PERFORMANCE IMPROVEMENT IN OVSF BASED CDMA NETWORKS USING FLEXIBLE ASSIGNMENT OF DATA CALLS

Davinder S Saini and Neeru Sharma

Department of Electronics and Communication Engineering, Jaypee University of Information Technology, Waknaghat, Distt. Solan, Himachal Pradesh, INDIA- 173215

davinder.saini@juit.ac.in , neeru.sharma@juit.ac.in

ABSTRACT

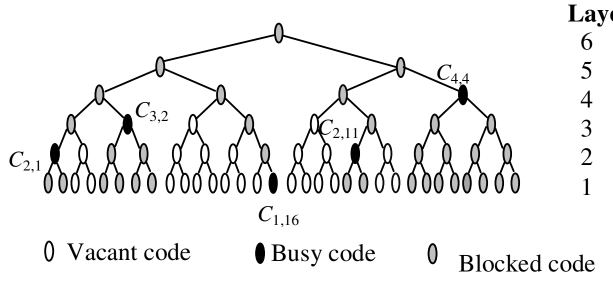
Orthogonal variable spreading factor (OVSF) codes are used to support multimedia calls in CDMA wireless networks. The inefficient use of OVSF code tree reduces the system throughput. The paper discusses a novel code assignment design to provide flexibility of rate variation in data calls maximizing channel utilization. The ongoing call uses different codes at different times depending on the code tree status and instantaneous traffic load. For low traffic loads, the high rate codes can be utilized. For medium to high instantaneous load conditions, the vacant codes with capacity less than the call rate can be utilized. For same call reassignments are done to assign different codes at different times. The utilization of the code tree can reach close to 100%. Simulation results are given to verify the superiority of the proposed design.

KEYWORDS

OVSF codes, code blocking, QoS, spreading factor, OVSF code assignment.

1. INTRODUCTION

In 3G and beyond CDMA wireless networks [1,2], OVSF codes are used to handle multimedia calls. These calls can be delay sensitive like voice, video conferencing or bandwidth sensitive like internet, music download etc. The delay sensitive calls (known as real time calls) are given higher priority compared to bandwidth sensitive calls (known as data calls). While the rate requirement of real time calls is fixed, the rate requirement of data calls is flexible, and hence the rate of data calls can be increased or decreased according to available resources. The different rate calls use different spreading factor (SF) code from the OVSF code tree which is available at each base station (BS) and mobile station (MS). As the number of codes in the OVSF code tree is limited, the efficient use of the code tree is critical for better utilization. While the uplink transmission is dedicated the downlink transmission is shared and because there is only one code tree for each station (BS or MS), the efficient use of the code tree is essential. Further, the multihop nature of wireless networks makes the available bandwidth variable for different hops. For a typical transmission the link bandwidth varies for ongoing call due to arrival and termination of other calls. This requires variable SF at different links in one call transmission. OVSF codes are generated using binary tree [3]. If a code in OVSF tree is used for an incoming call, all its parents and children are blocked from assignment and cannot be utilized by other calls. This is called code blocking property of OVSF codes and illustrated using a six layer code tree as shown in Fig. 1, where code $C_{l,n}$ represents code in layer l with code id n. The tree has five busy codes with code id $C_{1,16}$, $C_{2,1}$, $C_{2,11}$, $C_{3,2}$, and $C_{4,4}$. The total capacity of the tree is $32R$, out of which $17R$ is currently used (due to five busy codes) and $32R$-$17R$=$15R$ is free. Despite of having $15R$ free capacity, if a new call with rate $8R$ comes, it will be rejected by the code tree as there is no 8R capacity vacant code available. This is called *code*



Layer	**SF**	**User Rate**
6	1	32R
5	2	16R
4	4	8R
3	8	4R
2	16	2R
1	32	R

Fig. 1 Illustration of code blocking in a 6 layer OVSF code tree

blocking, and to avoid it, divide 8R call into two 4R rate fractions, and handle each fraction by 4R rate code (which are available in the code tree). Handling of one call with multiple codes requires multiple rake combiners at the receiver end making cost and complexity of the multi code design high. Further, by using the multicode design the tree becomes so much fragmented that it is not able to handle the calls further even if it has the capacity available. This is known as external fragmentation [4] and limits the use of multi code designs. Similarly, suppose a new call of rate 3R arrives and as the tree has no 3R capacity code theoretically, a 4R code is used. This over serving of capacity is known as internal fragmentation [4]. The proposed design has the flexibility of rate variation for data calls making handling of data calls efficient. The number of vacant codes for real time calls also increases providing better code tree utilization.

The paper is organized as follows. Section 2 discusses the related work giving various assignment techniques available in literature. Section 3 describes the system model. Section 4 discusses the proposed design. Section 5 provides simulation results to prove the superiority of the design. The paper is concluded in section 6.

2. Related Work

A large number of code assignment designs are given in literature. The single code assignment designs [5-10] use single code to handle incoming calls. In the leftmost code assignment design (LCA) [5], the code assignment is done from the left side of the code tree. In the fixed set partitioning (FSP) [6] design, the code tree is divided into a number of sub trees according to the input traffic distribution. Both LCA and FSP suffer from high code blocking. In crowded first assignment (CFA) [5], the code is assigned to new call in such a way that the availability of higher rate vacant codes in future is more. In dynamic code assignment (DCA) [7] design, if the call cannot be handled directly, one or more calls currently handled by low rate codes are reassigned to other codes in the same layer. This makes that particular blocked code free whose capacity is equal to the requested rate. The blocked code with least busy children is the optimum candidate for new call. There are several variants to DCA like a DCA design with a greedy call admission control (DCA-CAC) [8], where a call request is never rejected as long as the system can accommodate a new call in addition to calls already in progress through code reassignment. In optimal and suboptimal DCA designs, the assignment and reassignment is done in such a way that the system throughput is improved and the complexity is reduced. Based on complexity and system throughput, FSP with DCA is used to reduce complexity. The design divides all OVSF codes into mutually exclusive groups of codes and uniquely assigns a group to each service class. The number of codes for each class is fixed in this way, and as a result, code reassignment can be avoided. An improvement is done to make the design comparable with DCA where a decision is made regarding the optimal partition of codes to maximize system throughput.

