SERGIO LUIZ NORONHA FRAIHA

**UMA ANÁLISE DO PROBLEMA DA
ALOCÇÃO DOS CANAIS DE TRÁFEGO EM
SISTEMAS MÓVEIS CELULARES**

Dissertação de Mestrado apresentada ao
departamento de Engenharia Elétrica da
PUC/Rio, como parte dos requisitos para a
obtenção do título de Mestre em Ciências
em Engenharia Elétrica.

Orientador: José Roberto Boisson de Marca

Centro de Estudos em Telecomunicações
Departamento de Engenharia Elétrica
Pontifícia Universidade Católica do Rio de Janeiro

Rio de Janeiro, setembro de 1996.

*Aos meus pais
Habib e Suely*

AGRADECIMENTOS

Ao professor Boisson pela orientação competente e confiança em mim depositada.

À minha esposa Silvia pelo suporte afetivo e por acreditar na importância deste trabalho.

A todos os meus familiares, em especial a meus pais e meus irmãos Salim e Rogerio, por constantemente tornarem especiais meus momentos extra-classe.

Ao JoPaRiNe pela amizade e constante apoio espiritual.

Aos professores do CETUC, em especial a João Celio, Raimundo, José Mauro e Glaucio, pelos vários ensinamentos ao longo do curso.

Ao amigo Fabricio pelo auxílio na parte inicial das simulações.

Aos funcionários do CETUC e do DEE pelo apoio técnico.

À CAPES-PICD pelo auxílio financeiro.

Aos colegas da Ericsson pelo convívio agradável neste último ano e meio.

Aos colegas do CETUC, em especial a Gloria, Alexandra, Guilherme, o quadrilha de Marcelos (Pinho, Augusto, Bacarini e Leister), a dupla de gaúchos (Alexandre e Prola), Adela, Ana Lucia, Javier e a dupla da Embratel (Luiz Sergio e Geraldo), pelo companheirismo e ambiente sempre agradável.

RESUMO

Este trabalho aborda o problema da alocação dos canais de tráfego em sistemas móveis celulares. A utilização de métodos de alocação mais eficientes constitui uma das principais alternativas para a melhoria do desempenho de sistemas congestionados. Além disto, a constante redução do tamanho das células exige a aplicação de métodos mais flexíveis no que diz respeito ao pré-planejamento das frequências. Alguns dos inúmeros métodos já propostos são apresentados, e dois deles - *CBWL* e Segregação de Canais - são simulados em um sistema *FDMA* micro-celular uni-dimensional. É feita uma comparação do desempenho oferecido por estes dois métodos, e os resultados obtidos indicam a necessidade da utilização de alguma estratégia de prioridade para o atendimento das solicitações de *handoff*.

ABSTRACT

In this work the problem of traffic channel assignment in mobile cellular systems is addressed. The use of more efficient assignment methods represents one of the main alternatives to enhance the performance of congested systems. Moreover, the ceaseless reduction of cell sizes requires the adoption of more flexible methods with regard to the frequency preplanning. Some of the large number of methods already proposed are presented, and two of them - CBWL and Channel Segregation - are simulated in a FDMA 1-dimensional microcellular system. The performance offered by these two methods is compared, and the results indicate the need of using some priority strategy for attending handoff requests.

SUMÁRIO

LISTA DE ABREVIATURAS

LISTA DE NOTAÇÕES GERAIS

LISTA DE FIGURAS

LISTA DE TABELAS

1 - INTRODUÇÃO	01
2 - CONCEITOS IMPORTANTES	06
2.1 - CLASSIFICAÇÃO DOS ALGORITMOS	06
2.2 - MODELO DE PROPAGAÇÃO	07
2.3 - MODELO DE TRÁFEGO	08
2.4 - TOPOLOGIA DO SISTEMA	09
2.5 - MEDIDAS DE DESEMPENHO	11
3 - EXEMPLOS DE ALGORITMOS DE ALOCAÇÃO	13
3.1 - ESQUEMAS ADAPTÁVEIS ÀS VARIAÇÕES DO TRÁFEGO	13
3.1.1 - <i>SB</i> (EMPRÉSTIMO SIMPLES)	13
3.1.2 - <i>HA</i> (ALOCAÇÃO HÍBRIDA)	13
3.1.3 - <i>BCO</i> (EMPRÉSTIMO COM ORDENAÇÃO DE CANAIS)	13
3.1.4 - <i>BDCL</i> (EMPRÉSTIMO DE CANAIS COM TRAVAMENTOS DIRECIONAIS)	14
3.1.5 - <i>CBWL</i> (EMPRÉSTIMO DE CANAIS SEM TRAVAMENTOS)	15
3.2 - ESQUEMAS ADAPTÁVEIS À POTÊNCIA DO SINAL RECEBIDO	19
3.2.1 - PARTICIONAMENTO	21
3.3 - ESQUEMAS ADAPTÁVEIS À INTERFERÊNCIA	22
3.3.1 - CONTROLE CENTRALIZADO	22
3.3.1.1 - ALVO	23
3.3.1.2 - MASCARAMENTO	24

3.3.2 - CONTROLE SEMI-DISTRIBUÍDO	26
3.3.2.1 - <i>LODA</i> (ALOCAÇÃO DINÂMICA LOCALMENTE OTIMIZADA)	26
3.3.3 - CONTROLE DISTRIBUÍDO	27
3.3.3.1 - SEGREGAÇÃO DE CANAIS	27
3.4 - ESQUEMAS COMPLEMENTARES	29
3.4.1 - DIMENSIONAMENTO DO SISTEMA	29
3.4.1.1 - <i>SP1</i> (CASO DE CÉLULA ÚNICA NÚMERO 1)	31
3.4.1.2 - <i>SP2</i> (CASO DE CÉLULA ÚNICA NÚMERO 2)	32
3.4.1.3 - <i>MP</i> (CASO DE MULTI-CÉLULAS)	33
3.4.2 - ALOCAÇÃO DE ERB	34
3.4.2.1 - RETENTATIVA DIRECIONADA	35
3.5 - RESULTADOS FORNECIDOS	36
3.5.1 - <i>BCO</i> , <i>BDCL</i> & <i>LODA</i>	36
3.5.2 - <i>CBWL</i>	36
3.5.3 - PARTICIONAMENTO	39
3.5.4 - ALVO, MASCARAMENTO & RETENTATIVA DIRECIONADA	40
3.5.5 - SEGREGAÇÃO DE CANAIS	41
3.5.6 - <i>SP1</i> , <i>SP2</i> & <i>MP</i>	43
 4 - SIMULAÇÕES	 44
4.1 - ALGORITMOS ESCOLHIDOS	44
4.2 - AMBIENTE OPERACIONAL	44
4.3 - CARACTERÍSTICAS DO SISTEMA	45
4.4 - PRIORIDADE A <i>HANDOFF</i>	48
4.5 - <i>CBWL</i>	50
4.5.1 - ADAPTAÇÃO À ESTRUTURA UNI-DIMENSIONAL	50
4.5.2 - <i>HANDOFF</i> INTRA-CELULAR	53
4.5.3 - RESULTADOS OBTIDOS	54
4.6 - SEGREGAÇÃO DE CANAIS	60
4.6.1 - LIMAR PARA <i>HANDOFFS</i>	60
4.6.2 - RESULTADOS OBTIDOS	62

5 - CONCLUSÕES E SUGESTÕES	68
5.1 - CONCLUSÕES	68
5.2 - SUGESTÕES	69

REFERÊNCIAS BIBLIOGRÁFICAS	71
---	-----------

APÊNDICE A - O MELHOR *CBWL* PARA $N = 2$

APÊNDICE B - O MELHOR *CS* EQUIVALENTE A $N = 3$

LISTA DE ABREVIATURAS

AMPS	-	Serviço Avançado de Telefonia Móvel (<i>Advanced Mobile Phone Service</i>)
BCO	-	Empréstimo com Ordenação de Canais (<i>Borrowing with Channel-Ordering</i>)
BDCL	-	Empréstimo de Canais com Travamentos Direcionais (<i>Borrowing with Directional Channel-Locking</i>)
CBWL	-	Empréstimo de Canais sem Travamentos (<i>Channel Borrowing Without Locking</i>)
CBWL/CR	-	CBWL com Rearranjo de Canais (<i>CBWL with Channel Rearrangement</i>)
CBWL/NR	-	CBWL sem Rearranjo de Canais (<i>CBWL without Channel Rearrangement</i>)
CPS	-	Esquema de Prioridade com Corte (<i>Cutoff Priority Scheme</i>)
CS	-	Segregação de Canais (<i>Channel Segregation</i>)
DCA	-	Alocação Dinâmica de Canais (<i>Dynamic Channel Assignment</i>)
ERB	-	Estação Rádio Base
FCA	-	Alocação Fixa de Canais (<i>Fixed Channel Assignment</i>)
FDMA	-	Múltiplo Acesso por Divisão de Frequência (<i>Frequency Division Multiple Access</i>)
GOS	-	Grau de Serviço (<i>Grade Of Service</i>)
HA	-	Alocação Híbrida (<i>Hybrid Assignment</i>)

LODA	-	Alocação Dinâmica Localmente Otimizada (<i>Locally Optimized Dynamic Assignment</i>)
MP	-	Caso de Multi-células (<i>Multi-cell Problem</i>)
PCS	-	Sistema de Comunicações Pessoais (<i>Personal Communication System</i>)
SB	-	Empréstimo Simples (<i>Simple Borrowing</i>)
SIR	-	Razão Sinal-Interferência (<i>Signal-to-Interference Ratio</i>)
SP1	-	Caso de Célula Única número 1 (<i>Single cell Problem number 1</i>)
SP2	-	Caso de Célula Única número 2 (<i>Single cell Problem number 2</i>)
TDMA	-	Múltiplo Acesso por Divisão de Tempo (<i>Time Division Multiple Access</i>)

LISTA DE NOTAÇÕES GERAIS

α	-	constante de atenuação na propagação dos sinais
C	-	número total de canais no sistema
D	-	distância de reuso
N	-	tamanho do <i>cluster</i>
P_B	-	probabilidade de bloqueio de novas chamadas
P_F	-	probabilidade de falha no <i>handoff</i>
P_I	-	probabilidade de interferência
R	-	raio da célula
S	-	número total de canais pré-alocados a cada célula
SIR_B	-	limiar de <i>SIR</i> para o bloqueio de novas chamadas
SIR_F	-	limiar de <i>SIR</i> para a tentativa de <i>handoff</i>

LISTA DE FIGURAS

1.1 - Estrutura celular com $N = 7$	03
3.1 - O problema do travamento de canais	14
3.2 - Esquema <i>CBWL</i> bi-dimensional com $N = 7$	16
3.3 - Particionamento do reuso com K sub-células	20
3.4 - Mecanismo de verificação das interferências	22
3.5 - Fluxograma do Segregação de Canais em cada ERB	28
4.1 - Pior caso de interferência para $N = 3$ usando <i>FCA</i>	47
4.2 - Pior caso de interferência para $N = 2$ usando <i>FCA</i>	47
4.3 - Pior caso de interferência em canal tomado emprestado ($N = 3$)	50
4.4 - Pior caso de interferência em canal regular ($N = 3$)	51
4.5 - Pior caso de interferência em canal tomado emprestado ($N = 2$)	52
4.6 - Pior caso de interferência em canal regular ($N = 2$)	52

LISTA DE TABELAS

3.1	- Valores de p correspondentes a diversos valores de N	18
3.2	- Resultados do <i>CBWL/NR</i> para tráfego uniforme	37
3.3	- Resultados do <i>CBWL/CR</i> para tráfego uniforme	37
3.4	- Resultados do <i>CBWL/CR</i> para a célula <i>hot spot</i>	38
3.5	- Resultados do <i>CBWL/CR</i> para as células normais	38
3.6	- Resultados do Segregação de Canais para $C = 36$	42
3.7	- Resultados do Segregação de Canais para $C = 54$	43
4.1	- Resultados do <i>CBWL</i> básico para $N = 3$	55
4.2	- Resultados do <i>CBWL</i> usando prioridade com corte para $N = 3$	56
4.3	- Resultados do <i>CBWL</i> com prioridade a <i>handoff</i> para $N = 3$	57
4.4	- Resultados do <i>CBWL</i> básico para $N = 2$	58
4.5	- Resultados do <i>CBWL</i> usando prioridade com corte para $N = 2$	58
4.6	- Resultados do <i>CBWL</i> com prioridade a <i>handoff</i> para $N = 2$	59
4.7	- Valores de d_{inter} e d_{intra} para $N = 2$ e $N = 3$	61
4.8	- Resultados do <i>CS</i> original para $N = 3$	62
4.9	- Resultados do <i>CS</i> com <i>sliding window</i> para $N = 3$	64
4.10	- Resultados do <i>CS</i> com prioridade a <i>handoff</i> para $N = 3$	64
4.11	- Resultados dos três algoritmos para $N = 3$	65
4.12	- Resultados iniciais do <i>CS</i> para $N = 2$	65
4.13	- Resultados do <i>CS</i> com prioridade a <i>handoff</i> para $N = 2$	66
4.14	- Resultados dos três algoritmos para $N = 2$	67

1 - INTRODUÇÃO

Este trabalho visa estudar e discutir um dos assuntos mais importantes no que diz respeito ao projeto de um Sistema Móvel Celular: a escolha do método de alocação dos canais de tráfego (isto é, canais de voz) a ser utilizado neste tipo de sistema.

Começemos pensando nesta questão sob o ponto de vista prático: quando o usuário de um sistema móvel deseja iniciar uma chamada telefônica, o sucesso da operação depende da disponibilidade de um meio de comunicação a este usuário, sendo que o meio neste caso constitui-se em um canal de frequência. O grande problema aqui existente é que a quantidade total de canais definidos tem sua limitação, e é grande o número de usuários a ser atendido.

Suponhamos agora que tivessem sido definidos C canais e que o sistema tivesse apenas uma Estação Rádio Base (ERB). Como cada usuário comunicante utiliza um canal, no máximo C usuários poderiam usar o sistema simultaneamente. Foi diante deste problema que imaginou-se dividir a área a ser coberta por um sistema móvel em várias sub-áreas (conhecidas como células, daí o nome sistema celular), onde cada sub-área viria a ter a sua própria ERB. Graças a este novo conceito tornou-se possível o chamado reuso de canal, que consiste na utilização de um mesmo canal por duas ou mais chamadas simultaneamente.

Obviamente esta nova possibilidade levantou questões acerca da interferência: até quantos usuários podem usar o mesmo canal simultaneamente? Quão distante um usuário deve estar do outro para que os dois possam usar o mesmo canal? Desta forma, sistemas celulares são ditos “limitados pela interferência”, em contraste aos sistemas de comunicações típicos, os quais dizemos ser “limitados por ruído”.

Dentro deste contexto, sistemas celulares utilizam o conceito de Razão Sinal-Interferência (SIR) para que seja definida a possibilidade ou não do reuso

de um canal. Assim, tendo sido definido um valor (em dB) de limiar para esta razão, existe em correspondência uma distância mínima (a distância de reuso - D) entre dois usuários para que possam utilizar simultaneamente o mesmo canal. Desde que o nível de SIR dos usuários de um mesmo canal permaneça acima do limiar pré-estabelecido não existe um limite teórico para o número máximo de usuários simultâneos neste canal.

Durante a chamada é possível que o nível de SIR para um determinado usuário diminua ao ponto de ser necessário que ele tente trocar de canal. Existem duas causas principais para que este tipo de evento aconteça. A primeira é a própria movimentação do usuário, que pode fazer com que ele se afaste muito da ERB que o controla.

A segunda causa é a possibilidade de que a entrada de um novo usuário naquele canal provoque níveis de interferência acima do tolerável. No jargão celular o evento troca de canal durante a chamada é conhecido como *handover* ou *handoff*. Maiores detalhes a respeito dos conceitos básicos de sistemas celulares podem ser encontrados em [1]-[3].

Tendo sido dadas algumas idéias do que vem a ser um sistema celular podemos agora passar a tratar do assunto deste trabalho. Vamos voltar a supor que um usuário deseja iniciar uma chamada. Suponhamos ainda que neste instante dois ou mais canais estejam disponíveis a este usuário. A pergunta é: será que a nível de sistema como um todo não há diferença entre alocar à chamada o canal x ou o canal y ou ...? A resposta: se este usuário estivesse “sozinho” no sistema a questão seria irrelevante, no entanto, sabe-se que, na prática, é grande o número de chamadas simultâneas e, assim, a escolha do canal específico passa a ter grande importância.

Nos primeiros sistemas celulares (ditos “da primeira geração”), ainda usados no Brasil e em outras partes do mundo, o projeto do esquema de alocação dos canais foi concebido de maneira tão simples quanto as demais idéias de então. Nestes sistemas, a área como um todo é dividida em grupos

(*clusters*) formados por N células, e a cada *cluster* ficam disponíveis todos os C canais do sistema. De acordo com aquele projeto os C canais são então subdivididos em N grupos, ficando assim disponíveis $S = C/N$ canais para cada célula do sistema. A figura 1.1 ilustra uma estrutura celular com tamanho de *cluster* N igual a 7 (valor típico).

Figura 1.1 - Estrutura celular com $N = 7$

Nesta situação, quando um usuário está em uma determinada célula ele só pode ter acesso a um dos S canais a ela pré-alocados, de modo que se no momento de uma solicitação todos estes canais estiverem indisponíveis, ocorrerá então o bloqueio ou interrupção da chamada. No que se refere à escolha específica entre dois ou mais canais, possivelmente disponíveis naquele momento, a simplicidade opta por “alocar o primeiro canal livre que for encontrado durante a varredura”, sendo que a ordem da varredura varia de acordo com a implementação do sistema.

Em consequência, o que se verifica na prática é uma utilização ineficiente dos recursos de frequência neste tipo de sistema celular. Somente para exemplificar, pensemos novamente na situação em que um usuário quisesse iniciar uma chamada e encontrasse todos os canais de sua célula

ocupados. Neste momento, mesmo que todos os canais das células vizinhas estivessem disponíveis a chamada seria bloqueada.

Sabendo-se que a definição de maior banda de frequência por parte das organizações que concedem estes recursos representa uma questão delicada, e notando ainda que os recursos existentes são invariavelmente incapazes de atender satisfatoriamente toda a demanda, torna-se constante a busca de soluções alternativas que possibilitem alguma forma de expansão dos sistemas.

