rate of a call (i.e., $R_A \geq$ the requested data rate), the call is blocked." In [12], the following theorem has been proven.

Theorem 1: If a rearrangeable code allocation scheme is applied, the code blocking would not occur.

In other words, for the rearrangeable code allocation schemes, the code blocking probability is equal to zero. Thus, the measurement of code blocking probability for code rearrangeable scheme cannot clearly show the system capacity. Therefore, for meaningful performance comparison between the rearrangeable scheme "DCA" and our schemes, we use the "call blocking probability for the type$_i$ calls (i.e., $P_{b,i}$)" as the performance indicator, which is defined as the probability that "a type$_i$ call request cannot be served by the network." This occurs when a type$_i$ call request arrives, and the left data rate of the network is less than the requested data rate (i.e., $R_A <$ requested data rate). The call blocking probability can precisely reflect the ratio of call requests that can not be served.

Besides $P_{b,i}$, we also derive the weighted blocking probability γ [12], that is

$$\gamma = \frac{\sum_{1 \leq i \leq m} a_i \lambda_i P_{b,i}}{\sum_{1 \leq i \leq m} a_i \lambda_i}. \tag{9}$$

The intention of (9) is that the damage of QoS (i.e., $P_{b,i}$) for the high data rate calls is higher than that for the low data rate calls. By applying γ, we can more precisely investigate the overall QoS for the network. The fairness index F [14] is used to measure the fairness of code assignment for each type of calls, which is

$$F = \frac{\left(\sum_{1 \leq i \leq m} P_{b,i}\right)^2}{m \sum_{1 \leq i \leq m} (P_{b,i})^2}. \tag{10}$$

In (10), it is clear that $0 \leq F \leq 1$. When the $P_{b,i}$ values for $i \in \{1, 2, \ldots, m\}$ approach to the same value, the F value approaches to one. On the other hand, if the variance of all $P_{b,i}$s where $i \in \{1, 2, \ldots, m\}$ becomes large, F approaches to zero. Therefore, a larger F implies that each type of calls can get service more fairly.

To clearly indicate the improvement of our schemes over the previously proposed schemes, we investigate the performances of the dynamic code assignment (DCA) [12] and First Fit (FFit) [7], [8] schemes based on simulation experiments. In DCA, for an incoming type$_i$ call request, the network checks if there is an assignable code that can serve it. If so, the assignable code is allocated to the request. Otherwise, the network reassigns some calls (being served with low data rate) other OVSF codes to hold a satisfied OVSF code for the request. The FFit scheme is a nonrearrangeable scheme. In FFit, for a type$_i$ call request, the network searches an assignable code supporting the data rate $a_i r$ bits/s from the left to the right. If no such code is found, the request is blocked immediately. This scheme is considered simple.

From Theorem 1, we know that if the code assignment schemes implement the code rearrangement operation following Theorem 1, then they will have the same performance in terms of call blocking probability. Since most rearrangeable schemes (e.g., the TP scheme [7], [8] and the DCA scheme [12]) follow Theorem 1, we implement the simulation model for the DCA scheme (which can show the trend for the call blocking performance of the most rearrangeable schemes) to compare the performance of our schemes and rearrangeable schemes.

In this section, we first consider four types of calls—type$_1$, type$_2$, type$_3$, type$_4$—to investigate the performance trend of our schemes, where requested data rates are $r, 2r, 4r$, and $8r$ bits/s, respectively. The UMTS system is mainly designed to support wireless transmission services for the existing Internet applications (in particular for moving users to access Internet). In our study, the transmission rate $8 \times 7.5 = 60$ kbits/s is sufficient to support streaming video applications (either with .asf or .rm file format). Therefore, we run several scenarios by considering the requested data rate up to $8r$. The input parameter are set up as follows.

1) The threshold Δ is set to 10 000.
2) The δ^- and δ^+ thresholds are set to 1/10 and 1/12, respectively.

3) The requested data rates $a_1 r$ bits/s, $a_2 r$ bits/s, $a_3 r$ bits/s, $a_4 r$ bits/s for the four types of calls are set as $a_1 = 1, a_2 = 2, a_3 = 4, a_4 = 8$.
4) The buffer size Q_i is initially set as $Q_i = (1/2)N_i$.
5) We set the mean $1/\mu_i$ of the service times for the type$_i$ calls as

$$\frac{1}{\mu_1} = \frac{1}{\mu_2} = \frac{1}{\mu_3} = \frac{1}{\mu_4} = \frac{1}{\mu}. \tag{11}$$

6) We normalize λ_i by μ. That is, during the time period $1/\mu$, averagely, $\lambda_i$type$_i$ calls arrive at a cell, and $\lambda_1, \lambda_2, \lambda_3$, and λ_4 have the following relationship:

$$\lambda_1 : \lambda_2 : \lambda_3 : \lambda_4 = \alpha_1 : \alpha_2 : \alpha_3 : \alpha_4 \tag{12}$$

where $\alpha_1, \alpha_2, \alpha_3$, and α_4 are constant numbers.
7) The total data rate R_T supported by a Node B is $256r$ bits/s.
8) We define the total traffic load ρ_T

$$\rho_T = \sum_{1 \le i \le 4} \rho_i = \sum_{1 \le i \le 4} a_i \left(\frac{\lambda_i}{\mu_i} \right). \tag{13}$$

Note that, in the previous studies [7], [8], [12], the following equation is used to interpret the traffic load.