In recursive fewer codes blocked (RFCB) [9] design, code blocking can be reduced with careful selection of optimum code among possible candidate codes during the assignment process. It works on the top of CFA design, and the criterion for the selection of a candidate code is number of parents blocked which were free earlier. The code with least new codes blocked is the one to be selected. Ties are resolved by ordered selection (leftmost or rightmost) among candidate codes. This is a simple way to choose a candidate code, but performance is improved if ties are resolved by recursively searching for the most appropriate branch of the OVSF code tree to place the new call. In fast dynamic code assignment (FDCA) [10], the code assignments are done considering the cost for an OVSF code allocation, where the cost is determined by keeping a track on the number of available and occupied descendant codes of the code. The code is assigned by reassigning occupied descendant codes for a requested data rate with least cost function. The paper [4] describes code blocking in terms of internal fragmentation and external fragmentation. As discussed in section 1, the internal fragmentation is due to quantized nature of the rate handling capability of the OVSF code tree. The external fragmentation is because of the scattering of the vacant codes in the code tree. Both internal and external fragmentations are reduced by applying assignment and reassignment strategies with single code or multiple codes. The multi code assignment design [11] utilizes multiple rakes to handle non quantized data rates, making internal code fragmentation close to zero. The optimum multi code combination is chosen out of a set of multi code options. The multicode multirate compact assignment (MMCA) [12] is based on the concept of compact index, where the objective is to keep the remaining candidate codes in the most compact state after each code assignment without rearranging codes. This can be achieved by finding the candidate codes in the most crowded positions for newly arrived calls and data packets. MMCA considers mobile terminals with multi code transmission capabilities and different quality of service (QoS) requirements. Priority differentiation between multi rate real time traffic and best effort data traffic is also supported in MMCA. A different approach is given in [13] where the service time of the requests are known or estimated at the arrival. The remaining time of each code occupied in the code tree is known. The calls with similar remaining time are allocated to the same subtree making subtree available for higher data rate requests when calls are released. As a result the system is able to support more calls and reduce the code blocking probability. In [14], the concept of flexibility index defined, and based on this index two single code assignment designs namely non rearrangeable and rearrangeable compact assignments are given. Both designs can offer maximal flexibility for the resulting code tree after code assignment. This reduces call blocking probability providing improvement in system throughput and fairness index. Further, to improve the system throughput, a scheduling design is given in [15] that dynamically assign OVSF codes to mobile users on a timeslot basis maximizing system throughput. Also, the average data rate guarantee is provided to each mobile user. There is no need for a mobile user to overbook its required rate. In [16], a code selection design is given that can be used in the assignment process with or without reassignments. The selection is made using a simple measure which differentiates each code irrespective of the incoming or reassigned call rate. It simply counts the number of new codes that will be blocked due to a potential allocation of the candidate code. The average busyness of a call being served is analyzed in [17], and the vacant code selected for assignment is the one whose neighbor is found to be busy for longest duration of time. As a result, the crowded portion remains crowded for longer duration. This makes compact code assignment in the OVSF tree. In [18], the code assignment and reassignment is based on QoS requests. If there are multiple options for the vacant code, the optimum code is one whose ancestor code has the maximum free capacity. The used codes are scattered in the code tree, and therefore, there is space for a call to raise its data rate without the need of a code reassignment. In dynamic code assignment [19], the multimedia rates are differentiated by QoS parameters like delay, jitter, bandwidth and reliability. The multi code design [20] identifies the optimal code for a given set of allocated codes and specified maximal resource wastage ratio. The superior performance of multi code design is achieved using two or three codes with crowded-group-first facility. The code utilization and blocking benefits are achieved for a resource wastage ratio of 40%. The paper [21] discusses credit management and compensation management mechanisms to provide fair access and data rate guarantee. The multiple codes are used for two reasons, (*a*) to compensate terminal encountered errors, (*b*) to adopt an environment with multiple linked states. Additionally, the user is given guaranteed bandwidth support. In [22], the code allocation scheme consists of two parts, namely, rate allocation and code assignment. In the first step the rate for each active connection is decided according to user profiles of the connection and available data in data buffer. In step 2, OVSF code is assigned for each active connection according to the results of the rate allocation in the first step. The paper [23] gives the performance evaluation of the nonblocking codes in WCDMA, where it is shown that although the nonblocking codes don't have code blocking, but cost and complexity is too much for the design to be practically useful. The paper [24] describes a code assignment scheme to reduce the call establishment delay significantly. In this scheme, identifying status of the higher layer codes is sufficient to derive the status of the children codes.

The design proposed in this paper makes rate of data calls (non real time calls) flexible improving code tree utilization. Although the reassignments are required but the amount of reassignments are significantly less than DCA and other related designs. The data call rate can be increased or decreased depending upon available resources.

3. System Model

Most of the multimedia calls fall into two categories, (*a*) real time calls with fixed rate

Table 1 Definition of various symbols and notation

Symbol	Meaning
L	Number of layers in the code tree
$C_{l,n}$	Code in layer l with branch number n, $1 \le n \le 2^{L-l}$, $1 \le l \le L$
C_l	Capacity reserved for class l, where $1 \le l \le L$ for real time calls and $1 \le l \le 2^{L'-1}, L' < L$ for data calls.
C_l^u	Used capacity in layer l
C_{max}	Maximum capacity of the code tree, $C_{max}=2^{L-1}$
G_j	Amount of data transmitted for j^{th} bandwidth variation
$C_{\max}^l$	Maximum capacity available for class l users
$C_{\max}^{r,i}$	Maximum capacity reserved for i^{th} real time call.
$C_{\max}^{d,i}$	Maximum capacity reserved for i^{th} data call.
λ_l	Average arrival rate of l^{th} class calls.
$1/\mu_l$	Call duration for l^{th} class calls.
$\rho_l = \lambda_l / \mu_l$	Traffic load for l^{th} class calls.
A_i	Scaling/Suppression factor of data calls for i^{th} link.
T_l	Capacity threshold signifying lower rate usage condition.
r	Number of rakes in the system.
$t_{p_i} = d_i / v$	Propagation time for i^{th} link with length d_i and v is the speed of light
$I_{l,n}$	Code index for code $C_{l,n}$
N_r	Total number real time classes
N_d	Total number of data classes
N_m	Total number of mixed classes

requirement, (*b*) data calls with flexibility of rate variation depending upon resource availability. The treatment of single hop and multihop networks are different.

2.1. Single hop networks

The various symbols and notation used in the paper are given in Table 1. The single hop network is shown in Fig. 2(*a*). If a new call with rate $2^l R$ arrives, the code assignment depends upon type of call (real time or data), code tree status and link bandwidth. For real time call with rate $2^l R$, if the link bandwidth $B \geq 2^l R$, the call is handled, otherwise the call is rejected.