Neste contexto então é que surge a idéia do aperfeiçoamento do método de alocação dos canais como uma das mais importantes estratégias para a expansão de sistemas congestionados. Esta e as demais estratégias possíveis são amplamente discutidas em [3] - capítulo 2.

Das várias estratégias de expansão de um sistema celular, uma das mais utilizadas é a diminuição do tamanho das células. Entretanto, na prática, quanto menor o tamanho da célula mais árdua é a tarefa de se fazer um pré-planejamento de frequências que garanta a qualidade do sistema. Este problema, na realidade, vem a ser o principal incentivo para a busca de métodos mais flexíveis de alocação dos canais, que se possível eliminem o pré-planejamento existente no método inicial, o qual é conhecido como Alocação Fixa de Canais (*FCA*).

Devido à constante necessidade do surgimento de novas soluções, e ainda por representar uma área bastante interessante para o campo da pesquisa, tem sido grande a quantidade de métodos propostos em todo o mundo, principalmente nestes últimos dez anos. Os algoritmos variam desde simples esquemas de empréstimo de canais até alguns de implementação atual inviável, que no entanto não deixam de ser interessantes por darem uma idéia de quanto o desempenho do sistema pode vir a ser melhorado.

Nosso trabalho inicial consistiu de uma vasta pesquisa bibliográfica que envolveu dezenas de artigos - todos relacionados a sistemas *FDMA* (Múltiplo Acesso por Divisão de Frequência). Em seguida, selecionamos alguns ([8], [9],

[11], [12], [13] e [14]) destes artigos, procurando, no entanto, preservar diferentes tipos de estratégia.

Dos vários algoritmos descritos nos artigos selecionados, dois (*CBWL* e *Segregação de Canais*) vieram a ser escolhidos para uma análise mais detalhada, e os motivos de tal seleção estão descritos na seção 4.1. Estes dois algoritmos foram então simulados em um sistema *FDMA* micro-celular unidimensional.

O capítulo 2 focaliza alguns conceitos importantes, necessários ao melhor entendimento do restante do trabalho. No capítulo 3 são apresentados os algoritmos de alocação contidos nos artigos escolhidos. Nossas simulações são o assunto do capítulo 4, e o capítulo 5 encerra o trabalho apresentando nossas conclusões e sugestões para trabalhos futuros.

2 - CONCEITOS IMPORTANTES

2.1 - CLASSIFICAÇÃO DOS ALGORITMOS

Neste quesito há uma divergência de idéias entre os autores. Para alguns, algoritmos de Alocação Dinâmica de Canais (*DCA*) são aqueles em que todos os canais são colocados à disposição de todas as células, não havendo assim nenhum pré-planejamento de frequências.

Outros autores no entanto consideram como *DCA* qualquer algoritmo que ofereça algum tipo de flexibilidade na operação - característica inexistente no método inicial (*FCA*). No contexto desta segunda visão, Beck e Panzer propuseram a divisão dos algoritmos *DCA* em três categorias [4]: esquemas adaptáveis às variações do tráfego, esquemas adaptáveis à potência do sinal recebido e esquemas adaptáveis à interferência.

Nos esquemas adaptáveis às variações do tráfego, o número de canais alocados a cada célula depende da quantidade de chamadas em andamento nestas células. Aqui estão enquadrados, por exemplo, os algoritmos que prevêem empréstimos de canais entre as células.

A idéia básica dos esquemas adaptáveis à potência do sinal recebido é de que se um usuário recebe sinal mais forte - normalmente por estar próximo à ERB que o controla - então ele pode tolerar níveis de interferência também mais fortes. Canais usados por este tipo de usuário podem então ser reusados a distâncias menores, aumentando a capacidade do sistema.

Por último, esquemas adaptáveis à interferência são aqueles em que medidas de interferência são utilizadas como dados de entrada para cada novo procedimento de alocação de canal. É claro que existem algoritmos pertencentes a mais de uma destas três categorias.

Neste trabalho será utilizada a classificação de Beck e Panzer. No entanto, os algoritmos que não requerem um pré-planejamento de frequências serão aqui denominados *DCA* “puros”.

2.2 - MODELO DE PROPAGAÇÃO

É através deste modelo que são definidas as influências que o meio de transmissão exerce nos sinais transmitidos, tanto pelos terminais móveis, como pelas ERBs.

Para o caso específico de um sistema celular típico, o meio de transmissão nada mais é do que o espaço livre (com reflexões), o qual pode ser interpretado através de modelos com variados graus de complexidade.

O modelo de propagação mais simples é aquele no qual a perda de potência dos sinais ao longo do meio de transmissão é determinística e dependente tão somente da distância entre o transmissor e o receptor. Neste modelo a Potência do Sinal Recebido (P_R) pode ser definida pela equação:

$$P_R(d) = P_T \cdot d^{-\alpha} \quad (2.1)$$

onde:

- P_T = potência do sinal transmitido
- d = distância entre transmissor e receptor
- α = constante de atenuação, com valor típico entre 2 e 5

Haja vista a grande aleatoriedade existente no meio de transmissão utilizado pelos sistemas celulares, algumas simulações incluem variáveis aleatórias no modelo de propagação, para que se tenha maior grau de realismo nos resultados obtidos.

De acordo com o nosso estudo, quando o pesquisador decide incluir perdas randômicas em seu modelo, ele normalmente opta pela existência do desvanecimento lento devido ao sombreamento (*slow shadow fading*). Este tipo de perda é provocado pela presença de obstáculos (morros, edifícios, árvores...) entre o transmissor e o receptor, e é modelado por uma variável aleatória log-

normal, com valor médio igual a zero e desvio-padrão com valor típico entre 6 e 10 dB.

Um outro tipo de desvanecimento, raramente considerado em simulações, é o rápido causado pelos multi-percursos (*fast multipath fading*). Na prática este desvanecimento ocorre porque o sinal resultante recebido é constituído pela soma de vários sinais - originados do sinal transmitido - que, com o movimento dos terminais, sofreram diferentes graus de dispersão e reflexão ao longo dos percursos, chegando então ao receptor com fases e amplitudes diferentes uns dos outros.

2.3 - MODELO DE TRÁFEGO

É este modelo que caracteriza o comportamento dos usuários e estabelece a quantidade dos recursos (canais de frequência) disponíveis no sistema.

Quando as simulações consideram que todas as células do sistema possuem a mesma carga média de tráfego, dizemos que o tráfego é do tipo uniforme. Sabendo que, na prática, este tipo de uniformidade raramente acontece, algumas simulações definem a existência de *hot spots*, ou seja células com valores médios de carga maiores do que os valores das demais, e isto vem a caracterizar o que chamamos de tráfego não-uniforme.

O comportamento estatístico das chamadas dos usuários invariavelmente é modelado como um Processo de Nascimento e Morte. Neste tipo de processo tanto os intervalos entre as chegadas (“nascimentos”) de chamadas como a duração (“tempo de vida”) destas chamadas constituem variáveis aleatórias independentes e do tipo exponencial.

Costuma-se definir para o tempo de duração das chamadas um valor médio entre 2 e 3 minutos, enquanto que o intervalo entre as chegadas depende da intensidade de tráfego definida no Processo de Poisson utilizado.

2.4 - TOPOLOGIA DO SISTEMA

É bem verdade que sistemas celulares práticos envolvem ambientes tri-dimensionais. Não obstante, quando modelado em simulações, quase sempre considera-se que o sistema é bi, ou mesmo uni-dimensional, de modo a permitir análises menos complexas.

Apesar de irreal, outra consideração amplamente utilizada é a definição do formato hexagonal para as células. Três arranjos de polígonos regulares (triângulos equiláteros, quadrados e hexágonos) são capazes de cobrir um plano sem deixar vazios nem provocar superposições. O consenso em torno do hexágono vem do fato de que menos hexágonos são necessários para cobrir uma área pré-estabelecida, o que o torna economicamente mais interessante - haja visto ser alto o custo de uma ERB [1].

Um ponto importante a ser considerado em simulações é o chamado “efeito de borda”. Tal efeito representa a possível iniquidade de carga de tráfego - e também de interferência - oferecida para as células do sistema, devido ao fato de que as células das extremidades possuem menos células vizinhas do que as células centrais.

Para contornar o efeito de borda utiliza-se normalmente uma de duas estratégias. A primeira é conhecida como *wrap-around* e consiste em considerar que as células de uma extremidade do sistema são vizinhas às células da extremidade oposta. A outra estratégia é simplesmente analisar apenas os resultados obtidos para algumas células pertencentes ao centro do

sistema. Nada impede, no entanto, que estas duas estratégias sejam usadas em conjunto.

Finalmente quando as simulações envolvem algum algoritmo que exige um pré-planejamento de frequências, o parâmetro N (tamanho do *cluster*) possui importância fundamental. Merece ser destacado que, quanto menor for o valor de N , maior será a quantidade de canais disponíveis em cada célula, sendo maior portanto a capacidade do sistema. Por outro lado, quanto maior for N , maior será a distância de reuso dos canais, sendo melhor portanto a qualidade do sistema.

Em sistemas bi-dimensionais nem todos os números naturais podem ser definidos como tamanho de *cluster*, sendo isto uma decorrência das próprias limitações da geometria celular hexagonal. Na teoria, 1, 3, 4, 7, 9, ... são valores permitidos ao passo que 2, 5, 6, 8, 10, ... (além obviamente do 0) não o são. Os valores permitidos para N são determinados pela seguinte equação [1]:

$$N = i^2 + ij + j^2 \quad (2.2)$$

onde:

$$(i \in \mathbf{N}^*) \quad (i \geq j) \quad (j \in \mathbf{N})$$

Em termos práticos a qualidade é definida através de limiares de Razão Sinal-Interferência (SIR), sendo que o limiar (SIR_B) utilizado para a definição do bloqueio de uma nova chamada costuma ser maior do que o limiar (SIR_F) utilizado para a definição da necessidade de uma tentativa de *handoff*. A razão para a diferença entre os valores destes limiares é descrita na seção 4.4.

2.5 - MEDIDAS DE DESEMPENHO

Existem várias formas de se avaliar o desempenho dos algoritmos de alocação. No entanto, quatro são as medidas referidas neste trabalho:

- Probabilidade de Bloqueio de Novas Chamadas (P_B):

Define-se como bloqueio o não-atendimento de uma nova chamada. O bloqueio pode ocorrer devido à inexistência de um canal livre, ou ainda devido à inexistência de um canal com qualidade suficiente. P_B mede o percentual de ocorrências destes eventos com relação ao número acumulado de novas chamadas no sistema.

- Probabilidade de Falha no Handoff (P_F):

Este tipo de falha acontece quando não há canal disponível - ou com qualidade suficiente - para atender a uma tentativa de transferência de chamada em andamento, isto é, uma tentativa de *handoff*. P_F mede o percentual de ocorrências deste tipo de evento com relação ao número acumulado de tentativas de *handoff*.

- Probabilidade de Interferência (P_I):

Uma situação crítica de interferência ocorre quando, durante uma chamada, o nível de *SIR* de um usuário cai abaixo de um certo limiar de qualidade pré-definido. P_I mede o percentual de ocorrências deste tipo de evento com relação ao número acumulado de novas chamadas no sistema.

O limiar de *SIR* que define uma situação crítica de interferência não é necessariamente igual ao limiar SIR_F . No entanto, se esta igualdade vier a ser pré-estabelecida, e se o nível de *SIR* do usuário cair abaixo deste limiar, então haverá uma tentativa de *handoff* para este usuário.

- Taxa de Falha no Enlace (χ):

Esta medida de desempenho é na verdade equivalente à Probabilidade de Bloqueio de Novas Chamadas (P_B), e a designação diferente deve-se ao fato de que χ é utilizada em análises do tipo *snapshot*. Este tipo de análise é descrito no decorrer do próximo capítulo (na seção 3.5.3), o qual apresenta alguns exemplos de algoritmos de alocação de canais.

3 - EXEMPLOS DE ALGORITMOS DE ALOCAÇÃO

A ordem de apresentação dos algoritmos neste capítulo segue a classificação proposta por Beck e Panzer em [4]. Ao final do capítulo são apresentados ainda alguns resultados obtidos por estes algoritmos.

3.1 - ESQUEMAS ADAPTÁVEIS ÀS VARIAÇÕES DO TRÁFEGO

3.1.1 - *SB* (EMPRÉSTIMO SIMPLES)

Usando o algoritmo *SB*, um conjunto de canais nominais é previamente alocado a cada célula e, na ocorrência de uma solicitação, a célula tentará utilizar um destes canais. Se isto não for possível esta célula tentará tomar emprestado um canal nominal de uma de suas células vizinhas. Não sendo possível também o empréstimo a chamada será bloqueada ou interrompida [5].

3.1.2 - *HA* (ALOCAÇÃO HÍBRIDA)

No *HA* o número de canais nominais de cada célula é dividido em dois sub-conjuntos: *A* e *B*. Canais de *A* são usados somente na própria célula, ao passo que os canais de *B* podem ser emprestados às células vizinhas. Definindo-se $\#A$ e $\#B$ como as quantidades de canais pertencentes aos sub-conjuntos *A* e *B*, então a razão $\#A/\#B$ é estabelecida *a priori*, e seu valor ótimo depende da intensidade de tráfego oferecida [6].

3.1.3 - *BCO* (EMPRÉSTIMO COM ORDENAÇÃO DE CANAIS)

No *BCO* não existe uma pré-divisão dos canais de *A* e de *B*. Em cada célula, os canais nominais são ordenados de modo que o primeiro canal tem a maior prioridade de ser alocado à próxima chamada local, enquanto o último canal tem a maior prioridade de ser emprestado às células vizinhas [7]. Isto faz

com que a razão $\#A/\#B$ varie automaticamente conforme a carga de tráfego oferecida.

3.1.4 - BDCL (EMPRÉSTIMO DE CANAIS COM TRAVAMENTOS DIRECIONAIS)

O grande problema comum aos algoritmos das três seções anteriores é o fato de que quando um canal é emprestado ele fica indisponível (travado) não apenas para a célula que o emprestou mas também para as células que estejam dentro da distância de reuso da célula que o tomou emprestado. Considerando $N = 7$, a figura 3.1 ilustra esta situação.

Figura 3.1 - O problema do travamento de canais

Nesta figura $A1, A2, \dots, A5$ são células co-canais, ou seja células que utilizam o mesmo grupo de canais. Suponhamos inicialmente que x é um dos canais deste grupo. Neste caso se x fosse emprestado pela célula $A1$ à célula P , então este canal ficaria travado para $A1, A2$ e $A3$. Estar travado na célula i significa que o canal não pode ser utilizado em i nem ser emprestado às células vizinhas de i .

Neste contexto se acontecesse de a célula Q pedir um canal emprestado à célula $A3$, o canal x não poderia ser utilizado na operação. Esta é sem dúvida uma grande limitação, haja visto que as células P e Q estão separadas pela

distância de reuso D . Além disto, estando x travado para $A3$, este canal também não poderia ser emprestado por $A4$ à célula R , apesar de R também estar suficientemente distante de P .

Visando diminuir os travamentos é proposto em [8] o algoritmo *BDCL*, no qual quando um canal é emprestado ele fica travado apenas para as direções realmente afetadas por este empréstimo.

Referindo-se ainda à figura 3.1, se o canal x fosse emprestado de $A1$ para P , então na célula $A3$ este canal só ficaria travado para as direções 3, 4 e 5, de modo que, se necessário fosse, a célula Q - que está na direção 2 de $A3$ - poderia a princípio usar o canal x . Dizemos “a princípio” porque o uso de x em Q dependeria também da situação deste canal nas células $A4$ e $A5$.

Conclui-se que as penalidades decorrentes de empréstimos são diminuídas, mas continuam a existir. Desta forma o *BDCL* utiliza ainda as estratégias de ordenação - também usada pelo algoritmo *BCO* (seção anterior) - e de realocação de canais, visando a minimização do uso de canais emprestados.

A idéia básica das realocações é que uma chamada só permanecerá usando um canal tomado emprestado se não for liberado nenhum canal nominal que possa atendê-la. Na prática, no entanto, isto aumenta a taxa de ocorrências de *handoff*.

Maiores detalhes a respeito do funcionamento do *BDCL* são encontrados em [8].

3.1.5 - *CBWL* (EMPRÉSTIMO DE CANAIS SEM TRAVAMENTOS)

O principal trunfo do algoritmo *CBWL* está explicitado em seu próprio nome: ele consiste de um esquema de empréstimo de canais que elimina todos os travamentos existentes nos algoritmos descritos nas quatro seções anteriores.

Para que não ocorram os travamentos, os canais tomados emprestados são usados com potência de transmissão reduzida [9]. Devido à redução na

potência, apenas uma parte dos usuários do sistema pode vir a utilizar canais emprestados. Na prática, em cada célula, existirá uma região - com área percentual igual a p - em torno da antena da ERB, dentro da qual o usuário deverá estar situado para que a ele possa ser alocado um canal emprestado.

Usando o *CBWL*, se uma célula possui ao todo n células vizinhas, então os canais pré-alocados a cada célula são sub-divididos em $n+1$ grupos ($0, 1, \dots, n$). Canais do grupo 0 não podem ser emprestados, ao passo que para os demais grupos vale a regra: canais do grupo i só podem ser emprestados para a célula vizinha rotulada na direção i . A figura 3.2 ilustra um esquema *CBWL* no qual cada célula possui seis células vizinhas, e onde o tamanho do *cluster* $N = 7$.

Figura 3.2 - Esquema *CBWL* bi-dimensional com $N = 7$

De acordo com a ilustração os canais da célula A são sub-divididos nos grupos $a_0, a_1, \dots, a_6$. Canais do grupo a_1 só podem ser emprestados para a célula

B , canais do grupo a_2 só podem ser emprestados para a célula C , e assim por diante. Esta mesma sub-divisão de canais é então adotada para a célula A' e para todas as outras células co-canais da célula A . Em seguida procedimentos similares são aplicados a todas as demais células do sistema ($B, B', C, C', \dots$).

Conclui-se que todos os empréstimos são direcionais, o que garante a inexistência de conflitos. Este aspecto, aliado à definição de uma região de área p , permite uma diminuição na troca de informações entre as células, haja visto que a princípio nenhum empréstimo violará os critérios de interferência do sistema.