$$\frac{\Lambda}{\mu} = \frac{\sum_{i=1}^{m} \lambda_i}{\mu}$$

where Λ is the sum of arrival rate for different types of calls, and $1/\mu$ is the mean of service time per call. To allow for a transparent comparison between our study and the previous studies [7], [8], [12], in the simulation results, we use Λ/μ to interpret the total traffic load. The following equations are used to map the input parameters in our study to that in the previous studies [7], [8], [12]:

$$\rho_T = \frac{\Lambda}{\mu} \left[\sum_{i \in \{1,2,\ldots,m\}} \frac{a_i \alpha_i}{\left(\sum_{j \in \{1,2,\ldots,m\}} \alpha_j \right)} \right]$$

or

$$\frac{\Lambda}{\mu} = \frac{\rho_T}{\left[\sum_{i \in \{1,2,\ldots,m\}} \left(\frac{a_i \alpha_i}{\sum_{j \in \{1,2,\ldots,m\}} \alpha_j} \right) \right]}.$$

From (11)–(13), λ_i can be expressed as

$$\lambda_i = \frac{\alpha_i \rho_T}{\sum_{1 \le j \le 4} a_j \alpha_j}. \tag{14}$$

Applying (12)–(14) into (7), the size of each code set N_i can be obtained as follows:

$$\begin{aligned}
N_4 &= \text{round}\left(\frac{a_4 \alpha_4}{a_1\alpha_1 + a_2\alpha_2 + a_3\alpha_3 + a_4\alpha_4} \times \frac{R_T}{a_4 r} \right) \\
N_3 &= \text{round}\left(\frac{a_3 \alpha_3}{a_1\alpha_1 + a_2\alpha_2 + a_3\alpha_3} \times \frac{R_T - a_4 r N_4}{a_3 r} \right) \\
N_2 &= \text{round}\left(\frac{a_2 \alpha_2}{a_1\alpha_1 + a_2\alpha_2} \times \frac{R_T - a_4 r N_4 - a_3 r N_3}{a_2 r} \right) \\
N_1 &= \text{round}\left(\frac{R_T - a_4 r N_4 - a_3 r N_3 - a_2 r N_2}{a_1 r} \right).
\end{aligned} \tag{15}$$

In our study, we consider four scenarios for the λ_i setups, which have been widely adopted in other pervious studies [7], [8], [12]:

Scenario I: $\lambda_1 : \lambda_2 : \lambda_3 : \lambda_4 = 1 : 1 : 1 : 1$;
Scenario II: $\lambda_1 : \lambda_2 : \lambda_3 : \lambda_4 = 4 : 1 : 1 : 4$;
Scenario III: $\lambda_1 : \lambda_2 : \lambda_3 : \lambda_4 = 1 : 4 : 4 : 1$;
Scenario IV: $\lambda_1 : \lambda_2 : \lambda_3 : \lambda_4 = 4 : 4 : 1 : 1$.

1) *Comparison for Call Blocking Probability $P_{b,i}$:* Fig. 5 compares the call blocking probability $P_{b,i}$ of the type$_i$ calls for CADPB1, CADPB2, DCA, and FFit, where Scenario I is considered. In this figure, the total traffic load Λ/μ ranges from 42.66 to 66.66 (i.e., ρ_T ranges from 160 to 250). This figure indicates that the $P_{b,1}, P_{b,2}, P_{b,3}$, and $P_{b,4}$ values for CADPB1 are almost the same (see "$\cdot$" curves). CADPB1 partitions the code tree according to the traffic load for each type of calls, and since $\lambda_1 : \lambda_2 : \lambda_3 : \lambda_4 = 1 : 1 : 1 : 1$, the sizes of four code sets are the same. Thus the four types of calls could be served with equal opportunity. In CADPB2 (see "$\star$" curves), the blocking probability for the high data rate call requests is lower than that for the low data rate call requests (i.e., $P_{b,4} < P_{b,3} < P_{b,2} < P_{b,1}$). This is because the ancestors of the codes in the set supporting low data rate services can also serve high data rate call requests. For DCA (see "$\diamond$" curves) and FFit (see "$\circ$" curves), we observe that $P_{b,1} < P_{b,2} < P_{b,3} < P_{b,4}$. This is due to the fact that DCA and FFit do not partition the code tree, and from the criteria of OVSF code assignment, we know that the low data rate call requests are more likely to be served than high data rate call requests. The $P_{b,i}$ performances for the four schemes are in the following order:

$$\begin{aligned}
P_{b,1}(\text{FFit}) &< P_{b,1}(\text{DCA}) < P_{b,1}(\text{CADPB1}) \\
&< P_{b,1}(\text{CADPB2}) \\
P_{b,2}(\text{FFit}) &< P_{b,2}(\text{DCA}) < P_{b,2}(\text{CADPB1}) \\
&\approx P_{b,2}(\text{CADPB2}) \\
P_{b,3}(\text{FFit}) &< P_{b,3}(\text{DCA}) \approx P_{b,3}(\text{CADPB1}) \\
&\approx P_{b,3}(\text{CADPB2}) \\
P_{b,4}(\text{CADPB2}) &< P_{b,4}(\text{CADPB1}) < P_{b,4}(\text{DCA}) \\
&< P_{b,4}(\text{FFit}).
\end{aligned}$$

From Fig. 5(d), DCA has much lower blocking rate for high data rate calls than FFit has. This is due to the fact that the reassignment operations are performed in DCA. In other words, with DCA, high data rate calls are more likely to occupy the OVSF codes, with the result that low data rate calls may not be served. Instead, in FFit, since the high data rate calls has worse chance to be served, more OVSF codes are left to serve low data rate requests. Thus, we observe the phenomenon shown in Fig. 5(a). To summarize, the $P_{b,4}$ performances for CADPB1 and CADPB2 are better than that of DCA and FFit. On the other hand, with FFit and DCA, the low data rate call requests has better chance to be served than high data rate call requests.