If there are M bandwidth variations B_j, $j \in [1,M]$ for one call and G_j represents data transmitted in j^{th} bandwidth share, the average bandwidth of channel for call $2^l R$ is given by

$$B = \frac{\sum_{j=1}^{M} B_j}{M} \tag{1}$$

Also, the total transmitted data G considering all rate variations can be represented as

$$G = \sum_{j=1}^{M} G_j \tag{2}$$

For incoming data call with rate $2^l R$, if the maximum capacity of single vacant code is $2^{l'} R$, the channel transmission rate can have following variants, (*i*) if $2^{l'} R \geq 2^l R$, and $B \geq 2^{l'} R$, the transmission rate is $2^{l'} R$ with scaling factor $2^{l'-l}$, (*ii*) if $2^{l'} R \geq 2^l R$, and $2^l R \leq B \leq 2^{l'} R$, the

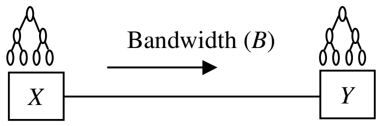


Fig. 2 (*a*) Single hop communication

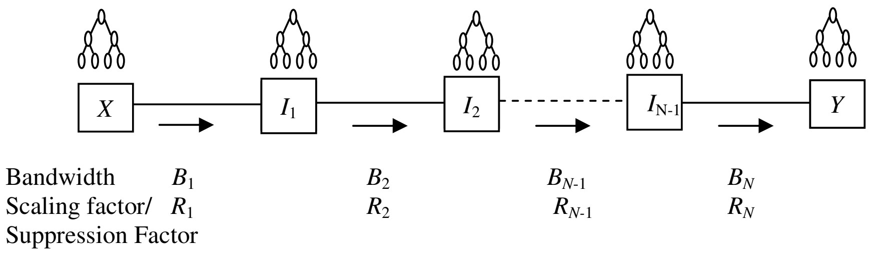


Fig. 2(*b*) Multi hop communication

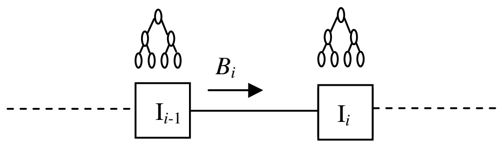


Fig. 2(*c*) i^{th} multihop link

transmission rate is $2^{\lfloor \log_2 B \rfloor} R$ with scaling factor $2^{\lfloor \log_2 B \rfloor - l}$ and the vacant code with capacity $2^{\lfloor \log_2 B \rfloor} R$ is used, (*iii*) if $2^{l'} R > 2^l R$, and $B < 2^l R$, the call is rejected because the channel has insufficient bandwidth to handle call, (iv) If $2^{l'} R < 2^l R$, and $B \geq 2^l R$, the call can be handled provided that the average traffic load is less than the maximum capacity of code tree.

If there are total L call classes, and the average arrival rate and call duration of lth class is λ_l and $1/\mu_l$ respectively, the traffic load for lth class is given by

$$\rho_l = \lambda_l / \mu_l \tag{3}$$

The average traffic load is

$$\rho = \sum_{l=1}^{L} (\lambda_l / \mu_l) \tag{4}$$

Let the maximum capacity of the code tree is C_{max}. If the system has zero code blocking, the new call with rate $2^l R$ can utilize low rate codes in layers 1 to l-1 if

$$\sum_{l=1}^{L} \rho_l + 2^l R / C_{\max} < 1 \tag{5}$$

If the system has average code blocking P_B, the code from lower layers can be utilized if

$$\sum_{l=1}^{L} \rho_l + 2^l R / C_{\max} + P_B < 1 \tag{6}$$

This makes the overall utilization of the code tree close to 100%. Let capacity threshold T_l represents fraction of total capacity C_{max} which prompts the use of codes in layers 1 to l-1. If total load ρ and P_B are known, ideal value of T_l for zero blocking condition is given by

$$T_l = C_{\max} (1 - (\sum_{l=1}^{L} \rho_l + P_B)) \tag{7}$$

For blocking systems, T_l can be defined as

$$T_l = C_{\max} \left(1 - (\sum_{l=1}^{L} \rho_l + P_B)\right) - aR, \; aR << C_{\max} \tag{8}$$

If aR is high, the utilization deviates significantly from 100% and the algorithm is simple as only few lower layer codes are used. If aR is low, the utilization is close to 100% but the algorithm is complex. The new call is almost guaranteed to be accepted.

2.2. Multihop networks

The multihop wireless network is shown in Fig. 2(b) for a single call flow with rate $2^l R$. Considering total N links in multihop network the details of a particular link (say link i) is shown in Fig. 2(c). Let $2^{l_i} R$ represents the capacity corresponds to maximum rate vacant code in the i^{th} link and B^i is the bandwidth of ith link. The rate scaling factor for ith link is $2^{l_i - l}$. For a real time call with rate $2^l R$, the call is handled only if $2^l R \le 2^{l_i} R$, and $2^l R \le B^i$. On the other hand, for data call with rate $2^l R$, the transmission rate for ith link can be calculated using steps similar to one used for single hop networks replacing B with B^i, $i \in [1,N]$ representing average bandwidth of i^{th} hop/link and $2^{l'} R$ with $2^{l_i} R$ respectively. Additionally, for ith link, define $R_i = \min[2^{l_i} R, B^i]$. If $R_i = 2^{l_i} R$, the transmission rate is $2^{\lfloor \log_2 B^i \rfloor} R$. The overall transmission rate between X and Y is

$$R_O = \min[R_i], i \in [1, N] \tag{9}$$

In general, if the scaling/suppression of data call at ith link is denoted by A_i, for $1 \le i \le L-1$, following values of A_i are permitted.

$$A_i = \begin{bmatrix} 2^i, \text{data rate increased due to excess resources} \\ 1 \\ 1/2^i, \text{data rate reduced as instantaneous load is high but average load} \\ \quad \text{is less than the maximum capacity of the code tree} \end{bmatrix} \tag{10}$$

For a new call with rate $2^l R$, for the i^{th} link if there are M_i bandwidth variations $B^i_{j_i}, j_i \in [1, M_i]$ due to shared channel, the average bandwidth of i^{th} link is given by

$$B^i = \frac{\sum_{j_i=1}^{M_i} B^i_{j_i}}{M_i}\ ;\ 1 \le i \le N \tag{11}$$

If for i^{th} link, $G^i_{j_i}, j_i \in [1, M_i]$ represents amount of data transmitted for j_i^{th} bandwidth variation, the total transmitted data G^i at the i^{th} link is given by

$$G^i = \sum_{j_i=1}^{M_i} B^i_{j_i}\ ;\ 1 \le i \le N \tag{12}$$

The propagation time for i^{th} link with length d_i, is defined as $t_{p_i} = d_i / v$ where v is the speed of light. If the total data size is D bits, the initial call duration for the single hop system is

$$t_d = D/2^l R\ . \tag{13}$$

Ignoring the propagation time (which is generally very small compared to transmission time), the new call duration is

$$t'_d = t_d \times \left(2^l R / 2^{\lfloor \log_2 B \rfloor}\right). \tag{14}$$

For N link multihop communication, the new call duration (ignoring propagation time and data storage time) is

$$t'_d = \left(t_d \times 2^l R/N\right) \times \left(\frac{1}{2^{\lfloor \log_2 B^1 \rfloor}} + \frac{1}{2^{\lfloor \log_2 B^2 \rfloor}} + \ldots\ldots \frac{1}{2^{\lfloor \log_2 B^N \rfloor}} \right)$$
(15)