Para aumentar o desempenho do *CBWL* o artigo [9] propõe a inclusão de duas funcionalidades: Troca de Canais (*Channel Swapping*) e Rearranjo de Canais (*Channel Rearrangement*).

Com o uso de Troca de Canais, quando uma ERB A recebe um pedido de empréstimo e todos os canais do grupo i associado à célula B requerente estão indisponíveis, o empréstimo não é imediatamente negado. Nesta situação a ERB A verifica se algum dos canais deste grupo está sendo usado localmente. Se este for o caso e se houver algum canal disponível em outro grupo j de A então a chamada em andamento no canal do grupo i será realocada para o canal do grupo j , permitindo assim o empréstimo a B do canal liberado no grupo i .

Antes de passarmos à descrição da função Rearranjo de Canais, definiremos como usuário do tipo 1 aquele que pode usar canais emprestados, e usuário do tipo 2 aquele que não pode usar tais canais.

Com o uso de Rearranjo de Canais, quando um usuário do tipo 2 requer um canal e todos os canais de sua célula A estão indisponíveis, a solicitação não é imediatamente negada. Nesta situação a ERB A verifica se algum de seus canais está sendo usado localmente por um usuário do tipo 1. Se este for o caso então a ERB A verifica ainda se alguma de suas células vizinhas pode lhe emprestar um canal. Caso o empréstimo seja possível então a chamada em

andamento do usuário do tipo 1 será realocada para o canal emprestado, e o canal liberado da célula A será alocado ao usuário do tipo 2.

O artigo citado descreve em detalhes a metodologia para o cálculo do valor de p . Esta metodologia é parcialmente descrita na seção 4.5.1 deste trabalho. Definindo-se r como o raio da região de área p , e R como o raio da célula, alguns dos resultados apresentados são os seguintes:

Tabela 3.1 - Valores de p correspondentes a diversos valores de N

N	r/R	p
3	0.27	0.09
4	0.19	0.04
7	0.40	0.19
9	0.43	0.22
12	0.57	0.39
13	0.53	0.36
16	0.55	0.36
19	0.63	0.48
21	0.61	0.45
25	0.64	0.49

Os valores da tabela 3.1 devem ser interpretados da seguinte forma: se o sistema celular utilizar um determinado tamanho de *cluster* N , então a aplicação do algoritmo *CBWL* produzirá em cada célula uma região de área percentual p na qual canais emprestados poderão ser alocados com a garantia de que será mantida a mesma qualidade de serviço que seria oferecida se o algoritmo aplicado fosse o *FCA*.

Merece ser destacado que o algoritmo *CBWL* foi - ao lado do Segregação de Canais (seção 3.3.3.1) - um dos esquemas escolhidos para as simulações deste trabalho (capítulo 4).

Em termos de classificação, na realidade, o esquema *CBWL* pertence também à categoria dos algoritmos adaptáveis à potência do sinal recebido - assunto da próxima seção.

3.2 - ESQUEMAS ADAPTÁVEIS À POTÊNCIA DO SINAL RECEBIDO

Quando as ERBs utilizam antenas omni-direcionais, a maneira mais intuitiva de projetar um esquema adaptável à potência do sinal recebido consiste em dividir cada célula do sistema em uma determinada quantidade de sub-células concêntricas.

Feita a mesma divisão em todas as células, cada conjunto de sub-células de um mesmo tipo passará a ter a sua própria distância de reuso, sendo que a distância de reuso para um conjunto de sub-células centrais será menor do que a distância de reuso para um conjunto de sub-células da extremidade.

Conhecidos todos os conjuntos de sub-células do sistema, será então aplicado em cada conjunto o esquema *FCA*. Esta estratégia foi introduzida em [10], e é conhecida pelo nome de “Particionamento do Reuso”.

A figura 3.3 ilustra um exemplo no qual uma determinada célula foi dividida em K sub-células, e onde r_i denota o raio da sub-célula i .

Lembrando que a cada valor de distância de reuso está associado um determinado valor de N (tamanho de *cluster*), então o artigo [11] propõe uma “generalização” do esquema de particionamento do reuso.

Figura 3.3 - Particionamento do reuso com K sub-células

Tendo sido definido o valor de K , então generalizar consiste em associar a cada um dos K conjuntos de sub-células um dos valores **permitidos** para o tamanho do *cluster*.

Assim, se o sistema for bi-dimensional, e se definirmos o vetor $\mathbf{n} = (n_1, n_2, \dots, n_K)$, onde n_i representa o tamanho do *cluster* associado ao conjunto de sub-células i , então ao fazer a generalização utilizaremos $n_1=1$, $n_2=3$, $n_3=4$, $n_4=7, \dots$

Suponhamos agora que, em uma célula qualquer, já existe uma medida w da potência média do sinal recebido de cada um dos x terminais que estejam solicitando um canal nesta célula.

Suponhamos ainda que estes x terminais são ordenados de modo que os elementos w_j do “vetor de observação” $\mathbf{w} = (w_1, w_2, \dots, w_x)$ são colocados em ordem decrescente, isto é $w_1 \geq w_2 \geq \dots \geq w_x$.

Tendo sido definido o vetor $\mathbf{w}$, então a ERB da célula em questão deverá tentar maximizar o atendimento dos x terminais requerentes. Neste contexto deverá ser construído o “vetor de alocação de sub-célula” $\mathbf{a} = (a_1, a_2, \dots, a_x)$, onde o elemento a_j será igual a i se o terminal j for alocado à sub-célula i , e será igual a 0 (zero) se a solicitação deste terminal não puder ser atendida.

É evidente que o número máximo de terminais atendidos por uma determinada ERB dependerá da quantidade S de canais a ela pré-alocados. Como cada célula é aqui dividida em K sub-células, torna-se necessária a

definição de mais um vetor: o “vetor de alocação da capacidade” $c = (c_1, c_2, \dots, c_K)$, no qual o elemento c_i expressa a quantidade de canais pré-alocados à sub-célula i . Assim, $S = \sum_{i=1}^K c_i$, e o limitante natural para os valores do vetor c é:

$$\sum_{i=1}^K c_i \cdot n_i \leq C \quad (3.1)$$

Em busca da estratégia que, dado o vetor w , maximize o atendimento dos terminais em cada célula, é proposto em [11] o algoritmo Particionamento (*Sorting*), o qual descrevemos a seguir.

3.2.1 - PARTICIONAMENTO

Do grupo dos x terminais requerentes a ERB seleciona - no máximo - os S terminais de maior valor w_j . De posse deste sub-grupo de terminais, a ERB começa a alocar os de menor valor w_j na sub-célula mais externa (K). Quando todos os c_K canais da sub-célula K estiverem indisponíveis, a ERB passará a utilizar os c_{K-1} canais da sub-célula $K-1$. Este procedimento continuará até que todos as possíveis alocações sejam realizadas.

Merece ser destacado que este algoritmo apenas determina a qual sub-célula um terminal requerente será alocado, de modo que ele não define o canal específico a ser utilizado em cada alocação. Além disto, não é difícil concluir que o desempenho oferecido por este algoritmo não constitui o desempenho ótimo que pode vir a ser obtido com a aplicação do esquema de particionamento do reuso generalizado.

Esquemas do tipo particionamento do reuso não eliminam o pré-planejamento de frequências - na verdade tornam tal procedimento ainda mais complicado, devido à existência de pré-alocações intra-celulares. A próxima seção trata dos algoritmos adaptáveis à interferência, os quais efetivamente não requerem nenhum pré-planejamento, sendo portanto esquemas *DCA* puros.

3.3 - ESQUEMAS ADAPTÁVEIS À INTERFERÊNCIA

Os algoritmos pertencentes a esta categoria são do tipo *DCA* puro, e podem ser divididos em três outras sub-categorias: esquemas de controle centralizado, esquemas de controle semi-distribuído e esquemas de controle distribuído.

3.3.1 - CONTROLE CENTRALIZADO

É proposta em [12] uma técnica de controle centralizado para a definição dos canais disponíveis em cada alocação, e os algoritmos Alvo (seção 3.3.1.1) e Mascaramento (seção 3.3.1.2) são aplicados à luz desta técnica.

Nela um canal de frequência c só poderá ser alocado a um usuário x se duas condições forem satisfeitas:

- o nível de interferência oferecido no canal c para o usuário x estiver dentro do limite tolerável.
- a entrada de x em c não causar excessiva interferência nas chamadas já em andamento neste canal.

A figura 3.4 ilustra o princípio do mecanismo de verificação das interferências. As medidas em dB nela apresentadas referem-se às perdas de propagação ocorridas em cada enlace.

Figura 3.4 - Mecanismo de verificação das interferências

Suponhamos que o usuário x está sob controle da ERB A e está usando o canal c . Neste caso quatro medidas de SIR deverão ser realizadas para definir se o usuário y poderá conectar-se à ERB B usando o mesmo canal c .

Se as duas ERBs e se os terminais dos dois usuários usam potência de transmissão constante, então a SIR no terminal x é igual a $(70-30 = 40 \text{ dB})$, a SIR no terminal y é igual a $(80-50 = 30 \text{ dB})$, a SIR na ERB A é igual a $(80-30 = 50 \text{ dB})$, e a SIR na ERB B é igual a $(70-50 = 20 \text{ dB})$. Destes cálculos conclui-se que o usuário y só poderá usar o canal c se o nível de SIR pré-definido no sistema for $\leq 20 \text{ dB}$.

A técnica aqui apresentada não constitui um algoritmo de alocação de canais. Ela apenas indica quais canais estão disponíveis a cada nova alocação. De posse desta informação é então aplicado um algoritmo de alocação, que decidirá qual canal específico será alocado.

Merece ainda ser destacado que a utilização de um controle centralizado requer o conhecimento de todas as medidas de interferência do sistema, o que certamente dificulta sua aplicação prática.

3.3.1.1 - ALVO

Com relação ao algoritmo Alvo (*Target*), sua operação define inicialmente um valor de objetivo (alvo) para a SIR - tipicamente 5 dB acima do limiar SIR_F pré-estabelecido.

Vamos definir agora $SIR_x(c)$ como a SIR experimentada pelo usuário x ao utilizar o canal c . Consideremos ainda que n usuários estão utilizando este canal. Neste contexto, define-se Q_c como o desvio médio quadrático das $SIRs$ dos usuários de c com relação ao valor alvo T , de modo que:

$$Q_c = \frac{1}{n} \sum_{j=1}^n [SIR_j(c) - T]^2 \quad (3.2)$$

Quando um usuário y solicita um canal a uma ERB A , o sistema determina o conjunto dos canais que podem vir a ser utilizados. De posse deste

conjunto, a ERB A tenta alocar ao usuário y o canal c que possui o menor valor de Q_c . É importante ressaltar que durante esta análise os cálculos de cada Q_c incluem ainda o termo $SIR_y(c)$ relativo ao usuário requerente y , de modo que o somatório da equação (3.2) passa a ter $n+1$ ao invés de n termos.

3.3.1.2 - MASCARAMENTO

O algoritmo Mascaramento (*Masking*) [12] também faz uso da técnica centralizada descrita na seção 3.3.1, e nele, para cada solicitação de canal, é também utilizado o algoritmo Alvo. A diferença é que o Mascaramento define ainda um período de amostra - no artigo igual a 5 segundos - e a cada novo período é possível que várias chamadas em andamento sejam realocadas (troquem de canal), de modo a manter maximizado o tráfego carregado no sistema.

Nestes procedimentos de realocações, os terminais ativos são classificados em tipos de acordo com seus potenciais de interferência, sendo que normalmente quanto maior for a distância entre um terminal e a ERB que o controla maior será o potencial de interferência deste terminal. O algoritmo então tenta agrupar o maior número possível de terminais de um mesmo tipo em cada canal.

O agrupamento - e consequente alocação - começa com os terminais de menor potencial de interferência, e prossegue admitindo os terminais que têm progressivamente maior potencial de interferência até que todas as possíveis alocações sejam realizadas.

A idéia portanto é dar prioridade aos terminais que podem ser agrupados com muitos outros em um mesmo canal. Em outras palavras os terminais de maior potencial de interferência possuem níveis maiores de P_B e de P_F .

O termo mascaramento vem do fato de que “máscaras” são as ferramentas utilizadas nos procedimentos de classificação e de agrupamento dos terminais.

Cada um dos terminais ativos possui a sua “máscara de terminal”, assim como cada canal de tráfego possui a sua “máscara de canal”. Ambos os tipos de máscara são vetores de símbolos, e o número de elementos de cada um destes vetores é igual ao número de células do sistema.

Em cada novo procedimento de classificação, cada um dos b elementos das máscaras de terminal assume um dentre três possíveis estados: p , i ou n .

Vamos definir ERB “preferida” de um usuário x como sendo a ERB cujo sinal recebido em x é mais forte do que os sinais recebidos de todas as outras ERBs do sistema. Neste contexto os possíveis estados de cada elemento da máscara de terminal de x são os seguintes:

- estado p : o elemento b corresponde à ERB preferida (p) de x . Em cada uma das máscaras, apenas um dos elementos assume este estado.
- estado i : o elemento b corresponde a uma ERB interferente (i) de x , isto é, se um canal da ERB preferida de x for alocado a este usuário, então este canal não poderá ser reusado na ERB i .
- estado n : o elemento b corresponde a uma ERB não-interferente (n) de x , isto é, se um canal da ERB preferida de x for alocado a este usuário, então este canal poderá ser reusado na ERB n .

A cada uma das máscaras de terminal é atribuído ainda um peso, sendo que o peso da máscara m corresponde ao número de elementos de m cujo estado equivale a p ou i .

Neste contexto se a máscara de terminal de um usuário tem peso igual a 1 , então este usuário possui baixo potencial de interferência. Se um canal é alocado a um usuário deste tipo, então este canal pode ser reusado em várias outras células do sistema.

Por outro lado, se a máscara de terminal de um usuário tem peso igual ao número de células, então este usuário possui grande potencial de interferência. Se um canal é alocado a um usuário deste tipo, então este canal não pode ser reusado em nenhuma outra célula do sistema.

Com relação às máscaras de canal, cada elemento de cada máscara também assume um dentre os estados p , i ou n . Além disto, da mesma forma que para as máscaras de terminal, a cada máscara de canal é também atribuído um peso.

O peso da máscara de um canal c indica o número de ERBs proibidas de usar tal canal, isto é, indica quão indisponível está o canal c . Maiores detalhes a respeito do funcionamento do algoritmo Mascaramento são encontrados em [12].

3.3.2 - CONTROLE SEMI-DISTRIBUÍDO

3.3.2.1 - *LODA* (ALOCÇÃO DINÂMICA LOCALMENTE OTIMIZADA)

No algoritmo *LODA*, a cada solicitação, a célula tenta alocar o canal de menor “custo”, sendo que aqui minimizar o custo significa minimizar a distância de reuso de cada canal [8]. O objetivo é tentar compactar ao máximo as distâncias entre as células que utilizam um mesmo canal, de modo que este canal seja reusado o maior número de vezes possível.

A estratégia é dita “localmente otimizada” porque as informações disponíveis para cada alocação pertencem apenas a uma determinada região de vizinhança da célula envolvida. O *LODA*, portanto, pode ser definido como um esquema semi-distribuído (ou semi-centralizado) de alocação de canais.

Apesar de não utilizar medidas de interferência, o *LODA* pode também ser encarado como um algoritmo adaptável à interferência. Maiores detalhes a respeito de seu funcionamento são encontrados em [8].

3.3.3 - CONTROLE DISTRIBUÍDO

3.3.3.1 - SEGREGAÇÃO DE CANAIS

Usando o algoritmo *Channel Segregation* [13] todos os canais são inicialmente colocados à disposição de todas as células, e em cada célula é atribuído um valor de prioridade a cada canal.

A figura 3.5 ilustra o procedimento utilizado por uma ERB na ocorrência de uma solicitação de canal.

Quando a ERB de uma célula recebe uma solicitação de canal, ela toma para análise o canal livre de maior prioridade. Se o nível de interferência sentido neste canal estiver abaixo do máximo tolerado, então o valor de prioridade deste canal será aumentado e ele será alocado à solicitação em questão.

Se, ao contrário, o nível de interferência medido estiver acima do máximo permitido, então o valor de prioridade do canal será diminuído e a ERB passará a analisar o próximo canal livre de maior prioridade. Se todos os canais livres forem analisados e nenhum propiciar a qualidade desejada, então a chamada em questão será bloqueada ou interrompida.

Conclui-se que se um canal i for usado com sucesso por uma ERB A , então esta ERB tenderá a tentar usar novamente este canal. Ao mesmo tempo as ERBs vizinhas tenderão a sentir níveis altos de interferência para i , decrementando o valor de prioridade deste canal. Nesta situação o canal i terá sido “capturado” pela ERB A .

À medida que o tempo passar, ERBs vizinhas tenderão a capturar canais diferentes, de modo que no estado estacionário os níveis de interferência tenderão a ser pequenos.

O termo segregação representa uma analogia com o mundo da ecologia. Ecologicamente falando, segregação constitui o processo no qual cada animal captura seu território após uma série de conflitos com outros animais.

Figura 3.5 - Fluxograma do Segregação de Canais em cada ERB

No algoritmo Segregação de Canais o critério adotado para a atualização do valor de prioridade $p(i)$ de um canal i em cada uma das ERBs é o seguinte:

- Se o nível de interferência for considerado tolerável, então:

$$p(i) \leftarrow \frac{[p(i).n(i)] + 1}{n(i) + 1} \quad \& \quad n(i) \leftarrow n(i) + 1 \quad (3.3)$$

onde:

$$0 \leq p(i) \leq 1.$$

$n(i)$ é o número de vezes que a ERB envolvida já analisou o canal i .

- Se, por outro lado, o nível de interferência for considerado intolerável, então:

$$p(i) \leftarrow \frac{p(i).n(i)}{n(i)+1} \quad \& \quad n(i) \leftarrow n(i)+1 \quad (3.4)$$

Segregação de Canais, portanto, constitui um algoritmo de controle distribuído, visto que as informações utilizadas por uma ERB em cada alocação são todas localmente armazenadas.

Merece ser destacado ainda que este algoritmo foi - ao lado do *CBWL* (seção 3.1.5) - um dos escolhidos para as simulações deste trabalho (capítulo 4).

3.4 - ESQUEMAS COMPLEMENTARES

Os algoritmos descritos nesta seção não constituem esquemas de alocação de canais propriamente ditos. Na verdade, cada um destes algoritmos pode ser aplicado em conjunto à utilização de algum esquema estrito de alocação de canais.