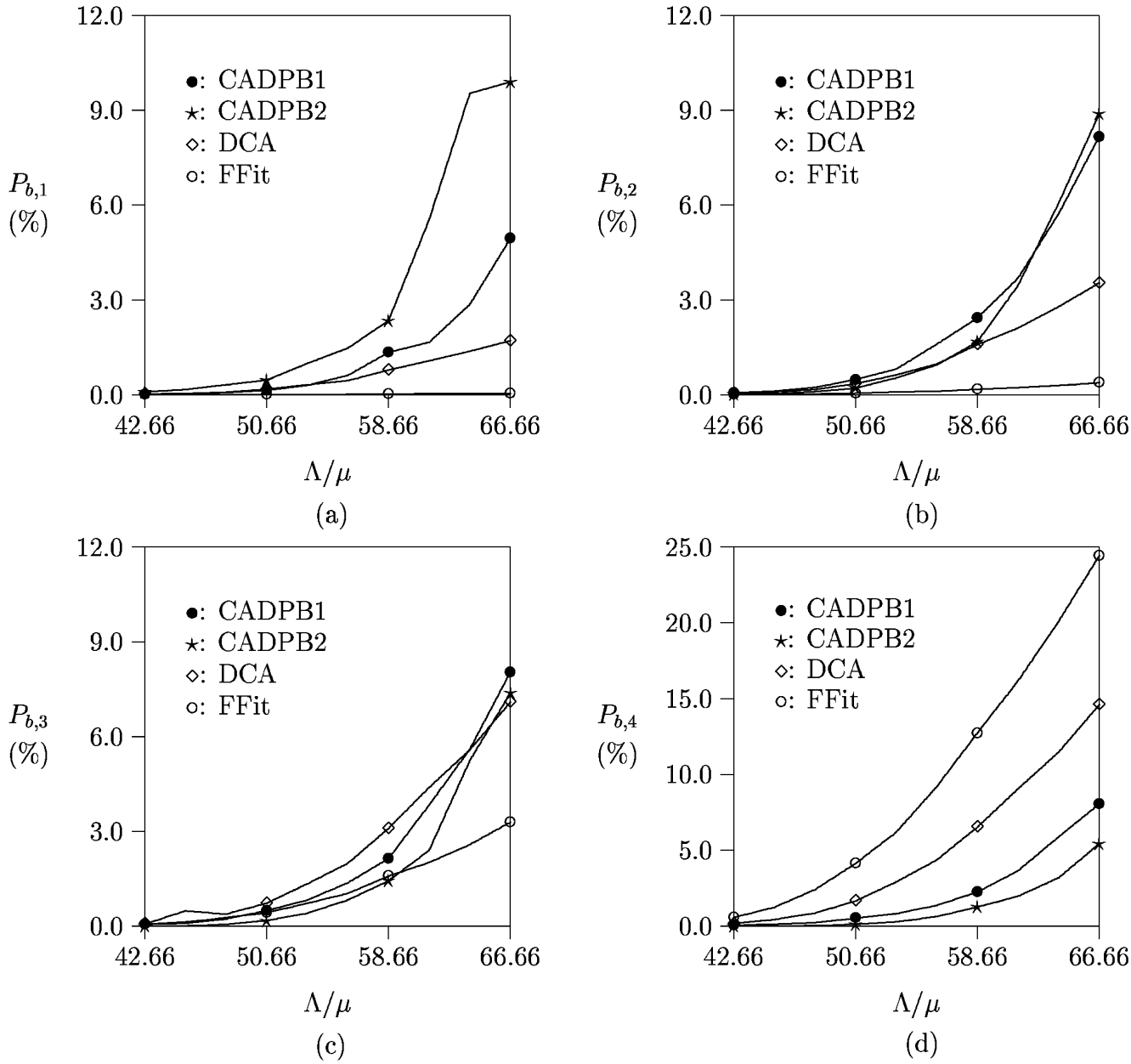

Fig. 5. Comparison for the call blocking probability ($R_T = 256r$ bits/s; $\Delta = 10\,000$; $\lambda_1 : \lambda_2 : \lambda_3 : \lambda_4 = 1 : 1 : 1 : 1$). (a) Blocking probability for type$_1$ calls. (b) Blocking probability for type$_2$ calls. (c) Blocking probability for type$_3$ calls. (d) Blocking probability for type$_4$ calls.

2) *Comparison for Weighted Blocking Probability γ:* Fig. 6 compares the weighted blocking probability γ for CADPB1, CADPB2, DCA, and FFit by considering Scenarios I–IV. In this figure, the total traffic load Λ/μ ranges from 42.66 to 66.66, from 38.10 to 59.52, from 48.48 to 75.75, and from 66.66 to 104.16 for Scenario I, Scenario II, Scenario III, and Scenario IV, respectively, that is, ρ_T ranges from 160 to 250 for Scenarios I–IV. From (9), we know that the blocking probability for the high data rate call is assigned large weight and affects the γ performance significantly. In CADPB1 and CADPB2, the code sets for different types of calls are dynamically partitioned based on (7), i.e., according to the ratio of the measured traffic load from Δ served calls (including four types). There is an error existing between the measured traffic load and actual traffic load. Especially when Δ is small, the error becomes significant. This error causes that the code tree cannot be partitioned precisely. For example, based on the actual traffic load, the code set size of the type$_2$ calls should be four. However, based on the measured traffic load, there are actually three codes in the code set. The error may significantly degrade the γ performance for the types of calls whose code set is small. In FFit or DCA, no tree partition is performed. The four types of calls compete with each other for the OVSF codes. The calls having larger arrival rate are more likely to be served than that having smaller arrival rate.