4. Proposed Design

Consider an L layer OVSF code tree. If $C_{l,n}$ represents a the code in layer l with branch number n, the code tree capacity assigned to a new call $2^l R$ can have two variants, (*a*) Guaranteed capacity: minimum vacant code capacity required to handle real time call (and is equal to $2^l R$. If the code tree does not have guaranteed capacity, the call is rejected., (*b*) Available capacity: capacity of the vacant code closest to the root code (the root code exists in layer L). For data call $2^l R$ (in layer l-1), the available capacity can take values from $2^{l'} R$ ($0 \le l' < l-1$ for rate suppression and $l \le l' < L-1$ for rate scaling), and if the call duration at call arrival is t, the updated call duration due to rate variation flexibility is given by

$$(2^l / 2^{l'}) \times t \tag{16}$$

For a code $C_{l,n}$, let code index $I_{l,n}$ defines the status of the code. The code indices can take values as follows, (*i*) $I_{l,n} = 0$, if code $C_{l,n}$ is free, (*ii*) $I_{l,n} = 1$, if code $C_{l,n}$ is blocked, (*iii*) $I_{l,n} = 2$, if code $C_{l,n}$ is used for a call with guaranteed capacity $2^l R$ and available capacity $2^l R$, (*iv*) $I_{l,n} = 2 + (l' - l)$, if the code $C_{l,n}$ is used for call with guaranteed capacity $2^l R$ and available capacity $2^{l'} R, l' > l$, and (*v*) $I_{l,n} = (l' - l)$ (negative number), if the vacant code is used from layer $l', l' < l$, and it happens when the code tree does not have vacant code in layer l due to high value of the instantaneous traffic load but the average traffic load allows the use of vacant code in layer l'. For a new $2^l R$ call (data or real time), the flexible data rate algorithm works as follows.

Case 1: Enough capacity available with highest capacity vacant code present in layer $l' \mid l' = \max(l''), l'' \in [l, L]$.

If the call is of data type, the vacant code in layer l', say $C_{l',n_{l'}}$ is assigned. The call rate is intentionally increased by a factor $2^{l'-l}$ and subsequently the call duration is decreased by a factor $2^{l'-l}$. If there are multiple options for vacant code, the code with least code index value $I_{l',n_{l'}}$ is the candidate code to handle the call. For real time call the vacant code with capacity $2^{l}R$ is assigned.

Case 2: Enough capacity is not available
Data call arrival: Check the code indices for all busy codes starting from layer *l*+1. If for a layer $l', l' > l$, there is at least one busy code with code index greater than $2+(l'-l)$ then the call can be handled by reducing the rate of previous ongoing call by shifting this call to lower layer code. More specifically, if there are $N_{l'}$ number of such busy codes in layer l', pick the code $C_{l',n_{l'}}$ with largest code index value $I_{l',n_{l'}}$. Considering this code index value *I*, the code is currently handling a call with rate $2^{l'-I-2}R$, and reducing this $2^{l'-l}$ times the reduced capacity can be utilized by incoming call $2^{l}R$. The code used to handle new $2^{l}R$ call is $C_{l,2^{l'-l}\times n_{l'}-2^{l'-l}+1}$ and the code $C_{l'-I-2,2^{l'-l}\times n_{l'}-2^{l'-l}+1}$ for call $2^{l'-I-2}R$. Therefore the new call is handled applying code shifts (reassignments).
Real time call arrival: For the identified code $C_{l',n_{l'}}$ with largest code index value $I_{l',n_{l'}}$, use one child with capacity $2^{l}R$ (in layer $l-1$) for incoming call and one of the other children with capacity $2^{l}R$ for previous ongoing call. The capacity $2^{l'-1}-2^{l+1}R$ can be used for future calls.

Case 3: Case 1 and 2 fails
Data call arrival: If both of the above cases do not provide vacant code, reassignments are required but the reassignments can be applied only if the total available capacity (sum of capacities of vacant codes in all layers) is more than the capacity required by incoming call. The algorithm identifies layer $l' \mid l' = \max(l''), l'' \in [2,l]$ for which at least one busy code with code index greater than 2 exists. For this layer list all the busy codes along with their code indices. Pick the code $C_{l',n_{l'}}$ with highest code index *I*. This busy code was initially handling an ongoing call with guaranteed rate $2^{l'-(I-2)}R$. The algorithm reduces the rate of this ongoing call by 2^{I-2} times. The new code used to handle this call is $C_{l'-I-2,2^{I-2}\times n_{l'-I-2}-2^{I-2}+1}$. If *t* is the total call duration had the call being handled fully by code $C_{l',n_{l'}}$ and t_1 is the call elapsed time at the arrival of new call $2^{l}R$, the new call duration for ongoing call is given by

$$t' = t_1 + (t - t_1)\times 2^{I-2} \tag{17}$$

If we take t-t_1=t_2, Equation (17) can be rewritten as

$$t' = t_1 + t_2 \times 2^{I-2} \tag{18}$$

In general, if there are total *N* rate transitions for ongoing call, the total call duration can be expressed as

$$\begin{aligned} t' &= t_1 + t_2\times 2^{I_1-2} + t_3\times 2^{I_2-2} + \ldots\ldots t_N \times 2^{I_{N-1}-2} \\ &= t_1 + \textstyle\sum_{i=1}^{N-1}(t_{i+1}\times 2^{I_i-2}) \end{aligned} \tag{19}$$

Also the average rate of the ongoing call becomes

$$\begin{aligned} R' &= \left[t_1\times 2^{l_i'}R + t_2\times 2^{l_i'-(I_1-2)}R + t_3\times 2^{l_i'-(I_2-2)}R \ldots\ldots\ldots + t_N\times 2^{l_i'-(I_{N-1}-2)}R\right]/t \\ &= \left[t_1\times 2^{l_i'}R + \textstyle\sum_{i=1}^{N-1}(t_{i+1}\times 2^{l_i'-(I_i-2)})R\right]/t \end{aligned} \tag{20}$$

Real time call arrival: The call is handled as per *Case* 3 data call handling algorithm.