3.4.1 - DIMENSIONAMENTO DO SISTEMA

Os algoritmos apresentados nas seções 3.4.1.1 (*SPI*), 3.4.1.2 (*SP2*) e 3.4.1.3 (*MP*), constituem soluções para o problema de dimensionamento básico de um sistema celular: a decisão sobre quantos canais de tráfego devem ser alocados a cada célula para que seja atingido um nível pré-especificado de Grau de Serviço (*GOS*) [14].

Sabe-se que na prática as solicitações de canal feitas a cada célula estão relacionadas a um de dois diferentes tipos de chamada: chamada nova ou

chamada em andamento (caso de *handoff*), de modo que durante o dimensionamento deve-se garantir que o *GOS* será atingido para ambos os tipos.

Existe um consenso de que é mais desagradável para o usuário ter sua chamada interrompida durante a conversação do que não obter acesso para uma nova chamada. Isto torna desejável que o sistema, de alguma forma, priorize as solicitações de *handoff* com relação às demais solicitações.

Dentre as maneiras possíveis de se dar prioridade às chamadas de *handoff*, os algoritmos em questão utilizam um método simples. Mencionado como *CPS* (Esquema de Prioridade com Corte), o método consiste em reservar alguns canais em cada célula para atender às chamadas em andamento.

Neste contexto, para cada célula i do sistema, é determinado o número total de canais (S_i) e o número de canais reservados necessários para que seja atendido o *GOS* definido para ambos os tipos de chamada.

Assim, para uma célula i , o termo “corte” está relacionado à definição de um certo valor, $< S_i$, abaixo do qual a quantidade atual de canais ocupados deve estar para que uma nova chamada possa ser atendida pela ERB desta célula.

Definindo-se G_i como o número de canais reservados para *handoff* (canais “de guarda”) na célula i , e O_i como o número restante de canais (canais “ordinários”) desta mesma célula ($O_i = S_i - G_i$), então o esquema *CPS* funciona da forma descrita a seguir.

Na célula i ambos os tipos de chamada serão atendidos até que O_i canais fiquem indisponíveis. A partir daí as novas chamadas serão bloqueadas e as de *handoff* atendidas desde que, obviamente, ainda haja algum canal disponível. Nesta situação portanto só haverá falha no *handoff* se todos os S_i canais estiverem indisponíveis.

Antes de passarmos à descrição dos três algoritmos, apresentaremos algumas propriedades demonstradas em [14]. Se uma célula i usa o esquema *CPS*, então definindo-se $P_{Bi}(O_i, G_i)$ e $P_{Fi}(O_i, G_i)$ como os valores de P_B e de P_F

correspondentes à utilização em i de O_i canais ordinários e G_i canais de guarda, são válidas as seguintes propriedades:

- Propriedade 1: $P_{Fi}(O_i, G_i) \leq P_{Bi}(O_i, G_i)$
- Propriedade 2: $P_{Fi}(O_i, G_i + 1) < P_{Fi}(O_i + 1, G_i) < P_{Fi}(O_i, G_i)$
- Propriedade 3: $P_{Bi}(O_i + 1, G_i) < P_{Bi}(O_i, G_i) < P_{Bi}(O_i, G_i + 1)$

A segunda desigualdade da propriedade 3 indica que a probabilidade de bloqueio de novas chamadas aumenta quando um novo canal é pré-alocado como canal de guarda.

Esta desigualdade não é - a princípio - intuitiva. No entanto, quando um novo canal de guarda é pré-alocado, é diminuída a probabilidade de falha no *handoff*, e portanto é aumentada a probabilidade de haver um grande número de canais ocupados.

Por sua vez, quanto maior é o número de canais ocupados maior tende a ser a probabilidade de bloqueio de novas chamadas. Esta e as demais desigualdades das três propriedades aqui apresentadas são demonstradas em [14].

São resumidos a seguir os problemas *SP1*, *SP2* e *MP*, bem como os algoritmos correspondentes, propostos no citado artigo como solução destes problemas.

3.4.1.1 - *SP1* (CASO DE CÉLULA ÚNICA NÚMERO 1)

- Problema *SP1*:

Consideremos que S canais foram pré-alocados a uma determinada célula. De posse do valor médio das intensidades de tráfego - tanto de novas chamadas como de chamadas de *handoff* - oferecidas para esta célula, então o problema *SP1* consiste em encontrar a sub-divisão ótima (O^*, G^*) destes S canais que não apenas minimize o valor de P_F , como também garanta um nível suficiente de *GOS* para as novas chamadas.

- Algoritmo SP1:

i) $G = \text{zero}$.

ii) encontre o menor valor de O ($O \leq S$) que consegue atender ao GOS pré-definido para P_B .

iii) faça enquanto o número de canais já alocados for $< S$:

$$G = G + 1.$$

se foi violado o GOS de P_B então:

$$G = G - 1.$$

$$O = O + 1.$$

Conclui-se que, eventualmente, pode acontecer de nenhum canal vir a ser reservado para *handoff*. É mostrado em [14] que o algoritmo *SP1* encontra a solução ótima (O^*, G^*) para o problema *SP1*. Em termos práticos este algoritmo poderia ser aplicado em cada célula de um sistema que utiliza o esquema *FCA*.

3.4.1.2 - SP2 (CASO DE CÉLULA ÚNICA NÚMERO 2)

- Problema SP2:

Consideremos que foi pré-definido um nível de GOS específico para cada tipo de chamada - normalmente o de P_F mais rigoroso que o de P_B . De posse do valor médio das intensidades de tráfego - tanto de novas chamadas como de chamadas de *handoff* - oferecidas para uma determinada célula, então o problema *SP2* consiste em encontrar a alocação (O^*, G^*) que, utilizando o menor número de canais, atenda aos dois níveis de GOS pré-especificados [14].

- Algoritmo SP2:

i) $G = \text{zero}$.

ii) encontre o menor valor de O ($O \leq S$) que consegue atender ao GOS pré-definido para P_B .

iii) faça enquanto não for atendido o GOS de P_F :

$$G = G + 1.$$

se foi violado o GOS de P_B então:

$$G = G - 1.$$

$$O = O + 1.$$

Conclui-se que, novamente, pode acontecer de nenhum canal vir a ser reservado para *handoff*. É mostrado em [14] que o algoritmo $SP2$ encontra a alocação ótima (O^*, G^*) para o problema $SP2$.

Em termos práticos o $SP2$ poderia ser aplicado em cada célula de um sistema que utiliza um esquema de “alocação flexível” dos canais. Usando este tipo de esquema [15] o número total de canais alocados para cada célula é determinado ou a intervalos de tempo pré-definidos ou de acordo com a carga de tráfego oferecida.

3.4.1.3 - MP (CASO DE MULTI-CÉLULAS)

- Problema MP :

MP na verdade é a extensão do problema $SP1$ (seção 3.4.1.1) para o *cluster* como um todo. Consideremos um *cluster* j em que os C canais do sistema estão disponíveis. De posse dos valores médios das intensidades de tráfego - tanto de novas chamadas como de chamadas de *handoff* - oferecidas para cada célula i deste *cluster*, então o problema MP consiste em encontrar as alocações ótimas (O_i^*, G_i^*) para cada célula i que minimize o valor de P_F no *cluster* j como um todo. Semelhante ao $SP1$, aqui também o nível de GOS de P_B deve ser garantido em cada uma das N células i [14].

- Algoritmo MP:

Definindo-se A_i como o potencial de decréscimo na P_F do *cluster* j causado pela alocação de um canal adicional na célula i , então o algoritmo *MP* funciona da seguinte forma:

i) para toda célula i ($i = 1, 2, \dots, N$):

$G_i = \text{zero}$.

encontre o menor valor de O_i ($\sum_{i=1}^N O_i \leq C$) que atende ao *GOS* de P_B .

ii) se o número de canais já alocados for $= C$ então:

termine a execução.

caso contrário:

iii) encontre a próxima célula de maior valor de A_i .

iv) tente alocar a ela um canal adicional ($G_i = G_i + 1$ ou $O_i = O_i + 1$).

v) se foi violado o *GOS* de P_B então:

desfaça o passo iv.

vá para o passo iii.

vi) vá para o passo ii.

Aqui também pode acontecer de nenhum canal vir a ser reservado para *handoff*. O artigo citado mostra que o algoritmo *MP* encontra as alocações ótimas (O_i^*, G_i^*) para o problema *MP*. Em termos práticos este algoritmo poderia ser aplicado em cada *cluster* de um sistema que utiliza o esquema *FCA*.

Maiores detalhes a respeito de seu funcionamento são encontrados em [14].

3.4.2 - ALOCAÇÃO DE ERB

Da mesma forma que os três algoritmos anteriores, o algoritmo apresentado na próxima seção também não constitui um esquema de alocação

de canais. Este algoritmo, em particular, pode ser melhor definido como um esquema de alocação de ERB.

3.4.2.1 - RETENTATIVA DIRECIONADA

O algoritmo *Directed Retry* foi proposto em [16], é utilizado no sistema *AMPS* (Serviço Avançado de Telefonia Móvel), e consiste de uma pequena melhoria introduzida no esquema *FCA*.

Vamos novamente definir ERB “preferida” de um usuário x como sendo a ERB cujo sinal recebido em x é mais forte do que os sinais recebidos de todas as outras ERBs do sistema.

Usando Retentativa Direcionada o procedimento de inicialização do sistema é o mesmo utilizado para o *FCA*: existe um pré-planejamento de frequências. A diferença é que no momento da alocação de um canal para x , se todos os canais de sua ERB preferida estiverem indisponíveis, este usuário será direcionado para a ERB de segundo sinal mais forte, isto é, para a sua segunda ERB preferida. Caso todos os canais desta ERB também estejam indisponíveis aí então ocorrerá o bloqueio ou interrupção da chamada de x .

Na próxima seção são apresentados alguns resultados fornecidos para os algoritmos descritos neste capítulo. Maior atenção é dada aos resultados referentes ao *CBWL* e ao Segregação de Canais, por terem sido os algoritmos escolhidos para as nossas simulações (capítulo 4).

3.5 - RESULTADOS FORNECIDOS

Em cada um dos seis artigos selecionados para este trabalho são fornecidos alguns resultados de desempenho dos algoritmos neles analisados. Nesta seção a apresentação de tais resultados é dividida de acordo com estes artigos, sendo destacados os resultados do *CBWL* e do Segregação de Canais.

3.5.1 - *BCO, BDCL & LODA*

As simulações descritas em [8] envolvem os algoritmos *FCA*, *BCO*, *BDCL* e *LODA*. *SB* e *HA* não são incluídos nestas simulações porque foi mostrado em [7] que o desempenho destes algoritmos em termos de P_B é sempre pior do que o desempenho de *BCO*.

Tais simulações desconsideram o movimento dos usuários, e duas são suas conclusões principais. A primeira é que o algoritmo *BDCL* oferece o melhor desempenho, tanto para tráfego uniforme como para não-uniforme. A segunda conclusão é que, se comparado ao algoritmo *BCO*, o *LODA* oferece pior desempenho para tráfego uniforme e desempenho equivalente para tráfego não-uniforme.

3.5.2 - *CBWL*

As análises descritas em [9] envolvem apenas os algoritmos *FCA* e *CBWL*. No caso do *CBWL* são analisadas duas implementações: com Rearranjo de Canais (*CBWL/CR*) e sem Rearranjo de Canais (*CBWL/NR*).

Com relação a estas análises, os dados mencionados são os seguintes:

- os usuários não se movimentam.
- o modelo de propagação é determinístico e de acordo com a equação (2.1).
- cada célula possui $S = 24$ canais nominais.
- o sistema é bi-dimensional e possui 37 células hexagonais.
- *wrap-around* é utilizado para contornar o efeito de borda.

- define-se o parâmetro l como sendo a quantidade de canais pré-alocados em cada um dos seis grupos que podem ser emprestados às células vizinhas. O valor utilizado para l é igual a 3.

- P_B é a única medida de desempenho utilizada.

Inicialmente considera-se que o tráfego é uniforme em todo o sistema, e alguns dos resultados obtidos para $P_B = 2\%$ são apresentados nas tabelas 3.2 e 3.3.

Tabela 3.2 - Resultados do CBWL/NR para tráfego uniforme

p	CARGA DE TRÁFEGO	GANHO COM RELAÇÃO AO FCA
0 (FCA)	16.63 Erlang/célula	0.00 %
0.1	16.83 Erlang/célula	1.20 %
0.2	17.05 Erlang/célula	2.53 %
0.3	17.31 Erlang/célula	4.09 %
0.4	17.63 Erlang/célula	6.01 %
0.5	18.00 Erlang/célula	8.24 %

Tabela 3.3 - Resultados do CBWL/CR para tráfego uniforme

p	CARGA DE TRÁFEGO	GANHO COM RELAÇÃO AO FCA
0 (FCA)	16.63 Erlang/célula	0.00 %
0.1	18.22 Erlang/célula	9.56 %
0.2	19.74 Erlang/célula	18.70 %
0.3	21.00 Erlang/célula	26.28 %
0.4	21.97 Erlang/célula	32.11 %
0.5	22.72 Erlang/célula	36.62 %

Os resultados destas tabelas devem ser interpretados da seguinte forma: assumindo-se que 2% das novas chamadas podem ser bloqueadas, estamos

interessados em saber qual a carga de tráfego que cada algoritmo pode suportar nesta condição. De acordo com os resultados, o desempenho pode ser bastante melhorado com a inclusão da função Rearranjo de Canais.

Em seguida é analisado o algoritmo *CBWL/CR* para uma distribuição de tráfego não-uniforme. Aqui a célula central do sistema é definida como *hot spot* e possui carga de tráfego média oferecida 50% maior do que a carga de tráfego média oferecida às demais células - ditas células “normais”.

Na situação de tráfego não-uniforme, alguns dos resultados obtidos para $P_B = 4\%$ são apresentados nas tabelas 3.4 e 3.5. Os valores de carga de tráfego mencionados nestas duas tabelas correspondem aos níveis médios aplicados às células normais do sistema.

Tabela 3.4 - Resultados do *CBWL/CR* para a célula *hot spot*

p	CARGA DE TRÁFEGO	GANHO COM RELAÇÃO AO <i>FCA</i>
0 (<i>FCA</i>)	12.20 <i>Erlang</i> /célula	0.00 %
0.1	13.75 <i>Erlang</i> /célula	12.70 %
0.3	20.95 <i>Erlang</i> /célula	71.72 %

Tabela 3.5 - Resultados do *CBWL/CR* para as células normais

p	CARGA DE TRÁFEGO	GANHO COM RELAÇÃO AO <i>FCA</i>
0 (<i>FCA</i>)	18.35 <i>Erlang</i> /célula	0.00 %
0.1	19.60 <i>Erlang</i> /célula	6.81 %
0.3	21.65 <i>Erlang</i> /célula	17.98 %

De acordo com os resultados, os ganhos oferecidos para a célula *hot spot* são maiores do que os ganhos oferecidos no caso de tráfego uniforme. Isto é consequência direta da inexistência de qualquer flexibilidade no algoritmo *FCA*.

A conclusão final das análises de [9] é que o algoritmo *CBWL* oferece melhor desempenho que o *FCA* tanto para tráfego uniforme como para o caso de célula *hot spot* única.

Neste trabalho as simulações realizadas com o algoritmo *CBWL* (capítulo 4) têm como destaque principal a consideração de movimento dos usuários, o que permite a obtenção de resultados da probabilidade de falha no *handoff*.

3.5.3 - PARTICIONAMENTO

As simulações descritas em [11] são do tipo *snapshot*, uma técnica de análise de sistemas celulares amplamente comentada em [17]. Nesta técnica, inicialmente, os terminais ativos são “congelados” em suas respectivas posições. Somente quando isto tiver sido feito em todas as células do sistema, será aplicado o algoritmo de alocação dos canais.

O resultado de cada um destes processos - cada *snapshot* - é uma determinada quantidade de terminais não-servidos, e o desempenho é medido pelo quociente entre o valor esperado do número de terminais não-servidos e o valor esperado do número de terminais ativos.

Supondo que o número de terminais ativos em cada célula é modelado por um processo de Poisson de intensidade igual a λ , então o valor esperado do número de terminais ativos em cada célula é igual a λ .

Considerando agora que, em cada célula, a v.a. Z representa o número de terminais não-servidos devido à falta de um canal livre, e a v.a. Q representa o número de terminais não-servidos devido à ausência de um canal com qualidade suficiente, então define-se Taxa de Falha no Enlace (χ) como:

$$\chi = \frac{E[Z + Q]}{\lambda} \quad (3.5)$$

As simulações de [11] envolvem apenas os algoritmos Particionamento e *FCA*. Uma das conclusões destas simulações é que a escolha do vetor de alocação da capacidade c possui importância fundamental no desempenho

oferecido pelo algoritmo Particionamento. Na prática, quanto maior for a quantidade de canais pré-allocados às sub-células centrais, mais canais poderão ser reusados em distâncias pequenas.

Merece ser destacado que se o sistema estiver muito carregado, então o algoritmo Particionamento dificilmente atenderá às solicitações dos terminais mais distantes, haja visto que estes terminais tenderão a possuir baixos valores de potência de sinal recebido. Este tipo de inequidade no serviço oferecido é, de certa forma, também encontrado nos algoritmos *CBWL* (seção 3.1.5) e Mascaramento (seção 3.3.1.2).

Com relação aos ganhos oferecidos pelo esquema de particionamento do reuso generalizado, os valores obtidos são significativos, no entanto este esquema induz à realização de um pré-planejamento de frequências ainda mais complicado do que o requerido pelo algoritmo *FCA*.

3.5.4 - ALVO, MASCARAMENTO & RETENTATIVA DIRECIONADA

As simulações descritas em [12] envolvem cinco algoritmos. Além do *FCA*, Alvo, Mascaramento e Retentativa Direcionada é simulado ainda o algoritmo que, utilizando a técnica centralizada proposta no artigo citado, aloca ao usuário requerente o primeiro canal encontrado que satisfaz aos critérios de interferência pré-definidos. Este algoritmo é referido como *No Optimization* (Sem Otimização).

Uma das conclusões destas simulações é que os ganhos obtidos por algoritmos *DCA* puros, centralizados e adaptáveis à interferência são substanciais, mesmo quando utilizado o método mais simples (Sem Otimização). Merece no entanto ser novamente destacado que a utilização de controle centralizado dificulta a aplicação destes métodos em sistemas práticos.