In Scenario I [see Fig. 6(a)], CADPB1 and CADPB2 significantly outperform both DCA and FFit in terms of the γ performance. As shown in Fig. 5, in Scenario I,

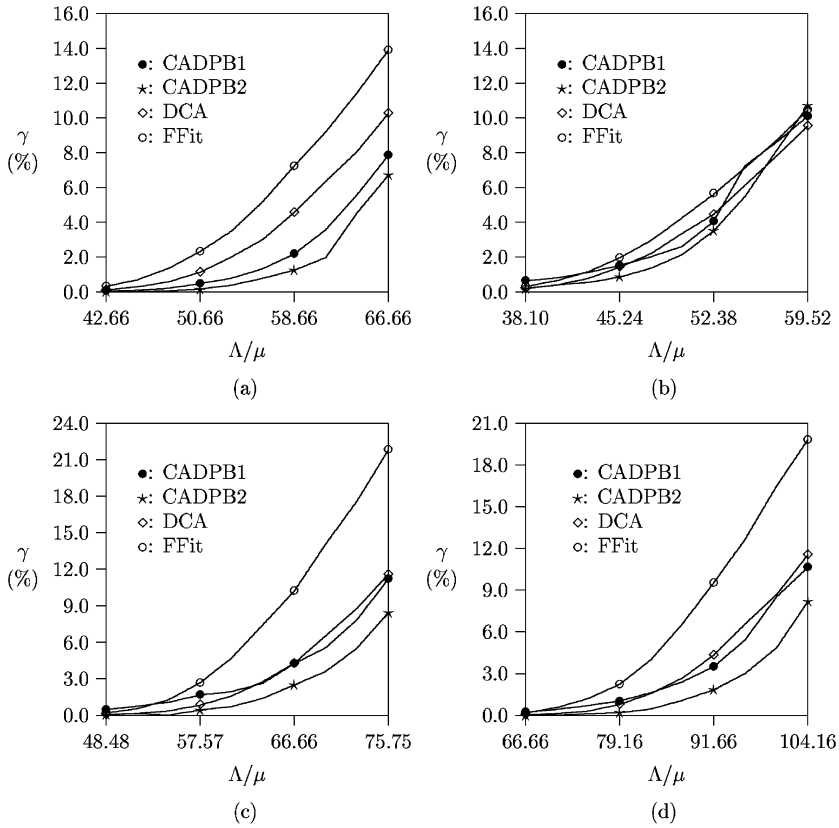

Fig. 6. Comparison for the weighted blocking probability ($R_T = 256r$ bits/s; $\Delta = 10\,000$). (a) $\lambda_1 : \lambda_2 : \lambda_3 : \lambda_4 = 1 : 1 : 1 : 1$. (b) $\lambda_1 : \lambda_2 : \lambda_3 : \lambda_4 = 4 : 1 : 1 : 4$. (c) $\lambda_1 : \lambda_2 : \lambda_3 : \lambda_4 = 1 : 4 : 4 : 1$. (d) $\lambda_1 : \lambda_2 : \lambda_3 : \lambda_4 = 4 : 4 : 1 : 1$.

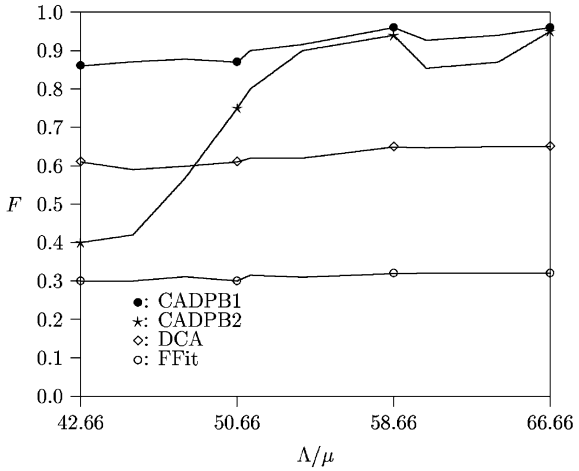

Fig. 7. Comparison for the fairness index ($R_T = 256r$ bits/s; $\Delta = 10\,000$; $\lambda_1 : \lambda_2 : \lambda_3 : \lambda_4 = 1 : 1 : 1 : 1$).

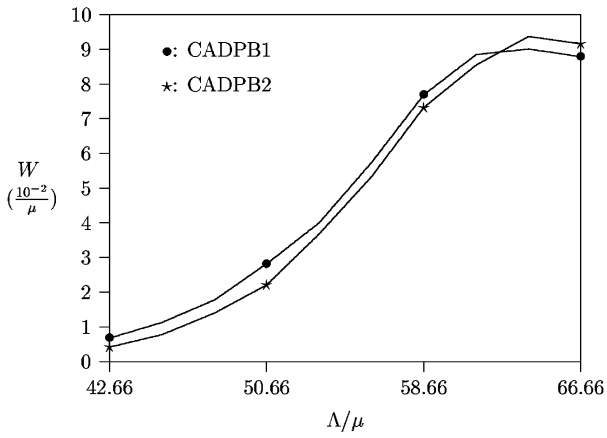

Fig. 8. Average call waiting times for the CADPB1 and CADPB2 schemes ($R_T = 256r$ bits/s; $\Delta = 10\,000$; $\lambda_1 : \lambda_2 : \lambda_3 : \lambda_4 = 1 : 1 : 1 : 1$).

CADPB1 and CADPB2 have smaller $P_{b,4}$ values than that other schemes have. With lower blocking probability for high data rate calls, our schemes offer the better γ performance in Scenario I. The γ performances for the four schemes in Scenario I have the order: $\gamma(\text{CADPB2}) < \gamma(\text{CADPB1}) < \gamma(\text{DCA}) < \gamma(\text{FFit})$. For example, when $\Lambda/\mu = 66.66$ (i.e., $\rho_T = 250$), CADPB2 has 14.1%, 34.3%, and 51.8% improvement over CADPB1, DCA, and FFit in terms of the γ performance, respectively.