Case 4: Case 1 to 3 fails
Data call arrival: The call can be handled using vacant codes in layer 1 to *l*-1 if the average

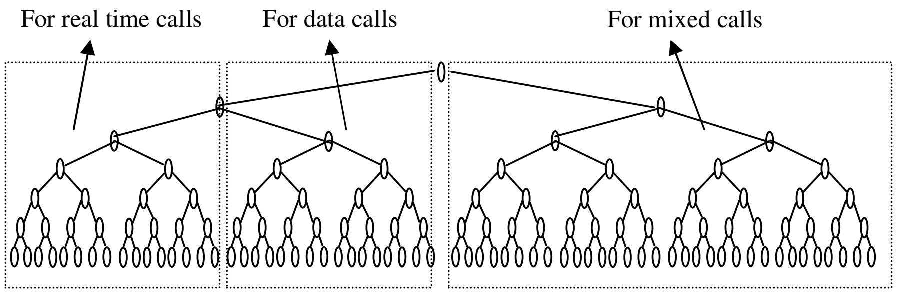


Fig. 3 Code tree division according to call types

traffic load of the system is less than the maximum capacity of code tree. When an ongoing call is completed, the vacant code with the highest rate is identified and the codes which are currently handling the suppressed rate shift the call to the identified vacant code when the identified code has capacity more than the currently used code. The aim is to make code tree capacity utilization close to 100%. The utilization in *Case* 3 can be further increased if we use code reassignments till utilization becomes 100%. Typically there may be many such reassignments to make available capacity more than $2^l R$. If the suppressed call is currently handled by code $C_{l'-I-2,2^{I-2}\times n_{l'-I-2}-2^{I-2}+1}$, the algorithm shifts the call assigned to this code to the appropriate location within the code tree in a layer between l' to $l'-(I-2)$ provided the new code is not $C_{l',n_{l'}}$ or its child.

Real time call arrival: The call is rejected.

While *Case* 3 requires one reassignment per call, *Case* 4 requires large number of reassignments (although greater utilization) even more than traditional DCA. In addition, the code indices are updated at call completion. If any ongoing call occupied by code $C_{l,n}$ is completed, the code indices of the sibling (or children of sibling) has to be updated.

The design discussed so far does not divide code tree according to the type of traffic. Although this increases throughput but complexity and cost of the design may be inappropriate. In another approach, the code tree can have different portions for real time calls, data calls and mixed (a connection where rate may be real or data type) calls. The code tree utilization may not be 100% although the design is simple and cost effective. The mechanism divide call rates into three categories (*a*) pure real time rate (*b*) data call (c) mixed rate as illustrated in Fig. 3. Consequently, the code tree is also divided into three portions to treat these three categories differently. The data portion of the code tree can always be fully utilized as for pure data calls description given earlier. The second portion of the code tree may provide wastage and the only way to optimize utilization is to use full dynamic code assignment (DCA) [7] design. But too many reassignments are not preferred for real time calls. The third region is the most complex as far as code allocation is concerned. Let N_r, N_d and N_m represents total real time, data and mixed classes. Also, $N_r^i, i \in [1, N_r]$, $N_d^i, i \in [1, N_d]$ and $N_m^i, i \in [1, N_m]$ represents the i^{th} call in each category. A particular wireless network may have one, two or all three categories of rates. If $C_{\max}^{r,i}$ denotes the maximum capacity reserved for i^{th} pure real time rate, the portion of total code tree capacity used for real time calls is given by

$$C^{r}_{\max} = \sum_{i=1}^{N_r} C^{r,i}_{\max} \tag{21}$$

Similarly if *d* and *m* represents data and mixed calls, we have

$$C^{d}_{\max} = \sum_{i=1}^{N_d} C^{d,i}_{\max} \tag{22}$$

$$C^{m}_{\max} = \sum_{i=1}^{N_m} C^{m,i}_{\max} \tag{23}$$

Handling mixed rates require fairness among data and real time rates so that none of the two is overserved or underserved.

The division of code tree into various regions is made proportional to the amount of calls. The division changes frequently if arrival distribution changes rapidly.

Fairness issue

Due to the use of data calls for real time calls, some specific layer calls may be over used or under used. Also for real time calls, some specific layer may be overserved, which is unfair for other classes. To make the assignment perfectly good, the fairness is required for following calls.

1. Fairness among real time and data calls.
2. Fairness within real time and data calls.
3. Class (layer) fairness.

Let F_l denotes the fairness of code tree capacity which must be available for rate $2^l R$ calls

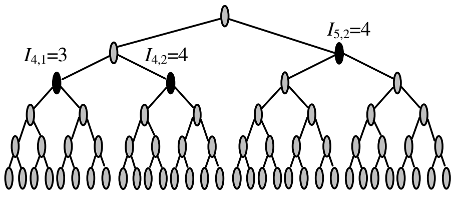


Fig. 4(*a*) A 6 layer OVSF code tree

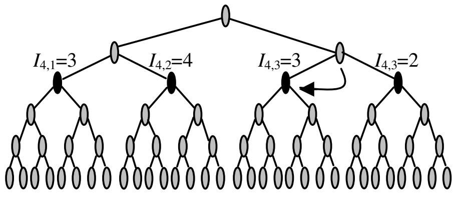


Fig. 4(*b*) 8R call handling

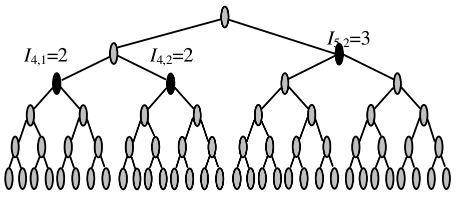


Fig. 4(*c*) A 6 layer OVSF code tree

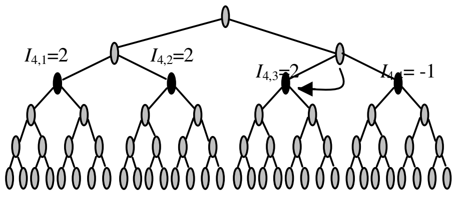


Fig. 4(*d*) 16R call handling

(real time or data)

The value of F_l depends upon the lth class average arrival rate λ_l . Ideally, F_l is defined as