O algoritmo que obtém o melhor desempenho é o Mascaramento, seguido pelo Alvo, Sem Otimização, Retentativa Direcionada e *FCA*.

Entretanto, uma outra conclusão das simulações é que, com o uso do algoritmo Mascaramento, a taxa de ocorrência de *handoffs* requerida é alta o suficiente para inviabilizar por completo a aplicação deste algoritmo nos sistemas atuais.

3.5.5 - SEGREGAÇÃO DE CANAIS

As simulações descritas em [13] envolvem três algoritmos. Além do *FCA* e Segregação de Canais é simulado ainda o algoritmo *DCA* puro distribuído que aloca ao usuário requerente o primeiro canal encontrado que satisfaz aos critérios de interferência pré-definidos. Este algoritmo é referido como *First Available* (Primeiro Disponível).

Com relação a estas simulações, os dados mencionados são os seguintes:

- somente tráfego uniforme é analisado.
- os usuários não se movimentam.
- dois sistemas diferentes são considerados: um uni-dimensional e o outro bi-dimensional.
- o sistema uni-dimensional possui ao todo $C = 10$ canais, ao passo que no bi-dimensional são usados três valores para C : 18, 36 e 54.
- o sistema uni-dimensional é composto por 4 células circulares, e o bi-dimensional por 100 células quadradas.
- para o *FCA* o tamanho do *cluster* N é igual a 9.
- P_B e P_I são as medidas de desempenho utilizadas.

Inicialmente são comparados os desempenhos dos algoritmos Segregação de Canais e Primeiro Disponível em termos de P_I . Neste caso é utilizado o sistema uni-dimensional, e a simulação assume que existem superposições entre as células. Considerando-se que estas células estão posicionadas na ordem (1, 2, 3, 4), os níveis de interferência - devido às superposições - assumidos são os seguintes:

- duas chamadas usando simultaneamente o mesmo canal i nas células 1 e 2 interferem entre si com probabilidade igual a 70%.
- nas células 1 e 3, com probabilidade igual a 30%.
- nas células 2 e 3, com probabilidade igual a 70%.
- nas células 2 e 4, com probabilidade igual a 30%.
- nas células 3 e 4, com probabilidade igual a 70%.
- as células 1 e 4 não interferem entre si.

Assumindo-se ainda que as medidas de interferência realizadas pelas ERBs falham com probabilidade igual a 10%, é mostrado que os níveis de P_i são menores com a utilização do Segregação de Canais.

Para exemplificar, quando o algoritmo Primeiro Disponível possui P_i igual a 1%, o valor obtido pelo Segregação de Canais é de aproximadamente 0.02%.

Em seguida é considerado o sistema bi-dimensional. Neste caso os três algoritmos são analisados, e para o Primeiro Disponível e o Segregação de Canais assume-se que cada célula quadrada interfere com as 24 células mais próximas, i.e. com os dois “anéis” de células que a rodeiam.

Os resultados obtidos para $P_B = 3\%$ quando $C = 36$ são os seguintes:

Tabela 3.6 - Resultados do Segregação de Canais para $C = 36$

ALGORITMO	CARGA DE TRÁFEGO	GANHO EM RELAÇÃO AO FCA
<i>FCA</i>	1.30 Erlang/célula	0.00 %
Primeiro Disponível	1.75 Erlang/célula	34.62 %
Segregação	2.20 Erlang/célula	69.23 %

Considerando-se agora $C = 54$, os resultados obtidos para $P_B = 3\%$ são os apresentados na tabela 3.7.

Conclui-se que o algoritmo Segregação de Canais, além de evitar o pré-planejamento de frequências e de permitir o controle distribuído, consegue ainda obter ganhos razoáveis quando comparado ao *FCA*.

Tabela 3.7 - Resultados do Segregação de Canais para $C = 54$

ALGORITMO	CARGA DE TRÁFEGO	GANHO EM RELAÇÃO AO <i>FCA</i>
<i>FCA</i>	2.55 Erlang/célula	0.00 %
Primeiro Disponível	2.85 Erlang/célula	11.76 %
Segregação	3.55 Erlang/célula	39.22 %

Comparando-se as tabelas 3.6 e 3.7 nota-se que os ganhos oferecidos por algoritmos *DCA* diminuem quando aumenta a quantidade de canais (C) no sistema. Este tipo de resultado é também encontrado em [17].

Neste trabalho as simulações realizadas com o algoritmo Segregação de Canais (capítulo 4) têm como destaque principal a consideração de movimento dos usuários, o que permite a obtenção de resultados da probabilidade de falha no *handoff*.

3.5.6 - *SP1*, *SP2* & *MP*

Para finalizar o capítulo 3, comentamos nesta seção os principais resultados das simulações descritas em [14]. Uma das conclusões é que o algoritmo *SP2* é mais vantajoso do que o *SP1*, haja visto obter melhor desempenho mesmo quando utiliza menor quantidade total de canais (C).

Ainda em comparação ao algoritmo *SP1*, é mostrado que a utilização do *MP* reduz as diferenças entre os valores de P_F das células do sistema. Em outras palavras, *MP* produz maior equidade entre os valores de *GOS* obtidos para cada célula.

No próximo capítulo são destacadas as **nossas** simulações, as quais envolvem os algoritmos *CBWL* e Segregação de Canais.

4 - SIMULAÇÕES

4.1 - ALGORITMOS ESCOLHIDOS

Nossas simulações envolvem os algoritmos *CBWL* [9] (seção 3.1.5 deste trabalho) e *Segregação de Canais* [13] (seção 3.3.3.1 deste trabalho), e os motivos de tal seleção são comentados a seguir.

Ao escolher esta dupla estamos lidando com as diferentes categorias de algoritmo *DCA*. De fato o *CBWL* é adaptável às variações do tráfego e à potência do sinal recebido, ao passo que o *Segregação de Canais* é *DCA* puro adaptável à interferência.

Além disto os artigos [9] e [13] desconsideram o movimento dos usuários ao longo das chamadas, tornando interessante analisar o comportamento destes algoritmos nesta situação, inclusive em casos de *handoff*.

Um terceiro motivo é que já existem trabalhos - [18] e [19] - considerando a aplicação destes algoritmos em sistemas digitais, os quais tendem a substituir os sistemas analógicos (*FDMA* puros) em todo o mundo.

Finalmente, de nossa parte, acreditamos que *CBWL* e *Segregação de Canais* são dois dos mais importantes algoritmos de alocação de canais já apresentados.

4.2 - AMBIENTE OPERACIONAL

A linguagem de programação utilizada é o *Turbo Pascal* para *DOS*, e após cada execução - em que todos os usuários são examinados - o programa exibe os valores atualizados das seguintes variáveis:

- número acumulado de novas chamadas iniciadas no sistema.
- número acumulado de novas chamadas bloqueadas.

- taxa de bloqueio de novas chamadas.
- número acumulado de tentativas de *handoff* no sistema.
- número acumulado de tentativas de *handoff* mal-sucedidas.
- taxa de falha no *handoff*.
- quantidade de canais ocupados em cada célula. Ao ser atingido o número máximo permitido para uma determinada célula o valor correspondente será exibido em tonalidade vermelha.
- número acumulado de execuções. Quando o tempo de simulação equivale a *uma* hora, o programa é interrompido.

Além destas variáveis o programa exibe uma representação de rede celular uni-dimensional. Nesta rede as ERBs são representadas pelo caracter “+”, as fronteiras entre células pelo caracter “|” e os usuários pelo caracter “!”. Se o usuário está inativo (isto é, não está usando um canal) ele é exibido em tonalidade azul, ao passo que se ele está ativo (isto é, está em conversação) é exibido em tonalidade verde.

4.3 - CARACTERÍSTICAS DO SISTEMA

Visando simplificar os algoritmos, nosso sistema celular é uni-dimensional. A estrutura uni-dimensional pode, no entanto, representar grandes avenidas urbanas ou, ainda, regiões rurais ao longo de auto-estradas.

O sistema possui ao todo 12 células, cada uma com diâmetro de 900 metros (micro-células, portanto). São modeladas antenas omni-direcionais, estando cada ERB localizada no centro da célula. Para contornar os efeitos de borda é utilizada a estratégia de *wrap-around*, de modo que a célula 1 é vizinha da célula 12.

O sistema é do tipo PCS (Sistema de Comunicações Pessoais), e todos os seus usuários são pedestres. O comportamento dos usuários é então

modelado da seguinte forma: se um usuário está inativo, são idênticas as probabilidades de que ele esteja parado ou andando (com velocidade igual a 5 Km/hora) em um dos dois sentidos. Se, no entanto, este usuário está ativo, é grande a probabilidade (80%) de que ele esteja parado.

Durante a inicialização do programa todos os usuários são distribuídos de maneira uniforme ao longo do sistema. Entretanto, a contínua movimentação destes usuários resulta na geração de várias situações aleatórias de tráfego não-uniforme.

O sistema possui ao todo 384 usuários e 48 canais de tráfego. O tempo médio de duração de cada chamada é de 2 minutos, e o valor médio do intervalo de tempo entre novas chamadas de cada usuário constitui o parâmetro utilizado para a variação da carga de tráfego do sistema. A carga de tráfego é aqui normalizada como o (número acumulado de novas chamadas / número de células / número total de canais) em *uma* hora de simulação.

Nosso modelo de propagação é determinístico, de acordo com a equação (2.1), e com parâmetro α igual a 3.5. O motivo principal para não considerarmos desvanecimentos é a simplificação das análises. No entanto, os efeitos causados pelo desvanecimento rápido são menos significantes para usuários pedestres do que para usuários em automóveis.

Além disto, os efeitos causados pelo desvanecimento lento são menos severos em sistemas micro-celulares do que em sistemas de grandes células [17]. Usualmente considera-se como micro-celular um sistema em que o diâmetro das células é menor do que 1 Km [20].

Ainda com referência ao modelo de propagação, nossas medidas consideram apenas a interferência co-canal, desconsiderando assim as interferências de canais adjacentes. Além disto estas medidas são todas referentes ao *downlink* (sentido de transmissão da ERB para o usuário).

Para determinar o valor do limiar SIR_B , consideramos como referência o esquema *FCA*, de modo que o cálculo deste valor refere-se ao “pior caso de

interferência”. Nesta situação supõe-se que o usuário analisado está situado na fronteira de sua célula e, além disto, assume-se que todas as ERBs interferentes estão utilizando o canal alocado a este usuário. Assim, para N igual a 3:

Figura 4.1 - Pior caso de interferência para $N = 3$ usando FCA

onde: ! $\equiv$ usuário em questão

+ $\equiv$ ERB preferida deste usuário

* $\equiv$ ERBs interferentes

$R = 0.5$

$D = 3$

Assumindo-se ainda que os sinais interferentes são independentes, e que todas as ERBs utilizam a mesma potência de transmissão, temos o seguinte:

$$SIR_B = \frac{R^{-\alpha}}{(D-R)^{-\alpha} + (2D-R)^{-\alpha} + (D+R)^{-\alpha}} = \frac{(0.5)^{-3.5}}{(2.5)^{-3.5} + (5.5)^{-3.5} + (3.5)^{-3.5}} \cong 203.8 \equiv 23.1dB$$

Considerando agora N igual a 2 (e, portanto, $D = 2$):

Figura 4.2 - Pior caso de interferência para $N = 2$ usando FCA

Neste caso:

$$SIR_B = \frac{R^{-\alpha}}{(D-R)^{-\alpha} + (2D-R)^{-\alpha} + (3D-R)^{-\alpha} + (D+R)^{-\alpha} + (2D+R)^{-\alpha}} \cong 37.4 \equiv 15.7dB$$

Os dois únicos valores de tamanho de *cluster* considerados foram 3 e 2. O motivo é que valores maiores ou igual a 4 fornecem limiares muito distantes da realidade prática (e.g., para $N = 4$, $SIR_B \cong 28.1 \text{ dB}$).

4.4 - PRIORIDADE A *HANDOFF*

A grande maioria dos artigos considerando alocação de canais utiliza como única medida de desempenho a probabilidade de bloqueio de novas chamadas (P_B). Desta forma é comum encontrarmos argumentos do tipo “nosso algoritmo permite um aumento de 50% na carga de tráfego suportada pelo sistema”, quando, na realidade, este valor refere-se apenas ao limiar de qualidade - normalmente igual a 2% - usado exclusivamente para P_B .

Parece ser novamente destacado, no entanto, que é mais desagradável para o usuário ter sua chamada interrompida durante a conversação do que não obter acesso a uma nova chamada. Isto faz com que, normalmente, seja definido como limiar de qualidade para a probabilidade de falha no *handoff* (P_F) um valor abaixo do utilizado para P_B .

O grande problema de se comparar - com o *FCA* - um novo algoritmo que diminui o bloqueio de chamadas, e usar apenas P_B como medida de desempenho, é explicado da seguinte forma:

Com o uso do novo algoritmo, a cada instante, uma certa quantidade de usuários poderá ocupar canais que não poderiam ser ocupados se o sistema usasse o *FCA*. Esta situação traduz-se em mais usuários ativos e menos canais disponíveis.

O aumento da quantidade de usuários ativos provoca naturalmente o aumento da probabilidade de que tentativas de *handoff* sejam realizadas. Por outro lado a redução da quantidade de canais disponíveis faz com que, a menos que alguma medida seja tomada, aumente também o valor de P_F .

Tendo em mente que é desejável que o limiar de qualidade de P_F fique abaixo do limiar de P_B , tiramos duas conclusões da situação descrita acima.

- é melhor considerar como medida de desempenho algum grau de serviço (*GOS*) que envolva tanto P_B como P_F .
- alguma prioridade de alocação deve ser dada às tentativas de *handoff*, do contrário é muito provável que P_F venha a ser o grande limitante para o *GOS*.

Em nossas simulações definimos *GOS* como sendo o valor de carga de tráfego que o algoritmo pode suportar mantendo **simultaneamente** $P_B \leq 2\%$ e $P_F \leq 1\%$.

Existe mais de uma maneira de se dar prioridade a *handoff*. Uma alternativa seria enfileirar as solicitações de *handoff* - no caso de momentânea indisponibilidade de canais - até ser possível o atendimento. Para tal alternativa o ponto crítico é o tempo de espera que cada solicitação pode experimentar.

Em nossas simulações, por sua simplicidade, optamos pelo método de reservar alguns canais específicos para o atendimento das tentativas de *handoff*, sendo que, ao fazermos isto, estamos utilizando parte das idéias apresentadas na seção 3.4.1.

O método aqui empregado, no entanto, é um pouco diferente do Esquema de Prioridade com Corte utilizado naquela seção. Aqui, quando uma célula recebe um pedido de alocação de uma chamada em situação de *handoff*, ela tenta inicialmente alocar um dos canais - “de guarda” - reservados para este fim. Se todos estes canais estiverem indisponíveis, então ela tentará alocar um dos demais canais (“ordinários”).

Esta inversão na ordem das alocações é feita com o intuito de fornecer uma idéia melhor a respeito da quantidade de tentativas de *handoff* atendidas em cada célula do sistema. A inversão permite ainda que o desempenho em termos de P_B seja menos prejudicado do que no esquema da seção 3.4.1.

As próximas duas seções apresentam em detalhes as simulações dos algoritmos *CBWL* e Segregação de Canais.

4.5 - CBWL

4.5.1 - ADAPTAÇÃO À ESTRUTURA UNI-DIMENSIONAL

Merece aqui ser novamente destacado que, neste algoritmo, os canais tomados emprestados devem ser utilizados com potência de transmissão reduzida, de modo a evitar os travamentos. Por isto, somente aqueles usuários que estejam razoavelmente próximos de suas ERBs podem vir a acessar canais desta natureza. Usuários habilitados a usar tais canais são ditos do tipo *A* (ou tipo *1*), ao passo que os demais são ditos do tipo *B* (ou tipo *2*).

Conclui-se que o parâmetro fundamental a ser determinado para a análise de *CBWL* é o tamanho da região, de cada célula, em que canais tomados emprestados podem vir a ser utilizados. No caso particular de um sistema uni-dimensional, em cada uma das metades de uma célula, este parâmetro traduz-se em uma determinada porção do raio R , que aqui será referenciada como r .

Para determinar o valor de r nos casos de $N = 3$ e de $N = 2$, seguimos o procedimento descrito em [9], onde a idéia básica é garantir que a qualidade dos sinais oferecida pelo *CBWL* não seja pior do que a oferecida pelo esquema *FCA*. A diferença é que, neste trabalho, considera-se a interferência de todas as células co-canais, e não apenas a da célula interferente mais próxima. Assim, para N igual a 3:

- calcula-se inicialmente a razão sinal-interferência para um canal tomado emprestado (SIR_{bc}), sendo novamente considerado o pior caso de interferência.

Figura 4.3 - Pior caso de interferência em canal tomado emprestado ($N=3$)

onde: $!$ $\equiv$ usuário em questão

$+$ $\equiv$ ERB preferida deste usuário

- $\equiv$ ERB que emprestou o canal
- * $\equiv$ ERBs interferentes
- $D = 3$

Assumindo-se que os sinais interferentes são independentes, então:

$$SIR_{bc} = RP \cdot \left[\frac{r^{-\alpha}}{(D-1-r)^{-\alpha} + (2D-1-r)^{-\alpha} + (D+1+r)^{-\alpha}} \right]$$

onde RP é a razão entre a potência transmitida em um canal tomado emprestado (P_{bc}), e a potência (P_{rc}) transmitida em um canal regular (não-emprestado).

- em seguida calcula-se a razão sinal-interferência para um canal regular (SIR_{rc}).

Figura 4.4 - Pior caso de interferência em canal regular ($N=3$)

onde: $R = 0.5$

Neste caso:

$$SIR_{rc} = \frac{P_{rc} \cdot R^{-\alpha}}{P_{bc} \cdot [(D-1-R)^{-\alpha} + (2D-1-R)^{-\alpha}] + P_{rc} \cdot (D+R)^{-\alpha}} \cong \frac{11.31 P_{rc}}{0.25 P_{bc} + 0.01 P_{rc}}$$

Mas $P_{bc} = RP \cdot P_{rc}$, logo:

$$SIR_{rc} \cong \frac{11.31}{0.25RP + 0.01}$$

- para garantir que a qualidade oferecida pelo $CBWL$ é equivalente à do FCA , iguala-se SIR_{rc} ao valor de SIR_B obtido na seção 4.3. Assim:

$$SIR_{rc} \cong \frac{11.31}{0.25RP + 0.01} = 203.8 \cong SIR_B \quad \text{e portanto:} \quad RP \cong 0.18$$

- substituindo agora o valor de RP na expressão de SIR_{bc} , então:

$$SIR_{bc} = 0.18 \cdot \left[\frac{(r)^{-3.5}}{(2-r)^{-3.5} + (5-r)^{-3.5} + (4+r)^{-3.5}} \right]$$

- finalmente, iguala-se também SIR_{bc} ao valor de SIR_B ($\cong 203.8$), e o resultado final obtido, quando $N = 3$, é $r \cong 0.23$.