When λ_1 and λ_4 increase [i.e., Scenario II; see Fig. 6(b)] compared with Scenario I [Fig. 6(a)] the γ performance for FFit gets better, the γ performance for DCA is slightly improved, the γ performance for our schemes become worse, and thus the γ performances for the four schemes are almost identical. In Scenario II, $\lambda_1 : \lambda_2 : \lambda_3 : \lambda_4 = 4 : 1 : 1 : 4$, which results in that the code set sizes for type$_2$ and type$_3$ calls decrease and become much smaller compared with that in Scenario I. Hence, $P_{b,2}$ and $P_{b,3}$ increase, and we observe higher γ values for CADPB1 and CADPB2 in Fig. 6(b). In Scenario II, since $\lambda_1 : \lambda_2 : \lambda_3 : \lambda_4 = 4 : 1 : 1 : 4$, with DCA or FFit, the type$_4$ calls get better chance to be served than that in Scenario I [see Fig. 6(a)] and $P_{b,4}$ deceases. Since

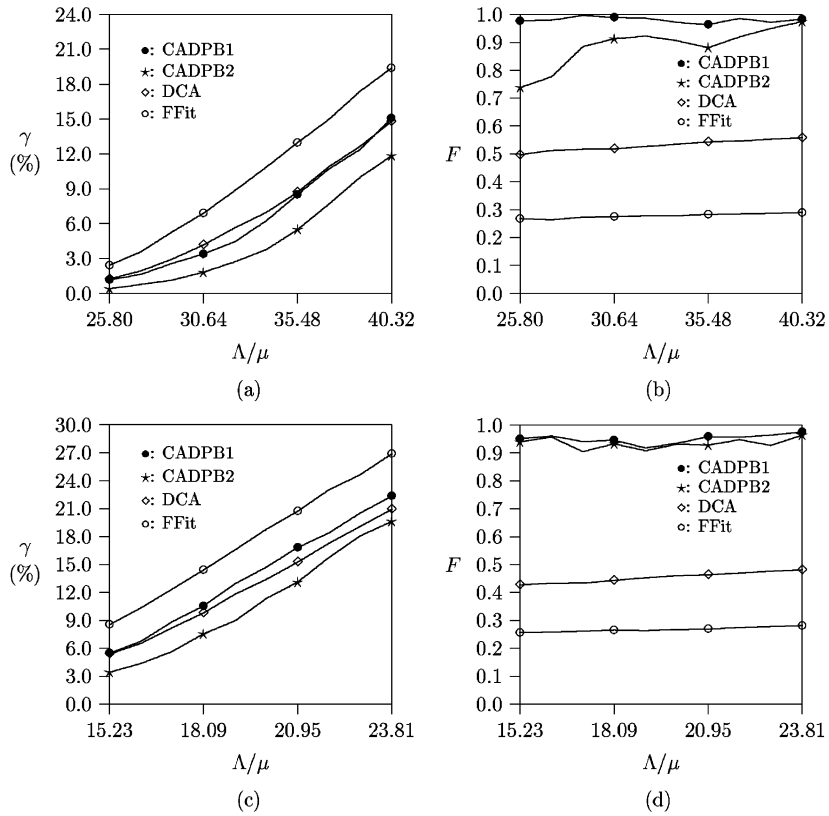

Fig. 9. The γ and F performances for more traffic types ($R_T = 256r$ bits/s; $\Delta = 10\,000$). $\lambda_1 : \lambda_2 : \lambda_3 : \lambda_4 : \lambda_5 = 1 : 1 : 1 : 1 : 1$. (b) $\lambda_1 : \lambda_2 : \lambda_3 : \lambda_4 : \lambda_5 = 1 : 1 : 1 : 1 : 1$. (c) $\lambda_1 : \lambda_2 : \lambda_3 : \lambda_4 : \lambda_5 : \lambda_6 = 1 : 1 : 1 : 1 : 1 : 1$. (d) $\lambda_1 : \lambda_2 : \lambda_3 : \lambda_4 : \lambda_5 : \lambda_6 = 1 : 1 : 1 : 1 : 1 : 1$.

the γ performance is dominated by the call blocking probability for high data rate calls, we observe the better γ performance for FFit and DCA in Scenario II.

The results in Fig. 6(c) (i.e., Scenario III) are similar to that in Fig. 6(d) (i.e., Scenario IV). The two figures indicate that the γ performances for CADPB1 and DCA are almost identical. The CADPB2 scheme has the best γ performance among the four schemes. Compared with Fig. 6(a), the γ values for CADPB1 and CADPB2 increase in these two scenarios. This is due to that the arrival rates for the four types of calls are not well mixed. For FFit and DCA, due to the smaller arrival rate for the type$_4$ calls in these two scenarios, we observe that the γ values for FFit and DCA become large.

To conclude, the CADPB2 scheme outperforms other schemes in terms of the γ performance in most cases. CADPB1 and DCA have the similar γ performance.