$$F_l = (\lambda_l / \sum_{l=1}^{L} \lambda_l) \times C_{\max} \tag{24}$$

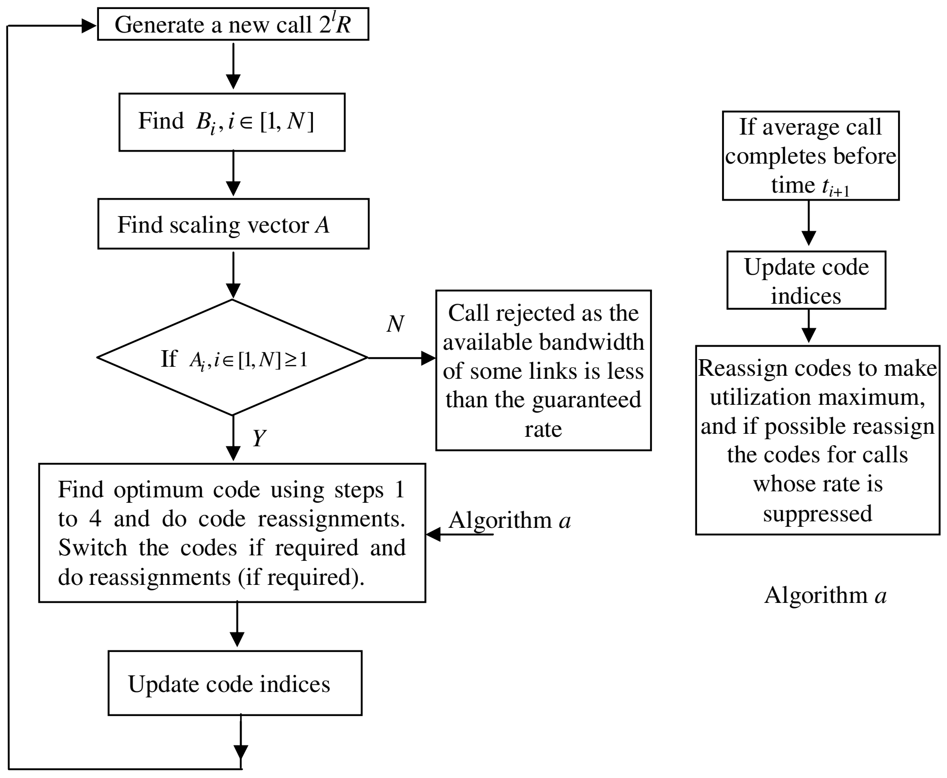

Generate a new call $2^l R$
Find $B_i, i \in [1, N]$
Find scaling vector A
If $A_i, i \in [1, N] \geq 1$
N
Y
Call rejected as the available bandwidth of some links is less than the guaranteed rate
Find optimum code using steps 1 to 4 and do code reassignments. Switch the codes if required and do reassignments (if required).
Algorithm a
Update code indices
If average call completes before time t_{i+1}
Update code indices
Reassign codes to make utilization maximum, and if possible reassign the codes for calls whose rate is suppressed
Algorithm a

Fig. 5 Flow chart of simulation model for real time calls

Practically the fairness index $F_l^{'}$ can be chosen as

$$F_l^{'} = aF_l, \quad 0 \leq a \leq 1 \tag{25}$$

The code assignments and reassignments are explained in Fig. 4 using a six layer code tree. In Fig. 4(*a*), the tree is fully utilized by three ongoing calls occupying codes $C_{5,2}$, $C_{4,1}$ and $C_{4,2}$ with code indices $I_{5,2}$=4, $I_{4,1}$=3, and $I_{4,2}$=4 respectively. If a new call of rate 8*R* arrives, there is no vacant code available, and the algorithm searches for busy code in layer $l \geq \log_2(8) = 3$ with highest code index value greater then 2. The code $I_{5,2}$ in the highest layer has an index equal to 4, and the ongoing call $C_{5,2}$ is shifted to $C_{4,3}$ making $I_{4,3}$ =3 and $C_{4,4}$ vacant. The call 8*R* can now be handled by $C_{4,3}$ (with $I_{4,3}$=2) as shown in Fig. 4(*b*). If the current status is as in Fig 4(*c*), and a new call of 16*R* arrives, there is no vacant code with capacity $\geq$16*R*. The call handled by $I_{5,2}$ will be handled by $I_{4,3}$. If the instanataneous traffic load is high with average traffic load less then 8*R*, the call can be handled. The call currently using code $C_{5,2}$ is shifted to $C_{4,3}$. The call rate is intentionally suppressed to 8*R*. The code $C_{4,4}$ can now be utilized for the incoming 16*R* call as shown in Fig. 4(*d*).

5. SIMULATION MODEL AND RESULTS

The flowchart of simulation model for real time calls is given in Fig. 5, and for data calls in Fig. 6. The simulation parameters used to compare proposed design with other existing designs are listed as follows.

- There are 5 classes of calls with rates *R*, 2*R*, 4*R*, 8*R*, 16*R*.

- The maximum capacity of the tree is 128*R*, with *R* equal to 7.5*kbps*, which require 8 layers in the code tree. The 128*R* code tree capacity is divided into three portions one each for real time, data and mixed rates with capacity 32*R*, 32*R* and 64*R* respectively.

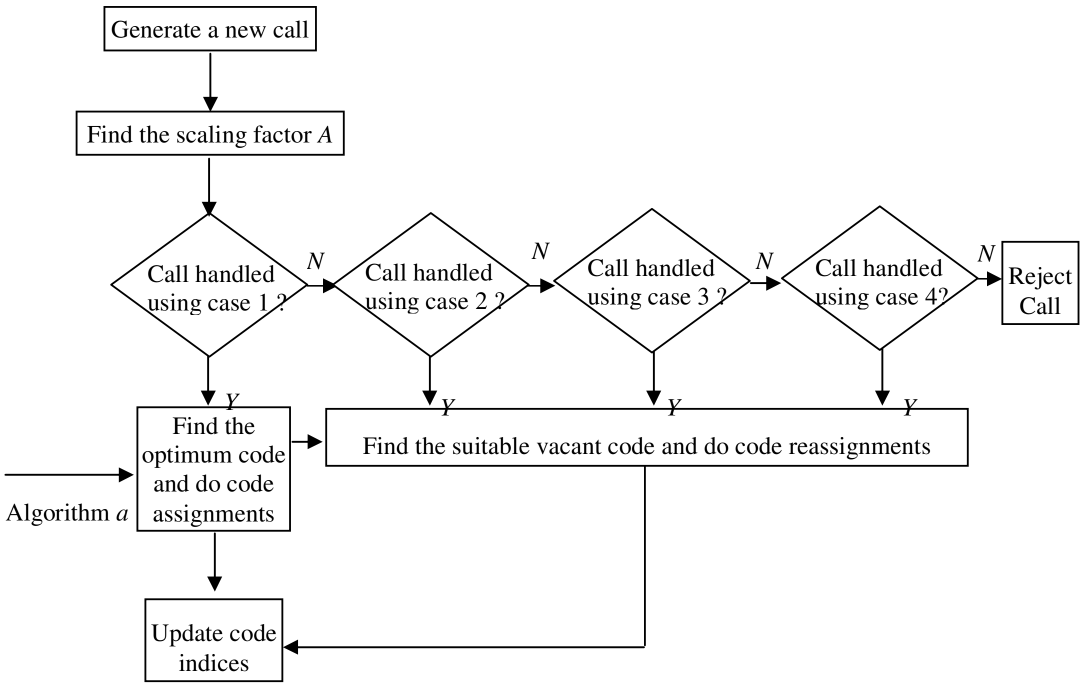
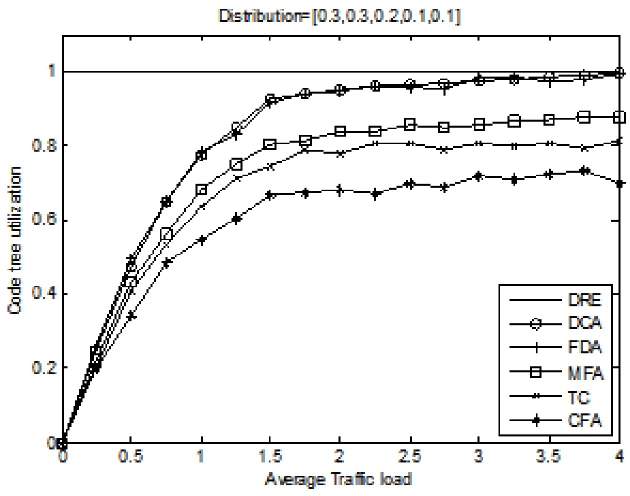
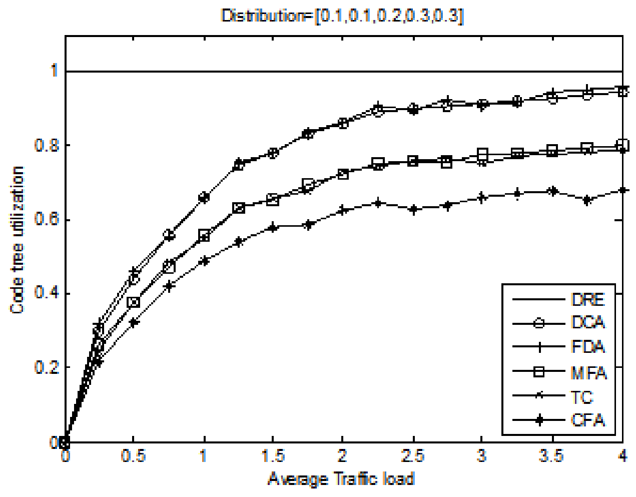