Considerando-se agora o caso em que $N = 2$ ($D = 2$), o mesmo procedimento é realizado.

- calcula-se inicialmente a razão sinal-interferência para um canal tomado emprestado (SIR_{bc}).

Figura 4.5 - Pior caso de interferência em canal tomado emprestado ($N=2$)

Assumindo-se que os sinais interferentes são independentes, então:

$$SIR_{bc} = RP \cdot \left[\frac{r^{-\alpha}}{(D-1-r)^{-\alpha} + (2D-1-r)^{-\alpha} + (3D-1-r)^{-\alpha} + (D+1+r)^{-\alpha} + (2D+1+r)^{-\alpha}} \right]$$

- em seguida calcula-se a razão sinal-interferência para um canal regular (SIR_{rc}).

Figura 4.6 - Pior caso de interferência em canal regular ($N=2$)

Neste caso:

$$SIR_{rc} = \frac{P_{rc} \cdot R^{-\alpha}}{P_{bc} \cdot [(D-1-R)^{-\alpha} + (2D-1-R)^{-\alpha} + (3D-1-R)^{-\alpha}] + P_{rc} \cdot [(D+R)^{-\alpha} + (2D+R)^{-\alpha}]}$$

E portanto:

$$SIR_{rc} \cong \frac{11.31 P_{rc}}{11.36 P_{bc} + 0.05 P_{rc}} \equiv \frac{11.31}{11.36 RP + 0.05}$$

- para garantir que a qualidade oferecida pelo *CBWL* é equivalente à do *FCA*, iguala-se SIR_{rc} ao valor de SIR_B obtido na seção 4.3. Assim:

$$SIR_{rc} \cong \frac{11.31}{11.36RP + 0.05} = 37.4 \cong SIR_B \quad \text{e portanto:} \quad RP \cong 0.02$$

- substituindo agora o valor de RP na expressão de SIR_{bc} , então:

$$SIR_{bc} = 0.02 \cdot \left[\frac{(r)^{-3.5}}{(1-r)^{-3.5} + (3-r)^{-3.5} + (5-r)^{-3.5} + (3+r)^{-3.5} + (5+r)^{-3.5}} \right]$$

- finalmente, iguala-se também SIR_{bc} ao valor de SIR_B ($\cong 37.4$), e o resultado final obtido, quando $N = 2$, é $r \cong 0.1$.

De acordo com os cálculos para a determinação de r , o valor deste parâmetro está diretamente relacionado ao tamanho do *cluster* considerado, de modo que, tendo sido definido o valor de N , r representa a máxima distância que um usuário pode estar com relação à sua ERB para que ele, ao usar um canal tomado emprestado, produza níveis de interferência no máximo iguais aos níveis produzidos por um usuário do esquema *FCA*.

Em sistemas uni-dimensionais cada célula possui apenas duas células vizinhas, de modo que, usando o *CBWL*, os canais pré-alocados a cada célula são sub-divididos em 3 grupos: $(1, 0, 2)$. Lembrando que l representa a quantidade de canais pré-alocados a cada um dos grupos - 1 e 2 - que podem ser emprestados, e que S representa a quantidade total de canais de cada célula, então, se o sistema analisado é homogêneo, os grupos de canais em cada célula têm a forma $(l, S-2l, l)$.

4.5.2 - *HANDOFF* INTRA-CELULAR

Com a utilização do *CBWL* um procedimento de *handoff* intra-celular deve ocorrer sempre que um usuário do tipo 1 está usando um canal tomado emprestado e ultrapassa o limiar de distância permitido (parâmetro r). De fato, nesta situação, o usuário passa a ser do tipo 2, e deverá tentar usar um dos canais regulares da célula em que se encontra.

Na próxima seção são apresentados os resultados obtidos pelo algoritmo *CBWL* em nossas simulações.

4.5.3 - RESULTADOS OBTIDOS

A existência de um pré-planejamento rigoroso, e a utilização de um modelo de propagação determinístico, levam-nos a realizar as simulações do *CBWL* - e também do *FCA* - tomando como base apenas as distâncias entre os usuários e as ERBs. Em outras palavras, assumimos que o planejamento eficiente, aliado à ausência de desvanecimentos, garante razoável confiabilidade mesmo sem a utilização de medidas da qualidade dos sinais.

Neste contexto um procedimento de *handoff* inter-celular deve ocorrer sempre que um usuário ultrapassa uma fronteira entre duas células.

Ao simular o *CBWL* em nosso ambiente operacional, além das variáveis referidas na seção 4.2, o programa exibe ainda os dados estatísticos dos *handoffs* do tipo intra-celular. Além disto, para cada nova execução, são fornecidas indicações do tipo “Canal x capturado (liberado) pelo usuário y na célula z ”.

Consideramos inicialmente o tamanho do *cluster* N igual a 3. Neste caso, usando um sistema homogêneo, 16 canais ($S = C/N = 48/3$) são pré-alocados a cada célula.

Tendo sido definido $S = 16$, três valores são simulados para o parâmetro l : 4, 5 e 6, de modo que as pré-alocações $(l, S-2l, l)$ aqui analisadas são (4, 8, 4), (5, 6, 5) e (6, 4, 6).

Lembrando que para o caso do *FCA* considera-se $r = 0$, os resultados obtidos são os apresentados na tabela 4.1.

A conclusão inicial é que os ganhos oferecidos em termos de P_B (até 12.9%) são bastante prejudicados pelas limitações em P_F . No caso em que $l = 4$ esta limitação chega a provocar perda de *GOS* com relação ao *FCA*.

Tabela 4.1 - Resultados do *CBWL* básico para $N = 3$

ALGORITMO	CARGA P_B	CARGA P_F	GOS	LIMITE	GANHO OFERECIDO
<i>FCA</i>	6.2	5.9	5.9	P_F	0.00 %
(4, 8, 4)	7.0	5.6	5.6	P_F	- 5.08 %
(5, 6, 5)	7.0	6.4	6.4	P_F	8.47 %
(6, 4, 6)	6.9	5.9	5.9	P_F	0.00 %

onde: P_B $\equiv$ taxa de bloqueio de novas chamadas após *uma* hora de simulação.

CARGA P_B $\equiv$ carga de tráfego suportada para $P_B = 2\%$.

P_F $\equiv$ taxa de falha no *handoff* após *uma* hora de simulação.

CARGA P_F $\equiv$ carga de tráfego suportada para $P_F = 1\%$.

GOS $\equiv$ menor dos valores entre *CARGA P_B* e *CARGA P_F* .

LIMITE indica se o limitante para o *GOS* é P_B ou P_F .

GANHO está relacionado ao *GOS* obtido pelo algoritmo *FCA*.

Em busca de melhores desempenhos, estudamos as melhorias propostas no artigo [9] original e em outros artigos - [21] e [22] - do *CBWL*. A maioria das estratégias - Troca de Canais (*Channel Swapping*), Rearranjo de Canais (*Channel Rearrangement*) e Retorno Rápido de Canais (*Channel Fast Returning*) - é do tipo “prejudica a qualidade do serviço de quem está falando para atender a uma nova solicitação”. Falando claramente, estas três estratégias prevêem constantes trocas de canais durante as chamadas, de modo que, pelo menos a princípio, preferimos evitá-las.

A estratégia restante é chamada Prioridade com Corte (*Cut-Off Priority*) [22], cujo objetivo é impedir a proliferação de empréstimos desnecessários entre as células. Usando prioridade com corte, define-se um limite m ($m < S$) no número de canais ocupados em uma célula para que ela aceite emprestar um de

seus canais disponíveis. Assim, se m ou mais canais de uma célula já estiverem ocupados, então ela rejeitará qualquer pedido de empréstimo.

Merece ser destacado que o corte, neste caso, está relacionado aos empréstimos de canais, sendo portanto diferente do corte definido no esquema *CPS* (seção 3.4.1), o qual está relacionado ao atendimento de novas chamadas.

Os resultados obtidos com o uso de prioridade com corte são os seguintes:

Tabela 4.2 - Resultados do *CBWL* usando prioridade com corte para $N = 3$

ALGORITMO	CARGA P_B	CARGA P_F	GOS	LIMITE	GANHO OFERECIDO
(4,8,4) $m=15$	6.9	5.5	5.5	P_F	- 6.78 %
(5,6,5) $m=15$	6.9	6.2	6.2	P_F	5.08 %
(6,4,6) $m=15$	7.0	5.5	5.5	P_F	- 6.78 %

Os desempenhos obtidos são decepcionantes. Em termos de P_B , a única situação de melhoria é para o caso de $l = 6$, e mesmo assim o ganho oferecido é menor do que a perda ocasionada em P_F .

É provável, no entanto, que para valores mais altos de carga de tráfego (mantendo o mesmo valor de N), e também para sistemas bi-dimensionais, a estratégia de prioridade com corte venha realmente a melhorar o desempenho. No caso dos sistemas bi-dimensionais, o motivo aparente é que neles cada célula possui *seis* (ao invés de *duas*) células vizinhas, aumentando assim a probabilidade da proliferação de empréstimos desnecessários.

Para conseguir ganhos maiores com o *CBWL*, pensamos ainda na hipótese de variar o valor do parâmetro r - afinal r foi dimensionado para o pior caso de interferência. Entretanto, ao simular situações deste tipo, estaríamos privilegiando o *CBWL* com relação ao *FCA*, haja visto que o cálculo de r está diretamente relacionado ao tamanho do *cluster* utilizado. Esta hipótese, contudo, poderia ser testada se o algoritmo simulado considerasse medidas de potência - e não de distância - em sua operação.

Finalizando as simulações do *CBWL* para $N = 3$ é utilizada a estratégia de prioridade a *handoff*. Definindo-se h como a quantidade de canais reservados para *handoff* em cada célula, os resultados obtidos são os seguintes:

Tabela 4.3 - Resultados do *CBWL* com prioridade a *handoff* para $N = 3$

ALGORITMO	CARGA P_B	CARGA P_F	GOS	LIMITE	GANHO OFERECIDO
(4,8,4) $h=1$	6.3	6.1	6.1	P_F	3.39 %
(4,8,4) $h=2$	6.2	> 6.2	6.2	P_B	5.08 %
(5,6,5) $h=1$	6.6	6.6	6.6	-	11.86 %
(6,4,6) $h=1$	6.8	6.4	6.4	P_F	8.47 %
(6,4,6) $h=2$	6.4	> 6.4	6.4	P_B	8.47 %

De acordo com os resultados, para $N = 3$, o máximo ganho oferecido pelo *CBWL* é de aproximadamente 12%. A tabela 4.3 mostra ainda que, ao ser dada a prioridade a *handoff*, as três configurações simuladas de *CBWL* conseguem melhor desempenho que o *FCA*.

Merece ser destacado que foram também consideradas situações de uso simultâneo dos parâmetros m e h , no entanto os ganhos obtidos para estas situações foram menores do que os ganhos referidos na tabela 4.3.

Em seguida é considerada a situação em que $N = 2$. Neste caso, se o sistema é homogêneo, 24 canais ($S = C/N = 48/2$) são pré-alocados a cada célula.

Para $S = 24$, três valores são também simulados para o parâmetro l : 7, 8 e 9, de modo que as pré-alocações $(l, S-2l, l)$ analisadas para este caso são (7, 10, 7), (8, 8, 8) e (9, 6, 9).

Lembrando novamente que, para o caso do *FCA* considera-se $r = 0$, os resultados obtidos são os apresentados na tabela 4.4. De acordo com aqueles resultados, para $N = 2$, o ganho máximo oferecido em termos de P_B é de

aproximadamente 9.6%, sendo porém novamente prejudicado pela limitação em P_F .

Tabela 4.4 - Resultados do CBWL básico para $N = 2$

ALGORITMO	CARGA P_B	CARGA P_F	GOS	LIMITE	GANHO OFERECIDO
<i>FCA</i>	10.4	10.3	10.3	P_F	0.00 %
(7, 10, 7)	11.2	10.4	10.4	P_F	0.97 %
(8, 8, 8)	11.3	10.3	10.3	P_F	0.00 %
(9, 6, 9)	11.4	11.0	11.0	P_F	6.80 %

Seguindo os mesmos passos usados para $N = 3$, simulamos então a estratégia de prioridade com corte, e os resultados obtidos são os seguintes:

Tabela 4.5 - Resultados do CBWL usando prioridade com corte para $N = 2$

ALGORITMO	CARGA P_B	CARGA P_F	GOS	LIMITE	GANHO OFERECIDO
(7,10,7) $m=23$	11.5	10.2	10.2	P_F	- 0.97 %
(8,8,8) $m=23$	11.5	10.2	10.2	P_F	- 0.97 %
(8,8,8) $m=22$	10.9	10.2	10.2	P_F	- 0.97 %
(9,6,9) $m=23$	11.3	10.6	10.6	P_F	2.91 %

De acordo com os resultados, usando $m = 23$, há uma certa melhoria em P_B para $l = 7$ e $l = 8$, melhorias estas novamente prejudicadas pelas perdas ocasionadas em P_F .

As simulações do CBWL para $N = 2$ são então concluídas com a utilização da estratégia de prioridade a *handoff*, e os resultados obtidos são os apresentados na tabela 4.6.

De acordo com aqueles resultados, para $N = 2$, o máximo ganho oferecido pelo CBWL é de aproximadamente 10%. Nota-se ainda que, novamente, o ganho obtido pelas três configurações simuladas de CBWL melhoram quando é dada a prioridade a *handoff*.

Tabela 4.6 - Resultados do *CBWL* com prioridade a *handoff* para $N = 2$

ALGORITMO	CARGA P_B	CARGA P_F	GOS	LIMITE	GANHO OFERECIDO
(7,10,7) $h=1$	11.0	10.8	10.8	P_F	4.85 %
(7,10,7) $h=2$	10.3	> 10.3	10.3	P_B	0.00 %
(8,8,8) $h=1$	11.2	10.8	10.8	P_F	4.85 %
(8,8,8) $h=2$	11.0	> 11.0	11.0	P_B	6.80 %
(9,6,9) $h=1$	11.3	11.8	11.3	P_B	9.71 %

Também para o caso em que $N = 2$, foram consideradas situações de uso simultâneo dos parâmetros m e h . Entretanto, os ganhos obtidos para este tipo de situação foram novamente menores do que os ganhos oferecidos pela utilização isolada do parâmetro h .

Comparando-se os resultados para $N = 3$ com os resultados para $N = 2$, nota-se que o ganho oferecido por algoritmos *DCA* é maior para $N = 3$ do que para $N = 2$. Resultados deste tipo são também encontrados em [17] e [23].

A listagem do programa contendo o *CBWL* - para $N = 2$, $l = 9$ e $h = 1$ - é apresentada no apêndice A deste trabalho.

Na próxima seção são comentadas as nossas simulações realizadas com o algoritmo Segregação de Canais.

4.6 - SEGREGAÇÃO DE CANAIS

A principal vantagem deste algoritmo - e de todos os demais esquemas *DCA* puros - é a ausência do pré-planejamento de frequências, sendo isto ainda mais importante se o sistema possuir micro (ou pico) células.

No que diz respeito às simulações deste trabalho, gostaríamos de colocar inicialmente duas observações. A primeira é que a estratégia utilizada para a ordenação dos canais em cada célula - baseada nos valores atualizados de prioridade destes canais - é a do procedimento *Bubble Sort* [24].

A outra observação diz respeito à interface gráfica do programa. Ao executar o algoritmo Segregação de Canais, 5 canais são escolhidos para que seus valores atualizados de prioridade em cada célula sejam exibidos. Isto permite verificar a eficiência deste algoritmo, no sentido de que é desejável que o mesmo canal não tenha altos valores de prioridade em células vizinhas.

Para o Segregação de Canais, esta visualização substitui as indicações do tipo “Canal x capturado (liberado) pelo usuário y na célula z ”, que são usados para o *CBWL* e para o *FCA*.

4.6.1 - LIMIAR PARA *HANDOFFS*

Em algoritmos de controle distribuído - caso do Segregação de Canais - é sempre maior a chance de que a utilização de um canal por um novo usuário provoque níveis de interferência intoleráveis na(s) chamada(s) de algum(ns) outro(s) usuário(s) deste mesmo canal.

Se isto acontecer, o(s) usuário(s) incomodado(s) deverá(ão) tentar uma troca de canal, i.e. deverá(ão) solicitar um *handoff*. Neste contexto, visando manter a taxa de tentativas de *handoff* em níveis aceitáveis, definimos o valor do limiar SIR_F como sendo igual a 5 dB abaixo do limiar SIR_B calculado na seção 4.3.

Assim:

- para $N = 3$, $SIR_F = SIR_B - 5 = 23.1 - 5 = 18.1 \text{ dB}$.
- para $N = 2$, $SIR_F = SIR_B - 5 = 15.7 - 5 = 10.7 \text{ dB}$.

Visto que as simulações da seção 4.5.3 (*FCA* e *CBWL*) não fazem distinção entre SIR_F e SIR_B , então todos os algoritmos *FCA* e *CBWL* aqui considerados são diferentes dos analisados naquela seção.

A diferença básica é que, aqui, uma solicitação de *handoff* ocorre somente se existe o risco de que o nível de *SIR* caia abaixo do limiar SIR_F (e não SIR_B) durante a chamada.

Em termos do *FCA* - e do *CBWL* - isto equivale a permitir uma certa tolerância na distância entre o usuário e a ERB que o controla, de modo que, por exemplo, ao ultrapassar uma fronteira entre duas células, o usuário não solicita um *handoff* inter-celular. Neste caso, a solicitação só será feita se a distância (d_{inter}) entre o usuário e sua ERB puder resultar em um valor de *SIR* menor do que SIR_F .