3) *Comparison for Fairness Index F:* Fig. 7 compares the fairness index F for CADPB1, CADPB2, DCA, and FFit, where Scenario I is considered. The figure shows that in FFit, DCA, and CADPB1, the F values do not change significantly as Λ/μ increases, but in CADPB2, F increases as Λ/μ increases. The CADPB1 scheme has the best F performance among the four schemes. When $42.66 \leq \Lambda/\mu \leq 48.0$ (i.e., $160 \leq \rho_T \leq 180$), the F performances for CADPB2, DCA, and FFit are in the order $F(\text{DCA}) > F(\text{CADPB2}) > F(\text{FFit})$. When $50.66 \leq \Lambda/\mu \leq 66.66$ (i.e., $190 \leq \rho_T \leq 250$), CADPB2 outperforms DCA and FFit in terms of the F performance. When Λ/μ increases, the F performance for CADPB1 gets closer to that for CADPB2. In CADPB2, the ancestors of the codes in the set supporting low data rate services can also serve high data rate call requests, which results in unfairness for different types of calls. When Λ/μ is low (i.e., $42.66 \leq \Lambda/\mu \leq 48.0$), the traffic load is not saturated, and the codes in the set supporting low data rate services are more likely free. Thus, the ancestors of these codes have high opportunity to serve high data rate calls. On the other hand, as the traffic load is saturated (i.e., $50.66 \leq \Lambda/\mu \leq 66.66$), it is more likely that there are call requests queued in B_i. The codes in the set supporting low data rate services are more likely to be allocated to the queued low data rate call requests. In summary, when traffic load is saturated, the behavior of CADPB2 is like that of CADPB1.

4) *Average Call Waiting Time for CADPB1 and CADPB2:* In CADPB1 and CADPB2, the buffering mechanism with dynamic buffer-size-adjustment is used to queue the calls that cannot be served immediately. In Fig. 8, we investigate the average call waiting time W for call requests (including four types) being queued in the buffer. The figure shows that W increases as Λ/μ increases. The W performances for CADPB1 and CADPB2 are almost identical, which are bounded by the range $0.01/\mu < W < 0.1/\mu$. The result indicates that our buffering mechanism improves the call acceptance rate by slightly introducing call waiting time.

5) *Performance Comparison for More Traffic Types:* Fig. 9 compares the γ and F performances for CADPB1,

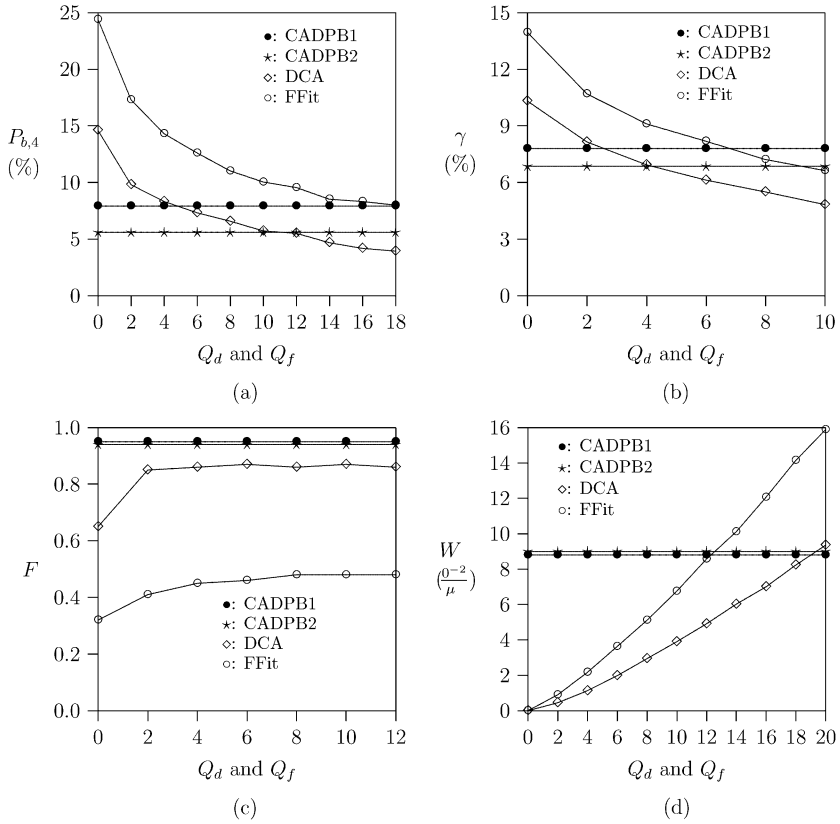

Fig. 10. The effects of buffering ($\lambda_1 : \lambda_2 : \lambda_3 : \lambda_4 = 1 : 1 : 1 : 1; \Lambda/\mu = 66.66; R_T = 256r$ bits/s; $\Delta = 10\,000$). (a) The $P_{b,4}$ performance. (b) The γ performance. (c) The F performance. (d) The W performance.

CADPB2, DCA and FFit when more than four traffic types are considered. In Fig. 9(a) and (b), we consider five traffic types, where the data rates requested by the $\text{type}_1, \text{type}_2, \text{type}_3, \text{type}_4$, and type_5 calls are $r, 2r, 4r, 8r$, and $16r$ bits/s, respectively, and $\lambda_1 : \lambda_2 : \lambda_3 : \lambda_4 : \lambda_5 = 1 : 1 : 1 : 1 : 1$. The total traffic load Λ/μ ranges from 25.80 to 40.32,i.e., ρ_T ranges from 160 to 250.

Compare Figs. 9(a) and 6(a), where the network traffic load (i.e., ρ_T) has the same range. We observe that the trends for the γ performances for the four schemes in the two figures are similar. CADPB2 outperforms CADPB1, DCA, and FFit. The γ performances for CADPB1 and DCA are almost the same when there are five traffic types [see Fig. 9(a)]. This is because, in CADPB1, the number of codes used to support each type of calls decreases (i.e., the size of each code set gets smaller) as the number of traffic types increases. The call requests are more likely to be blocked, which results in higher γ.