Generate a new call
Find the scaling factor *A*
Call handled using case 1 ?
Call handled using case 2 ?
Call handled using case 3 ?
Call handled using case 4?
Reject Call
N
N
N
N
Y
Y
Y
Y
Find the optimum code and do code assignments
Find the suitable vacant code and do code reassignments
Algorithm *a*
Update code indices

Fig. 6 Flow chart of simulation model for data calls

- Arrival rate λ is Poisson distributed with mean value 0-4 calls/minute.
- Call duration is exponentially distributed with mean value $1/\mu$ of 3 minutes.
- Simulation is done for 1000 calls and result is average of 10 simulations.



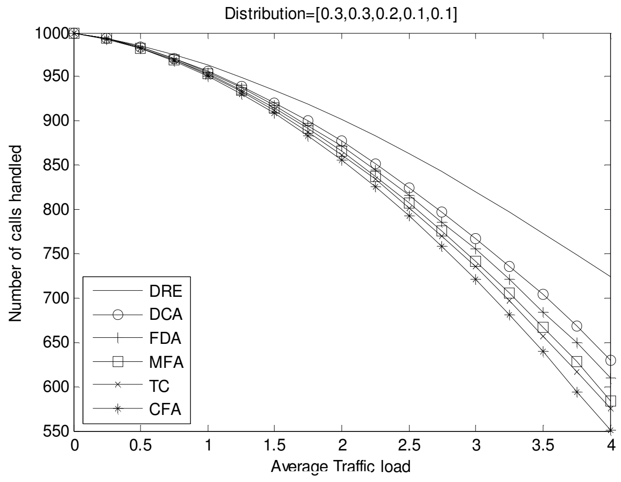
(*a*)



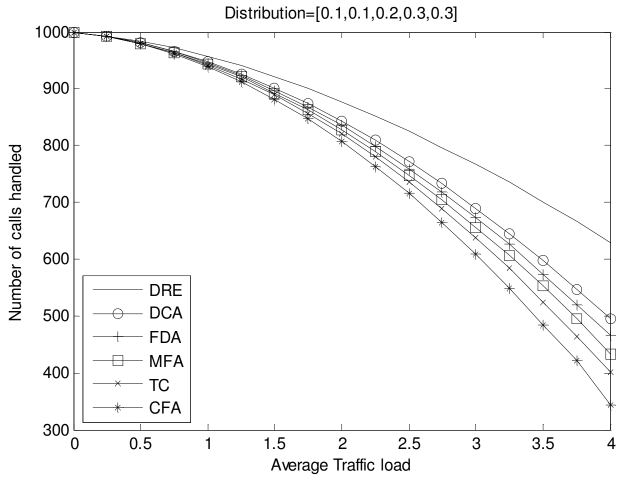
(*b*)

Fig.7 Comparison of code tree utilization for distributions (*a*) [0.3,0.3,0.2,0.1,0.1], (*b*) [0.1,0.1,0.2,0.3,0.3]

Let λ_i, $i \in [1,5]$ is the arrival rate and μ_i is service rate for i^{th} class calls. Define $\rho_i = \lambda_i / \mu_i$ as traffic load of the i^{th} class calls. The average arrival rate and average traffic load becomes $\lambda = \sum_{i=1}^{5} \lambda_i$ and $\rho = \sum_{i=1}^{5} (\lambda_i / \mu_i)$ respectively. In our simulation, we consider average call duration of all the classes equal, i.e. $1/\mu = 1/\mu_i$. Define $[p_1,p_2,p_3,p_4,p_5]$ as probability distribution vector where p_i is the arrival probability for i^{th} class calls on average. Two distribution scenarios are investigated, (*a*) [0.3,0.3,0.2,0.1,0.1] where lower rate calls dominate, (*b*) [0.1,0.1,0.2,0.3,0.3] where higher rate calls dominate. The performance of the proposed data rate expansion (DRE) design is compared with the crowded first assignment CFA[5], dynamic code assignment (DCA) [6], fast dynamic assignment FDA [9], time code design TC[13], and maximally flexible assignment (MFA) [14] designs discussed earlier. The code tree utilization comparison is shown in Fig. 7 for both distributions. The code tree utilization in the proposed design is always 1 because there is flexibility of rate expansion or suppression for data calls. The code tree utilization in dynamic assignment designs DCA and FDA is also close to 1 at higher loads as the code reassignments can make the assignment as compact as possible. The use of DCA requires significant reassignment overhead, which limits their use. The utilization for other designs deviates significantly from 1. The MFA, TC, and CFA designs suffer in utilization because there is no differentiation in real time and non real time calls. Further CFA has least code tree utilization compared to all other methods. The tree utilization further reduces for distribution favouring higher rate calls. This is because the low rate codes are scattered in the code tree, and the parents of these codes in higher layers are blocked. The total number of calls handled by various designs is also compared for two distributions in Fig. 8. The total calls are assumed to be 1000. As expected, the proposed data rate expansion (DRE) design handles highest number of calls for both distributions. The calls handled in MFA, TC and CFA designs are significantly less as there is no provision for rate scaling. Also, the vacant capacity is scattered in the code tree and the calls are rejected due to the problem of external fragmentation. The number of calls rejected further reduces when the higher rate calls are dominating. This is the reason why the calls handled in Fig. 8(*b*) are lesser than the calls handled in Fig. 8(*a*). The calls handled in DCA can approach DRE if the reassignments are done frequently and in addition at the arrival and departure of the call.