Os cálculos do valor de d_{inter} (para *handoff* inter-celular) são semelhantes aos utilizados na seção 4.3 (substituindo R por d_{inter}), ao passo que para *handoff* intra-celular (no caso do *CBWL*) os cálculos de d_{intra} são semelhantes aos da seção 4.5.1 (substituindo r por d_{intra}).

Neste contexto, os valores de d_{inter} e d_{intra} aqui utilizados são os seguintes:

Tabela 4.7 - Valores de d_{inter} e d_{intra} para $N = 2$ e $N = 3$

N	r	d_{intra}	R	d_{inter}
2	0.10	0.14	0.50	0.64
3	0.23	0.31	0.50	0.66

O uso destes parâmetros d_{inter} e d_{intra} faz com que a probabilidade de ocorrência de *handoffs* para o *FCA* e para o *CBWL* seja praticamente nula, de modo que, nas simulações da próxima seção, os ganhos referentes a estes dois algoritmos são todos limitados por P_B .

4.6.2 - RESULTADOS OBTIDOS

Com a utilização do Segregação de Canais (aqui referido como *CS*) inexistente o conceito de *cluster*, no entanto, para fins de comparação, dizemos que cada algoritmo *CS* simulado nesta seção é “equivalente” a um determinado tamanho de *cluster*.

Isto deve-se ao fato de que os limiares SIR_B aqui utilizados são aqueles determinados na seção 4.3, os quais estão diretamente relacionados a tamanhos de *cluster* específicos.

Inicialmente é considerado o *CS* equivalente a $N = 3$, e os resultados obtidos são os seguintes:

Tabela 4.8 - Resultados do *CS* original para $N = 3$

ALGORITMO	CARGA P_B	CARGA P_F	<i>GOS</i>	LIMITE	GANHO OFERECIDO
<i>FCA</i>	6.3	> 6.3	6.3	P_B	0.00 %
<i>CS original</i>	10.1	7.3	7.3	P_F	15.87 %

O termo “original” aqui empregado está relacionado à utilização do critério de atualização dos valores de prioridade dos canais contido no artigo que propôs inicialmente o algoritmo *CS* [13]. Neste trabalho, aquele critério equivale às equações (3.3) e (3.4) - seção 3.3.3.1.

De acordo com os resultados da tabela 4.8, o ganho oferecido em termos de P_B é de aproximadamente 60.3%, valor este que deve ser comparado aos 12.9% conseguidos pelo *CBWL* (tabela 4.1). Entretanto, ainda em comparação ao *CBWL*, nota-se que a diferença entre as cargas-limite de *GOS* de P_B e de P_F é relativamente bem maior no caso do *CS*.

Tentando melhorar ainda mais o desempenho do *CS*, testamos uma nova proposta para o critério de atualização dos valores de prioridade dos canais [25].

A razão para a busca de um novo critério advém das limitações impostas pelo critério do artigo original, no qual, à medida que o denominador - valor de $n(i)$ - aumenta, o valor de $p(i)$ torna-se cada vez menos suscetível a mudanças.

Isto não apenas resulta em uma fraca adaptabilidade às variações no tráfego como também evidencia que os valores de prioridade iniciais tendem a polarizar os resultados do estado estacionário, tornando difícil recuperar na prática possíveis definições iniciais infelizes.

O critério alternativo então apresentado é do tipo *sliding window*, onde o cálculo dos novos valores de prioridade levam em consideração apenas uma determinada quantidade de medidas recentes da qualidade do sinal. Assim, para um determinado canal i :

$$p(i) \leftarrow \frac{[p(i).WIN] - old(i) + new(i)}{WIN}$$

onde: WIN = tamanho da janela, ou seja, é a quantidade de amostras levadas em consideração em cada cálculo.

$old(i)$ = usando WIN como limitante, é a mais antiga das amostras consideradas no cálculo de cada novo valor de $p(i)$.

$new(i)$ = amostra atual.

Na prática as amostras $old(i)$ e $new(i)$ representam a pontuação dada a cada medida de qualidade do sinal realizada para o canal i . Neste trabalho é atribuído $+1$ para as medidas bem-sucedidas (interferência abaixo do limiar tolerável), e -1 para as mal-sucedidas (interferência acima do limiar permitido).

Referindo-se a WIN como w , quatro valores são analisados (2, 3, 4 e 5), e os resultados obtidos são os apresentados na tabela 4.9. Naquela tabela os ganhos obtidos pelo critério *sliding window* não são animadores. Comparados aos do critério original estes ganhos são equivalentes em termos de P_F , porém piores em termos de P_B . Os resultados posteriores, no entanto, são melhores.

Tabela 4.9 - Resultados do CS com *sliding window* para $N = 3$

ALGORITMO	CARGA P_B	CARGA P_F	GOS	LIMITE	GANHO OFERECIDO
<i>sliding w=2</i>	9.6	7.2	7.2	P_F	14.29 %
<i>sliding w=3</i>	9.1	7.3	7.3	P_F	15.87 %
<i>sliding w=4</i>	9.8	7.3	7.3	P_F	15.87 %
<i>sliding w=5</i>	9.8	7.2	7.2	P_F	14.29 %

Semelhante ao realizado com o CBWL, para finalizar as simulações do CS equivalente a $N = 3$, utilizamos a estratégia de prioridade a *handoff*.

Para o caso específico do CS, o parâmetro h refere-se ao número de canais reservados para *handoff* no sistema como um todo, sendo que cada um destes canais também possui um valor de prioridade em cada célula. Além disto, a atualização destes valores é feita de acordo com o critério utilizado para a atualização dos valores dos canais não-reservados.

Usando para w apenas o valor de melhor desempenho ($w=4$), os resultados obtidos são os seguintes:

Tabela 4.10 - Resultados do CS com prioridade a *handoff* para $N = 3$

ALGORITMO	CARGA P_B	CARGA P_F	GOS	LIMITE	GANHO OFERECIDO
original $h=5$	9.1	8.2	8.2	P_F	30.16 %
original $h=6$	8.4	8.7	8.4	P_B	33.33 %
<i>sliding $h=4$</i>	9.2	8.1	8.1	P_F	28.57 %
<i>sliding $h=5$</i>	9.1	9.2	9.1	P_B	44.44 %

Os resultados da tabela 4.10 comprovam a maior eficiência oferecida pelo critério *sliding window*. Um fato observado é que o *sliding window* obtém seu melhor desempenho reservando um canal a menos para *handoff* do que a quantidade usada no melhor desempenho obtido pelo critério original.

Uma outra conclusão é que a prioridade a *handoff* é ainda mais necessária para os algoritmos que oferecem ganhos maiores em relação ao desempenho do esquema *FCA*.

A listagem do programa contendo o *CS* equivalente a $N = 3$ - para $w = 4$ e $h = 5$ - é apresentada no apêndice B deste trabalho.

Comparando-se os desempenhos obtidos nas simulações do *FCA*, do melhor *CBWL* e do *CS*, para o caso de $N = 3$, temos o seguinte:

Tabela 4.11 - Resultados dos três algoritmos para $N = 3$

ALGORITMO	<i>GOS</i>	GANHO PARA O <i>CBWL</i>	GANHO PARA O <i>FCA</i>
<i>FCA</i>	6.3	0.00 %	0.00 %
<i>CBWL</i> (5,6,5)	7.2	0.00 %	14.29 %
Melhor <i>CS</i> sem h	7.3	1.39 %	15.87 %
Melhor <i>CS</i> com h	9.1	26.39 %	44.44 %

Em seguida é considerado o *CS* equivalente a $N = 2$, e os resultados iniciais obtidos são os seguintes:

Tabela 4.12 - Resultados iniciais do *CS* para $N = 2$

ALGORITMO	CARGA P_B	CARGA P_F	<i>GOS</i>	LIMITE	GANHO OFERECIDO
<i>FCA</i>	10.7	> 10.7	10.7	P_B	0.00 %
<i>CS</i> original	15.0	10.6	10.6	P_F	- 0.93 %
<i>sliding</i> $w=2$	14.9	11.5	11.5	P_F	7.48 %
<i>sliding</i> $w=3$	15.3	11.4	11.4	P_F	6.54 %
<i>sliding</i> $w=4$	14.9	11.0	11.0	P_F	2.80 %
<i>sliding</i> $w=5$	14.7	11.0	11.0	P_F	2.80 %

De acordo com os resultados, o máximo ganho oferecido em termos de P_B é de aproximadamente 43%, valor este que deve ser comparado aos 9.6% conseguidos pelo *CBWL* (tabela 4.4). Nota-se no entanto que, novamente, as limitações em P_F reduzem bastante o ganho oferecido em termos de *GOS*.

Ao contrário do acontecido com o *CS* equivalente a $N = 3$, para $N = 2$ mesmo antes de aplicada a prioridade a *handoff* o critério *sliding window* já consegue desempenho melhor que o critério original.

É interessante, ainda, notar que para $N = 2$ o valor de w que proporcionou o melhor desempenho relativo foi $w = 3$ - para $N = 3$ o melhor havia sido $w = 4$.

Concluindo as simulações de *CS* equivalente a $N = 2$, utilizamos novamente a estratégia de prioridade a *handoff*. Usando para w apenas o valor de melhor desempenho relativo ($w=3$), os resultados obtidos são os seguintes:

Tabela 4.13 - Resultados do *CS* com prioridade a *handoff* para $N = 2$

ALGORITMO	CARGA P_B	CARGA P_F	<i>GOS</i>	LIMITE	GANHO OFERECIDO
original $h=7$	13.5	13.4	13.4	P_F	25.23 %
original $h=8$	13.3	> 13.3	13.3	P_B	24.30 %
<i>sliding</i> $h=5$	14.6	13.5	13.5	P_F	26.17 %
<i>sliding</i> $h=6$	14.5	14.7	14.5	P_B	35.51 %

Nota-se que, novamente, a maior eficiência do critério *sliding window* permite que seu melhor desempenho utilize um canal a menos para fins de *handoff* do que a quantidade reservada no melhor desempenho do critério original.

Em comparação aos resultados obtidos para *CS* equivalente a $N = 3$, também verifica-se que, novamente, os ganhos oferecidos para $N = 2$ são menores do que os oferecidos para $N = 3$.

A tabela 4.14 apresenta a comparação dos desempenhos obtidos nas simulações do *FCA*, do melhor *CBWL* e do *CS* para o caso de $N = 2$.

Tabela 4.14 - Resultados dos três algoritmos para $N = 2$

ALGORITMO	GOS	GANHO PARA O <i>CBWL</i>	GANHO PARA O <i>FCA</i>
<i>FCA</i>	10.7	0.00 %	0.00 %
<i>CBWL</i> (9,6,9)	11.4	0.00 %	6.54 %
Melhor <i>CS</i> sem <i>h</i>	11.5	0.88 %	7.48 %
Melhor <i>CS</i> com <i>h</i>	14.5	27.19 %	35.51 %

É interessante observar que os ganhos obtidos pelo algoritmo Segregação de Canais com relação ao *CBWL* não sofreram grandes variações com a modificação do valor do tamanho do *cluster* (N) - tabelas 4.11 e 4.14.

A principal conclusão de nossas simulações é a grande importância da utilização de uma estratégia de prioridade às solicitações de *handoff*. Os ganhos oferecidos por algoritmos *DCA* podem ser bastante melhorados com o uso deste tipo de estratégia.

5 - CONCLUSÕES E SUGESTÕES

5.1 - CONCLUSÕES

Este trabalho analisou o problema da alocação dos canais de tráfego em sistemas móveis celulares. Foram apresentados alguns exemplos de algoritmos já propostos, e dois deles - *CBWL* e Segregação de Canais - foram simulados em um sistema *PCS FDMA*. Os resultados obtidos mostraram a importância da utilização de alguma estratégia de prioridade para o atendimento das chamadas em situação de *handoff*.

As conclusões mais importantes deste trabalho são as seguintes:

i) Nada aparenta justificar a escolha por um algoritmo que utilize empréstimos de canais. Além de não evitar o pré-planejamento de frequências - e, em alguns casos, como no *CBWL*, até mesmo torná-lo mais complexo - os ganhos oferecidos são pouco significativos.

ii) Algoritmos *DCA* puros e de controle distribuído - tais como o Segregação de Canais - tendem a ser cada vez mais utilizados, não apenas pela relativa melhoria no desempenho mas também pela redução da complexidade durante o projeto dos sistemas. A única dificuldade para a implementação prática de um algoritmo desta categoria é o requerimento de que todos os terminais e todas as ERBs estejam aptos a acessar todos os canais de frequência disponíveis no sistema.

iii) Usando o *FCA* como referência é trivial oferecer ganhos em termos de probabilidade de bloqueio de novas chamadas (P_B). No entanto, quanto maior for este ganho, mais recomendável torna-se incluir alguma forma de prioridade às solicitações de *handoff*, do contrário o grau de serviço tenderá a ser bastante prejudicado.

5.2 - SUGESTÕES

Os resultados aqui obtidos podem ser encarados como embriões para a realização de futuras pesquisas na área. Se este for o caso, gostaríamos de deixar registradas algumas observações:

i) É evidente que o sistema celular utilizado em nossos programas pode ser enriquecido. Dentre as possíveis melhorias destacamos a migração para a filosofia *TDMA/FDMA* (já em uso nos sistemas digitais), a inclusão de desvanecimentos, a utilização de estrutura bi-dimensional e a inclusão de usuários em automóveis.

ii) Ainda com referência aos nossos programas, seria interessante considerar outros valores para alguns parâmetros do sistema, tais como o comprimento das células, o tempo médio de duração das chamadas e a constante de atenuação. Isto permitiria analisar as influências que cada um destes parâmetros exerce sobre os resultados obtidos.

iii) Não vemos grandes razões para que esquemas de empréstimo de canais venham a ser novamente simulados. No entanto, se isto acontecer, e se além disto, o *CBWL* for reconsiderado, as demais estratégias de melhoria já propostas (Troca de Canais, Rearranjo de Canais e Retorno Rápido de Canais) poderiam ser analisadas. Com relação à interface do programa, seria melhor separar os dados de bloqueio de novas chamadas para cada um dos tipos de usuário - os que podem e os que não podem usar canal tomado emprestado. Isto certamente tornaria as análises mais completas.

iv) Com relação ao Segregação de Canais temos três sugestões: executar simulações mais demoradas, a fim de verificar se o desempenho melhora ou não; comparar o algoritmo com outras estratégias *DCA* puras; e utilizar outros valores para o tamanho da janela no critério *sliding window*.

v) Tendo sido mostrada a importância de se dar a prioridade a *handoff*, torna-se interessante a investigação de estratégias alternativas à que foi usada neste trabalho. Reservar canais específicos para *handoff* talvez provoque perdas desnecessárias no grau de serviço que os algoritmos poderiam vir a oferecer em termos de P_B . Como atender simultaneamente aos graus de serviço de P_B e de P_F , mantendo-os no nível mais alto possível é uma boa questão a ser resolvida.

REFERÊNCIAS BIBLIOGRÁFICAS

- [1] V. H. Mac Donald; “*Advanced Mobile Phone Service: The Cellular Concept*”; *The Bell System Technical Journal*, vol 58, no 1, pp 15-41, janeiro de 1979.

- [2] W. C. Y. Lee; “*Mobile Cellular Telecommunications Systems*”; McGraw-Hill, 1990.

- [3] M. D. Yacoub; “*Foundations of Mobile Radio Engineering*”; CRC Press, 1993.

- [4] R. Beck & H. Panzer; “*Strategies for Handover and Dynamic Channel Allocation in Micro-Cellular Mobile Radio Systems*”; in *Proc. IEEE Vehicular Technology Conference*, pp 178-185, maio de 1989.

- [5] J. S. Engel & M. M. Peritsky; “*Statistically-Optimum Dynamic Server Assignment in Systems with Interfering Servers*”; *IEEE Transactions on Communications*, vol 21, no 11, pp 1287-1293, novembro de 1973.

- [6] T. J. Kahwa & N. D. Georganas; “*A Hybrid Channel Assignment Scheme in Large-Scale, Cellular-Structured Mobile Communication Systems*”; *IEEE Transactions on Communications*, vol 26, no 4, pp 432-438, abril de 1978.

- [7] S. M. Elnoubi et al; “*A New Frequency Channel Assignment Algorithm in High Capacity Mobile Communication Systems*”; *IEEE Transactions on Vehicular Technology*, vol 31, no 3, pp 125-131, agosto de 1982.

- [8] M. Zhang & T-S. P. Yum; “*Comparisons of Channel-Assignment Strategies in Cellular Mobile Telephone Systems*”; *IEEE Transactions on Vehicular Technology*, vol 38, no 4, pp 211-215, novembro de 1989.

- [9] H. Jiang & S. S. Rappaport; “*CBWL: A New Channel Assignment and Sharing Method for Cellular Communication Systems*”; *IEEE Transactions on Vehicular Technology*, vol 43, no 2, pp 313-322, maio de 1994.

- [10] S. W. Halpern; “*Reuse Partitioning in Cellular Systems*”; in *Proc. IEEE Vehicular Technology Conference*, pp 322-327, maio de 1983.

- [11] J. Zander; “*Generalized Reuse Partitioning in Cellular Mobile Radio*”; in *Proc. IEEE Vehicular Technology Conference*, pp 181-184, maio de 1993.

- [12] R. W. Nettleton & G. R. Schloemer; “*A High Capacity Assignment Method for Cellular Mobile Telephone Systems*”; in *Proc. IEEE Vehicular Technology Conference*, pp 359-367, maio de 1989.

- [13] Y. Furuya & Y. Akaiwa; “*Channel Segregation, A Distributed Adaptive Channel Allocation Scheme for Mobile Communication Systems*”; in *Proc. Second Nordic Seminar on Digital Land Mobile Radio Communications*, pp 311-315, outubro de 1986.

- [14] S-H. Oh & D-W. Tcha; “*Prioritized Channel Assignment in a Cellular Radio Network*”; *IEEE Transactions on Communications*, vol 40, no 7, pp 1259-1269, julho de 1992.

- [15] J. Tajima & K. Imamura; “*A Strategy for Flexible Channel Assignment in Mobile Communication Systems*”; *IEEE Transactions on Vehicular Technology*, vol 37, no 2, pp 92-103, maio de 1988.

- [16] B. Eklundh; “*Channel Utilization and Blocking Probability in a Cellular Mobile Telephone System with Directed Retry*”; *IEEE Transactions on Communications*, vol 34, no 4, pp 329-337, abril de 1986.