Comparing Figs. 9(b) and 7, we observe that the trends for the F performances for the four schemes are similar. The difference is that as the number of traffic types increases, CADPB2 outperforms DCA in terms of the F performance in most cases [see Fig. 9(b)].

Moreover, we compare the γ and F performances for six traffic types in Fig. 9(c) and (d), respectively. The data rates requested by the $\text{type}_1, \text{type}_2, \text{type}_3, \text{type}_4, \text{type}_5$, and type_6 calls are $r, 2r, 4r, 8r, 16r$, and $32r$ bits/s, respectively, and $\lambda_1 : \lambda_2 : \lambda_3 : \lambda_4 : \lambda_5 : \lambda_6 = 1 : 1 : 1 : 1 : 1 : 1$. The total traffic load Λ/μ ranges from 15.23 to 23.81, i.e., ρ_T ranges from 160 to 250.

Compare Fig. 9(a) and (c), where the total traffic loads ρ_T are in the same range, and we observe that as the number of traffic types increases, the γ values become large. The γ performances for the four schemes have the similar trend shown in Fig. 9(a). Fig. 9(d) shows that the F values of CADPB2 get closer to that of CADPB1 in this figure. To conclude, as the number of traffic types increases, CADPB1 is still the scheme that can most fairly allocate the OVSF codes to each type of calls, and CADPB2 has the best γ performance.

6) *Effects of Buffering:* One may question that our comparison may not be fair since the DCA and FFit schemes are not implemented buffering. For this issue, we also apply the buffering on the DCA and FFit. As shown in Fig. 5, both DCA and our proposed schemes attempt to reduce the blocking probability for the high data rate calls. Here, we address the $P_{b,4}, \gamma, F$, and W performance comparison for DCA, FFit, CADPB1, and CADPB2 while buffering is also applied in DCA and FFit. Let Q_d and Q_f be the buffer sizes of DCA and FFit, respectively.

Fig. 10 shows the results, where Scenario I is considered, the total data rate is set to $256r$ bits/s, and the total traffic load $\Lambda/\mu = 66.66$ (i.e., $\rho_T = 250$). For other scenarios, we observe the similar results, which are not presented here. In Fig. 10(a), we observe that the $P_{b,4}$ values for DCA and FFit decrease as Q_d and Q_f increase. When Q_f and Q_d are less than 12, CADPB2 still has the

better $P_{b,4}$ performance than that of DCA and FFit. When Q_f and Q_d are larger than 12, DCA starts to outperform CADPB2 in terms of $P_{b,4}$. Fig. 10(b) indicates that as Q_f and Q_d increase, the γ values for DCA and FFit drop. When $Q_d < 4$, CADPB2 outperforms DCA and FFit. As shown in Fig. 10(c), for the F performance, when Q_d and Q_f are less than four, the F values for DCA and FFit increase as Q_d and Q_f increase. When Q_d and Q_f are larger than four, the change of the F values for DCA and FFit is insignificant, which are smaller than that of CADPB1 and CADPB2. The CADPB1 and CADPB2 more fairly allocate OVSF codes to different types of calls than that DCA or FFit does. Fig. 10(d) shows that the W values for DCA and FFit increase as Q_f and Q_d increase. When $Q_d = 12$ and $Q_f = 19$, the W values for DCA, FFit, CADPB1, and CADPB2 are almost the same.

In summary, the $P_{b,4}, \gamma$, and F performances for DCA and FFit are improved as the buffering mechanism is applied. As the buffer sizes are large enough, DCA and FFit outperform CADPB2 and CADPB1 in terms of the $P_{b,4}$ and γ performances. For the F performance, CADPB2 and CADPB1 outperform DCA and FFit in all cases.

However DCA is a rearrangeable scheme. As we pointed out in Section I, there are drawbacks for rearrangement operations[1] including increase of system complexity and extra message exchanges, and decrease of the QoS of the system. For the FFit scheme, although it is a nonrearrangeable scheme, it cannot fairly allocate the OVSF codes to different types of calls. Especially for high data rate call requests, they are easy to block.

On the other hand, the proposed CADPB1 and CADPB2 are nonrearrangeable schemes. With dynamically partitioning OVSF codes (according to the traffic loads of different types of calls), CADPB1 and CADPB2 more fairly serve different types of calls. With dynamic buffering mechanism, the γ performance for CADPB1 and CADPB2 is better than that of DCA and FFit in some cases. The proposed schemes are also simple and easy to deploy in the real system.

V. Conclusion

In this paper, we proposed two nonrearrangeable OVSF code assignment schemes CADPB1 and CADPB2 for UMTS. Both are simple, cost effective, and spectrally efficient for implementation. Many previous papers [7], [8], [12] addressed design rearrangeable schemes such that the number of the reassignment operations can be reduced, which may increase the complexity of the designed schemes, and lead the schemes not easily to be deployed in the real system. On the other hand, few studies addressed design nonrearrangeable schemes that can provide the QoS as good as or better than the rearrangeable schemes.

Simulation experiments were used to investigate the performances for our proposed schemes. We compared the performances of our schemes with those of previously proposed schemes: DCA and FFit. Our study indicated the following.

1) The CADPB2 scheme outperforms CADPB1, DCA, and FFit in terms of the weighted blocking probability γ in most cases.
2) Among the four schemes, the CADPB1 scheme has the most fairness for the OVSF code assignment for different types of calls.
3) The dynamic buffering mechanism in CADPB1 and CADPB2 improves the call acceptance rate by slightly introducing call waiting time.
4) The $P_{b,4}, \gamma$, and F performances for DCA and FFit are improved while buffering mechanism is applied on them.
5) As the buffer sizes are large enough, DCA and FFit outperform CADPB2 and CADPB1 in terms of the $P_{b,4}$ and γ performances.
6) For the F performance, either DCA and FFit with buffering or not, CADPB2 and CADPB1 outperform DCA and FFit in all cases.