(*a*)



(*b*)

Fig.8. Number of calls handled for distributions (*a*) [0.3,0.3,0.2,0.1,0.1], (*b*) [0.1,0.1,0.2,0.3,0.3]

6. CONCLUSIONS

In contrast to real time calls, data calls can be better handled because there is a flexibility to increase or decrease their rates according to available codes in the OVSF code tree. In this work, a code assignment design with rate variation flexibility is investigated. Further, the rate of the non real time call can be increased only if the call is stored for sufficient time in the buffer available at BS and MS. The code tree utilization is maximized as capacity of all the codes is fully utilized in the code tree. The rate of data call can be increased or decreased depending upon available codes, instantaneous and average traffic load. Both single hop networks and multiple hop networks can be benefited. In the proposed scheme, the size of buffers increase as the design stores most of the non real time calls before allocating codes. Hence, the complexity and cost factor increases, and work can be done to limit the complexity while compromising some utilization. Also, the fairness issue among various data call classes and for mixed system (containing both real time and data calls) can be investigated.

REFERENCES

[1] Dahimnan E, Gudmundson B, Nilsson M, Skold J. UMTS/IMT-2000 based on wideband CDMA. *IEEE Communication Magazine* 1998; 36 : 70-80.

[2] Adachi F, Sawahashi M, Suda H. Wideband CDMA for next generation mobile communication Systems. *IEEE Communication Magazine* 1998; 36 : 56-69.

[3] Adachi H, Sawahashi M, Okawa K. Tree structured generation of orthogonal spreading codes with different lengths for forward link of DS-CDMA mobile radio. *IEEE Electronics Letters* 1997; 33 : 27-28.

[4] Chao CM, Tseng YC, Wang LC. Reducing Internal and External Fragmentation of OVSF Codes in WCDMA Systems with Multiple Codes. *IEEE Transactions on Wireless Communication* 2005; 4 : 1516-1526.

[5] Tseng YC, Chao CM, Wu SL. Code placement and replacement strategies for wideband CDMA OVSF code tree management. *GLOBECOM* 2001; 1 : 562-566.

[6] Park JS, Lee DC. Enhanced fixed and dynamic code assignment policies for OVSF-CDMA systems. *ICWN* 2003; 620-625.

[7] Minn T, Siu KY. Dynamic Assignment of orthogonal variable spreading factor codes in W-CDMA. *IEEE Journal of Selected Areas in Communications* 1998; 18(8) : 1429-1440.

[8] Park JS, Huang L, Kuo CCJ. Computational Efficient Dynamic Code Assignment Schemes With Call Admission Control (DCA-CAC) For OVSF-CDMA Systems. *IEEE Transactions on Vehicular Technology* 2008; 57(1) : 286-296.

[9] Rouskas AN, Skoutas DN. Management of channelization codes at the forward link of WCDMA. *IEEE Communication Letters* 2005; 9(8) : 679-681.

[10] Wan CS, Shih WK, Chang RC. Fast dynamic code assignment in next generation wireless access networks. *Elsevier Journal of Computer Communication*s 2003; 26 : 634–1643.

[11] Saini DS, Upadhyay M. Multiple rake combiners and performance improvement in WCDMA systems. *IEEE Transactions on Vehicular Technology* 2009; 58(7) : 3361-3370.

[12] Yang Y, Yum TSP. Multicode Multirate Compact Assignment of OVSF codes for QoS Differentiated Terminals. *IEEE Transactions on Vehicular Technology* 2005; 54(6) : 2114-2124.

[13] Cheng ST, Hsieh MT. Design and Analysis of Time-Based Code Allocation Schemes in W-CDMA Systems. *IEEE Transactions on Mobile Computing* 2005; 4(6) : 604- 615.

[14] Yang Y, Yum TSP. Maximally flexible assignment of orthogonal variable spreading factor codes for multirate traffic. *IEEE Transactions on Wireless Communication* 2004; 3(3) : 781–792.

[15] Kam AC, Minn T, Siu KY. Supporting rate guarantee and fair access for bursty Data traffic W-CDMA. *IEEE Journal on Selected Areas Communication* 2001; 19 : 2121-2130.

[16] Rouskas AN, Skoutas DN. OVSF codes assignment and reassignment at the forward link of W-CDMA 3G systems. *PIMRC* 2002; 2404-2408.

[17] Balyan V, Saini DS. Call Elapsed Time and Reduction in Code Blocking for WCDMA Networks**.** *SOFTCOM* 2009; 141-145.

[18] Tsai YR, Lin LC. Quality-Based OVSF code assignment and reassignment strategies for WCDMA Systems. *IEEE Transactions on Vehicular Technology* 2009; 58(2) : 1027-1031.

[19] Fossa Jr CE, Davis IV, Nathaniel J. A dynamic code assignment algorithm for Quality of Service in 3G wireless networks. *WCNC* 2002; 1:1-6.

[20] Chen MX. Efficient integration OVSF code management architecture in UMTS. *Elsevier Journal of Computer Communications* 2008; 31(9) : 3103–3112.

[21] Chao CM, Tseng YC, Wang LC. Dynamic Bandwidth Allocation for Multimedia Traffic with Rate Guarantee and Fair Access in WCDMA Systems. *IEEE Transactions on Mobile Computing* 2005; 4(5) : 420- 429.

[22] Yeh CC, Wang PL. Dynamic Bandwidth Allocation and Scheduling for OVSF-WCDMA Systems. *International Conference on Communications and Mobile Computing* 2010; 3:475-479.

[23] Al-Adwany, MAS, Ayoub, MB. Performance evaluation of nonblocking OVSF codes in WCDMA systems. *MECAP* 2010; 1-6.

[24] Balyan V, Saini DS. An efficient multi code assignment scheme to reduce call establishment delay for WCDMA networks. *PGDC* 2010; 278-283.

Authors

Davinder S Saini was born in Nalagarh, India in January 1976. He received B.E degree in electronics and telecommunication engineering from College of Engineering Osmanabad, India in 1998. He received M.Tech degree in communication systems from Indian Institute of Technology (IIT) Roorkee, India in 2001. He received PhD degree in electronics and communication from Jaypee University of Information Technology Waknaghat, India in 2008. He is with Jaypee University of Information Technology Waknaghat since june 2002. Currently he is Associate Professor in electronics and communication department. His research areas include Channelization (OVSF) codes and optimization in WCDMA, routing algorithms and security issues in MANETs.

Neeru Sharma was born in Ludhiana, India in November 1979. She received B.E. degree in electronics and telecommunication engineering from College of Engineering, Badnera, Amravati, India in 2001. She received M.E degree in Digital Communication from M.B.M. Engineering College, Jodhpur, Rajasthan, India in 2005. She is with Jaypee University of Information Technology Waknaghat since August 2007. Currently she is Senior lecturer in electronics and communication engineering department. Her research areas include Channelization (OVSF) codes and optimization in WCDMA.