To conclude, our study carried out a detailed comparison for DCA, FFit, CADPB1, and CADPB2. System developers, based on the tradeoffs (given in our study), may choose suitable schemes to be deployed in the real UMTS systems.

[1]Note that in [12, Table I] (where Scenario I is considered, and the R_T and Λ/μ are set to $256r$ bits/s and 16, respectively), the authors have shown that on average 0.39 calls should be reassigned OVSF codes to accommodate a new call.

Acknowledgment

The authors would like to thank the three anonymous reviewers, whose comments have significantly improved the quality of this paper.

References

[1] 3rd Generation Partnership Project, "Radio interface protocol architecture,", Tech. Rep. 3G TS 25.301 version 3.4.0, Mar. 2000.
[2] 3rd Generation Partnership Project, "Technical specification group radio access network; Spreading and modulation (FDD),", Tech. Rep. 3G TS 25.213 version 3.5.0, Mar. 2001. 3GPP.
[3] 3rd Generation Partnership Project, "Technical specification group radio access network; Spreading and modulation (TDD),", Tech. Rep. 3G TS 25.223 version 3.5.0, Mar. 2001. 3GPP.
[4] 3rd Generation Partnership Project, "Technical specification group services and systems aspects; General packet radio service (GPRS); Service description; Stage 2,", Tech. Rep. 3G TS 23.060 version 4.1.0, Jun. 2001. 3GPP.
[5] R. Assarut, K. Kawanishi, U. Yamamoto, Y. Onozato, and M. Matsushita, "Region division assignment of orthogonal variable-spreading-factor codes in W-CDMA," in *Proc. IEEE VTC2001*, 2001, pp. 1884–1888.
[6] R.-G. Cheng and P. Lin, "OVSF code channel assignment for IMT-2000," in *Proc. IEEE VTC2000-Spring*, 2000, pp. 2188–2192.
[7] M. Dell'Amico, M. L. Merani, and F. Maffioli, "Efficient algorithms for the assignment of OVSF codes in wideband CDMA," in *IEEE Int. Conf. 2002*, 2002, pp. 3055–3060.
[8] ——, "A tree partitioning dynamic policy for OVSF codes assignment in wideband CDMA," *IEEE Trans. Wireless Commun.*, vol. 3, pp. 1013–1017, Jul. 2004.
[9] H. Holma and A. Toskala, *WCDMA for UMTS*. New York: Wiley, 2000.
[10] P. Lin and Y.-B. Lin, "Channel allocation for GPRS," *IEEE Trans. Veh. Technol.*, vol. 50, pp. 375–387, Mar. 2001.
[11] P. Lin, Y.-B. Lin, and J.-Y. Jeng, "Improving GSM call completion by call reestablishment," *IEEE J. Sel. Areas Commun.*, vol. 17, pp. 1305–1317, Jul. 1999.
[12] T. Minn and K.-Y. Siu, "Dynamic assignment of orthogonal variable-spreading-factor codes in W-CDMA," *IEEE J. Sel. Areas Commun.*, vol. 18, pp. 1429–1439, Aug. 2000.

[13] F. Shueh and W.-S. Eric Chen, "Code assignment for IMT-2000 on forward radio link," in *Proc. IEEE VTC2001-Fall*, 2001, pp. 906–910.
[14] Y. Yang and T.-S. P. Yum, "Nonrearrangeable compact assignment of orthogonal variable-spreading factor codes for multi-rate traffic," in *Proc. IEEE VTC2001-Fall*, 2001, pp. 906–910.

Phone Lin (M'02) received the B.S.C.S.I.E. and Ph.D. degrees from National Chiao Tung University, Taiwan, R.O.C., in 1996 and 2001, respectively.

From August 2001 to July 2004, he was an Assistant Professor in the Department of Computer Science and Information Engineering (CSIE), National Taiwan University, Taiwan. Since August 2004, he has been an Associate Professor there. His current research interests include personal communications services, wireless Internet, and performance modeling. He is an Associate Editorial Member for the *WCMC Journal.*

Dr. Lin is a Guest Editor for the IEEE WIRELESS COMMUNICATIONS Special Issue on Mobility and Resource Management. He is actively involved with IEEE INFOCOM'05.P.

Chai-Hien Gan was born in Malaysia in 1971. He received the M.S.C.S.I.E. degree from National Taiwan University, Taipei, Taiwan, R.O.C., in 1996, where he is currently pursuing the Ph.D. degree in the Department of Computer Science and Information Engineering.

His current research interests include mobile computing, personal communications services, and wireless Internet.

Ching-Chi Hsu was born in Taipei, Taiwan, R.O.C., in 1949. He received the B.S. degree in physics from National Tsing Hwa University, Hsishu, Taiwan, in 1971 and the M.S. and Ph.D. degrees in computer engineering from Electrical Engineering Department, National Taiwan University, Taipei, Taiwan, in 1975 and 1982, respectively.

In 1977, he joined the Faculty of the Department of Computer Science and Information Engineering, National Taiwan University, and became an Associate Professor in 1982. During 1987 and 2002, he was first engaged as a Professor and became Chairman of the department. During his tenure at National Taiwan University, he was a Visiting Scholar in the Computer Science Department, Stanford University, from 1984 to 1985. After 25 years at National Taiwan University, he became President of Kai Nan University in 2002. Since February 2004, he has been Executive Vice President of the Institute for Information Industry, in which he is mainly in charge of accelerating the growth of information industry in the whole nation. His research interests include distributed processing of data and knowledge, mobile computing, and wireless networks.