- [17] M. Frodigh; “*Radio Resource Allocation in Highway Micro Cellular Systems*”; *Ph.D. thesis, Royal Institute of Technology, Sweden*, novembro de 1993.

- [18] H. Jiang & S. S. Rappaport; “*A Channel Borrowing Scheme for TDMA Cellular Communication Systems*”; in *Proc. IEEE Vehicular Technology Conference*, pp 97-101, julho de 1995.

- [19] Y. Akaiwa & H. Andoh; “*Channel Segregation - A Self-Organized Dynamic Channel Allocation Method: Application to TDMA/FDMA Microcellular System*”; *IEEE Journal on Selected Areas in Communications*, vol 11, no 6, pp 949-954, agosto de 1993.

- [20] J. Sarnecki et al; “*Microcell Design Principles*”; *IEEE Communications Magazine*, pp 76-82, abril de 1993.

- [21] H. Jiang & S. S. Rappaport; “*CBWL with Channel Fast Returning: A Channel Sharing and Use Strategy for Cellular Communications*”; in *Proc. Second International Conference on Universal Personal Communications*, pp 471-475, outubro de 1993.

- [22] H. Jiang & S. S. Rappaport; “*Prioritized Channel Borrowing Without Locking: A Channel Sharing Strategy for Cellular Communications*”; in *Proc. IEEE Global Communications Conference*, pp 276-280, novembro de 1993.
- [23] D. C. Cox & D. O. Reudink; “*The Behavior of Dynamic Channel-Assignment Mobile Communication System as a Function of Numbers of Radio Channels*”; *IEEE Transactions on Communications*, vol 20, no 6, pp 471-479, junho de 1972.
- [24] I. Mecler & L. P. Maia; “*Programação e Lógica com Turbo Pascal*”; Editora Campus, 1989.
- [25] M. Frullone & P. Grazioso; “*Dynamic Channel Allocation for ATDMA*”; in *Proc. RACE Telecommunications Summit*, pp 299-303, novembro de 1995.

APÊNDICE A

O MELHOR *CBWL* PARA $N = 2$


```

program CBWL_WITH_HANDOFF_PRIORITY;
(**)
(* last revision: 1996/09/09 *)
(* programmers: Sergio Luiz Noronha Fraiha *)
(* Fabricio Mello Rodrigues da Silva *)
(**)
(*===== Declarations =====*)
Uses Crt, Graph;
const
  CALLINTERV = 150; (* Mean time between new call requests for each user. *)
                      (* Unity is equivalent to 1 second -- 486DX2-66 MHz *)
  ALFA = 3.5; (* Propagation exponent *)
  FLOORBLOC = 37.4; (* Minimum SIR for new calls (SIRb) -- 15.7 dB *)
  POPULATION = 384; (* Total number of users *)
  CLUSTER = 2; (* Number of cells in a cluster (N) *)
  RESOURCES = 48; (* Total number of channels (C) *)
  STATIONS = 12; (* Total number of cells *)
  SHARED = 9; (* Number of shared channels at each side of cell (l) *)
  BRWRADIUS = 0.1; (* Borrowing radius (r) *)
  STOPTIME = 3600; (* Maximum time for simulation -- 60 MINUTES *)
  HANDRESERV = 1; (* Number of channels reserved for handoff per cell (h) *)

type
  (* A channel is defined by an integer number (its number in the sequence of *)
  (* existing channels) and one cell in which it is active. *)
  channel = record
    num, cel : integer;
  end;

  (* The terminal type is the simulation's fundamental data structure. *)
  (* Each user is defined by: *)
  (* 1. A position, a real number between 0 and 12, the width of the system; *)
  (* 2. An old position, before execution of procedure MoveUsers; *)
  (* 3. A velocity, distributed according to procedure ChangeVelocity; *)
  (* 4. A desire to speak, boolean; *)
  (* 5. A channel presently using; *)
  (* 6. A countdown stopwatch, showing how longer the present call (if any) *)
  (* will last. = Zero if no call is in progress; *)
  (* 7. An indication whether channel borrowing was tried; *)
  (* 8. The cell from which the channel was borrowed, if borrowed occurred; *)
  (* 9. The user type. Type 1 can borrow a channel. Type 2 can not borrow; *)
  (* 10. A countdown stopwatch, showing the rest time for a new call request; *)
  (* 11. An indication whether the channel being used is a handoff channel. *)
  terminal = record
    posi : real; (* position *)
    oldposi : real; (* old position *)
    velo : real; (* velocity *)
    rqst : boolean; (* request for channel *)
    chnF : channel; (* present channel *)
    ttgo : integer; (* time-to-go: countdown to end call *)
    borrowtried : boolean; (* borrowing indication *)
    borrowcell : integer; (* borrowing cell *)
    usertype : integer; (* borrowing permission indicator *)
    timenewcall : integer; (* countdown for a new call request *)
    handoffchn : boolean; (* handoff channel indicator *)
  end;

var
  Time : integer; (* keeps track of elapsed simutime *)
  User : array[1..POPULATION] of terminal; (* collection of subscribers *)
  CIU : array[1..STATIONS,1..RESOURCES] of boolean;
  (* marks Channels In Use or not at each cell *)
  G : array[1..POPULATION,1..STATIONS] of real; (* link-gain matrix *)
  RCU : array[1..STATIONS] of integer; (* number of Regular Channels in Use*)

```

```

HCU : array[1..STATIONS] of integer; (* number of Handoff Channels in Use*)
Stats : record
    call_tries: integer; (* total number of new call requests *)
    call_fails: integer; (* total number of new call denials *)
    inte_tries: integer; (* total number of inter handoffs tries *)
    inte_fails: integer; (* total number of inter handoffs fails *)
    intr_tries: integer; (* total number of intra handoffs tries *)
    intr_fails: integer; (* total number of intra handoffs fails *)
end;
NULLCHANNEL : channel; (* .num = .cel = 0 *)
DISPLACEMENT : integer; (* = -1|0|+1; used by function ChannelChoice *)
(* to report globally if the channel comes *)
(* from Cellof(n) or if it was borrowed from *)
(* the left (-1) or from the right (+1). *)
NEWCALL : boolean; (* states whether the call is a new or an ongoing one *)
(*===== End of Declarations =====*)

(*===== Auxiliary Procedures & Functions =====*)
procedure ChangeVelocity(m:integer);
(**)
(* Sets a new velocity to User[m]. By altering this procedure, we can test *)
(* the assignment algorithm under different mobile behaviors. *)
(**)
(* Presently, the probability density functions are discrete: *)
(* - if user is talking: .10p x (-stp) + .80p x (0) + .10p x (stp) *)
(* - if user is not talking: .33p x (-stp) + .33p x (0) + .33p x (stp) *)
(* where stp is the user step length. The intent is to actually mimic *)
(* pedestrian users. When they speak, they will most probably stop walking, *)
(* hence the .80 probability on zero velocity. When they are disconnected, *)
(* they may walk either way or be still, all three velocities having *)
(* barely the same probability. *)
(**)
const
    stp = 0.0015432; (* user step length *)
(* velocity of user is equal to 5 kilometers per hour *)
(* diameter of cell is 900 meters, and this is equivalent to the space unit *)
var
    r, v : real; (* random num between 0 and 1; new velocity *)
begin
    r := Random;
    if User[m].chnP.num > 0 then (* if user is talking... *)
        begin
            if r <= 0.1 then v := -stp else
            if r < 0.9 then v := 0 else
            v := stp;
        end
    else (* if user is not talking... *)
        begin
            if r <= 0.33 then v := -stp else
            if r < 0.67 then v := 0 else
            v := stp;
        end;
    User[m].velo := v;
end; (* procedure ChangeVelocity(m) *)

function Expttgo : integer;
(**)
(* Exponential Distributed Random Integer Number. When an user is given a *)
(* channel, his time-to-go (ttgo) variable is set to such a number. *)
(* It measures the total time he (she) will hold the channel. *)
(**)
const
    mean = 120; (* mean time of call duration -- 2 MINUTES *)
var

```

```

numel, nume2 : real; (* scrap vars *)
begin
  numel := Random;      (* random number between 0 and 1 *)
  if numel = 0 then
    nume2 := 0.0000000001 (* to avoid overflow *)
  else
    nume2 := numel;
  Expttgo := Round(-mean*ln(nume2));
end; (* function Expttgo *)

function Expnewcall : integer;
(**)
(* Exponential Distributed Random Integer Number. *)
(* It measures the total time for a new call request. *)
(**)
var
  numel, nume2 : real; (* scrap vars *)
begin
  numel := Random;      (* random number between 0 and 1 *)
  if numel = 0 then
    nume2 := 0.0000000001 (* to avoid overflow *)
  else
    nume2 := numel;
  Expnewcall := Round(-CALLINTERV*ln(nume2));
end; (* function Expnewcall *)

function Cellof(m:integer) : integer;
(**)
(* Returns the cell number (1-12) where User[m] is. *)
(**)
begin
  (* wrap-around the user, if necessary *)
  if User[m].posi >= STATIONS then User[m].posi := User[m].posi - STATIONS;
  if User[m].posi < 0 then User[m].posi := User[m].posi + STATIONS;
  Cellof := Trunc(User[m].posi) + 1;
end; (* function Cellof *)

function Dist(i,j:integer) : real;
(**)
(* Returns the distance between user i and ERB j. *)
(**)
var
  d : real; (* scrap var *)
begin
  d := Abs(User[i].posi - (j-0.5)); (* ERB is at the center of the cell *)
  if d > STATIONS/2 then
    Dist := STATIONS - d (* because wrap-around is used *)
  else
    Dist := d;
end; (* function Dist *)

function FirstChannelOf(c:integer) : integer;
(**)
(* Returns the number of the first channel owned by cell c. *)
(**)
begin
  FirstChannelOf := Trunc(((c-1) MOD CLUSTER)*RESOURCES/CLUSTER + 1);
end; (* Function FirstChannelOf *)

function LastChannelOf(c:integer) : integer;
(**)
(* Returns the number of the last channel owned by cell c. *)
(**)
begin

```

```

LastChannelOf := Trunc(FirstChannelOf(c) + RESOURCES/CLUSTER - 1);
end; (* Function LastChannelOf *)

function ChannelChoice(u:integer) : integer;
(**)
(* Channel assignment module. This is the main intracellular channel *)
(* assignment algorithm, which means: this is the algorithm that each base *)
(* station uses to allocate channels to requests by users inside its cell. *)
(* To test the effectiveness of a new assignment algorithm, try it here. *)
(**)
Function NormalFirstFree(cell,group:integer) : integer;
(**)
(* The search is performed from left to right. *)
(* Returns the number of the first free channel at cell, in the *)
(* specified group. Group must be 0, 1, 2 or 3. 0 is the standard *)
(* group of channels in the cell, those which can not be lent. *)
(* 1 and 2, the channels which can be lent to the left and to the *)
(* right, respectively. Group 3 represents the channels reserved for *)
(* handoffs. *)
(**)
var
  n_chn,n_last : integer; (* normal channel counter, last channel *)
  n_foundfree : boolean; (* test *)
begin
  case group of
    0 : begin
      n_chn := FirstChannelOf(cell)+SHARED-1;
      n_last := LastChannelOf(cell)-SHARED-HANDRESERV;
    end;
    1 : begin
      n_chn := FirstChannelOf(cell)-1;
      n_last := n_chn+SHARED;
    end;
    2 : begin
      n_last := LastChannelOf(cell);
      n_chn := n_last-SHARED;
    end;
    3 : begin
      n_last := LastChannelOf(cell)-SHARED;
      n_chn := n_last-HANDRESERV;
    end
  else (* invalid group *)
    Exit;
  end; (* case *)
  n_foundfree := FALSE; (* assume free n_chn will not be found *)
  repeat
    n_chn := n_chn + 1;
    n_foundfree := not CIU[Cellof(u),n_chn]; (* channel is not in use *)
  until (n_foundfree) or (n_chn = n_last);
  if n_foundfree then
    NormalFirstFree := n_chn (* allocation will be possible *)
  else
    NormalFirstFree := 0;
  end; (* Function NormalFirstFree *)
(**)
Function BorrowFirstFree(cell,group:integer) : integer;
(**)
(* The search is performed from right to left. *)
(* Returns the number of the first free channel at cell, in the *)
(* specified group. Group must be 1 or 2. 1 is the group of *)
(* channels which can be lent by the right cell, and 2 the *)
(* channels which can be lent by the left cell. *)
(**)
var

```

```

b_chn,b_last : integer; (* borrow channel counter, last channel of *)
b_foundfree : boolean;  (* test *)
begin
  case group of
    1 : begin
      b_last := FirstChannelOf(cell);
      b_chn := b_last+SHARED;
    end;
    2 : begin
      b_chn := LastChannelOf(cell)+1;
      b_last := b_chn-SHARED;
    end;
    else (* invalid group *)
      Exit;
  end; (* case *)
  b_foundfree := FALSE;      (* assume free b_chn will not be found *)
  repeat
    b_chn := b_chn - 1;
    b_foundfree := not CIU[cell,b_chn]; (* channel is not in use *)
  until (b_foundfree) or (b_chn = b_last);
  if b_foundfree then
    BorrowFirstFree := b_chn (* borrowing will be possible *)
  else
    BorrowFirstFree := 0;
  end; (* Function BorrowFirstFree *)
  (**)
Function BorrowedFromLeft : integer;
(**)
(* Returns a borrowed channel from Cellof(u)-1; =0 if fails. *)
(**)
var
  chn : integer; (* scrap channel var *)
begin
  chn := 0; (* assume failure *)
  DISPLACEMENT := -1; (* borrow from cell on the left *)
  if Cellof(u) > 1 then (* if not at left border... *)
    chn := BorrowFirstFree((Cellof(u))-1,2)
  else
    chn := BorrowFirstFree(STATIONS,2);
  BorrowedFromLeft := chn;
end; (* Function BorrowedFromLeft *)
(**)
Function BorrowedFromRight : integer;
(**)
(* Returns a borrowed channel from Cellof(u)+1; =0 if fails. *)
(**)
var
  chn : integer; (* scrap channel var *)
begin
  chn := 0; (* assume failure *)
  DISPLACEMENT := 1; (* borrow from cell on the right *)
  if Cellof(u) < STATIONS then (* if not at right border... *)
    chn := BorrowFirstFree((Cellof(u))+1,1)
  else
    chn := BorrowFirstFree(1,1);
  BorrowedFromRight := chn;
end; (* Function BorrowedFromRight *)
(**)
Function Borrowed : integer;
(**)
(* Returns a borrowed channel from Cellof(u)-1 or Cellof(u)+1; *)
(* =0 if fails. Half of the time, it tries to borrow first from *)
(* cell at the left, half from the one at the right. *)
(**)

```

```

var
  chan : integer; (* scrap channel var *)
begin
  chan := 0;      (* assume failure *)
  if Random < 0.5 then
    begin
      chan := BorrowedFromLeft;      (* try first at the left; *)
      if chan = 0 then
        chan := BorrowedFromRight; (* if fails, try at the right *)
      end
    end
  else (* OR... *)
    begin
      chan := BorrowedFromRight;      (* try first at the right; *)
      if chan = 0 then
        chan := BorrowedFromLeft;      (* if fails, try at the left *)
      end
    end;
  Borrowed := chan;
end; (* Function Borrowed *)

(**)
begin
  DISPLACEMENT := 0; (* assume channel will not be a borrowed one *)
  ChannelChoice := 0; (* assume failure *)
  Case User[u].usertype of
    1 : (* user can use a borrowed channel, if necessary *)
      if NormalFirstFree(Cellof(u),0) > 0 then (* try first at group 0 *)
        ChannelChoice := NormalFirstFree(Cellof(u),0)
      else (* if fails... *)
        begin
          if Random < 0.5 then (* half of time... *)
            begin
              if NormalFirstFree(Cellof(u),1) > 0 then
                (* try then at group 1 *)
                ChannelChoice := NormalFirstFree(Cellof(u),1)
              else (* if still fails... *)
                begin
                  if NormalFirstFree(Cellof(u),2) > 0 then
                    (* try at group 2 *)
                    ChannelChoice := NormalFirstFree(Cellof(u),2)
                  else (* if fails again... *)
                    begin
                      (* try a borrowed one *)
                      ChannelChoice := Borrowed;
                      User[u].borrowtried := TRUE;
                    end;
                  end;
                end;
              end;
            end
          else (* OR... *)
            begin
              if NormalFirstFree(Cellof(u),2) > 0 then
                (* try then at group 2 *)
                ChannelChoice := NormalFirstFree(Cellof(u),2)
              else (* if still fails... *)
                begin
                  if NormalFirstFree(Cellof(u),1) > 0 then
                    (* try at group 1 *)
                    ChannelChoice := NormalFirstFree(Cellof(u),1)
                  else (* if fails again... *)
                    begin
                      (* try a borrowed one *)
                      ChannelChoice := Borrowed;
                      User[u].borrowtried := TRUE;
                    end;
                  end;
                end;
              end;
            end;
          end;
        end;
      end;
    end;
  end;
end;
end;

```

```

end;
2 : (* user can not use a borrowed channel *)
if (NEWCALL = FALSE) and (NormalFirstFree(Cellof(u),3) > 0) then
  (* if ongoing call try first at group 3 -- handoff attempt *)
  begin
    ChannelChoice := NormalFirstFree(Cellof(u),3); (* handoff channel *)
    User[u].handoffchn := TRUE;
  end
else (* new call or group 3 failed *)
  begin
    if NormalFirstFree(Cellof(u),0) > 0 then
      (* try then at group 0 *)
      ChannelChoice := NormalFirstFree(Cellof(u),0)
    else (* if fails... *)
      begin
        if Random < 0.5 then (* half of time... *)
          begin
            if NormalFirstFree(Cellof(u),1) > 0 then
              (* try then at group 1 *)
              ChannelChoice := NormalFirstFree(Cellof(u),1)
            else (* if still fails... *)
              (* try at group 2 *)
              ChannelChoice := NormalFirstFree(Cellof(u),2);
            end
          end
        else (* OR... *)
          begin
            if NormalFirstFree(Cellof(u),2) > 0 then
              (* try then at group 2 *)
              ChannelChoice := NormalFirstFree(Cellof(u),2)
            else (* if still fails... *)
              (* try at group 1 *)
              ChannelChoice := NormalFirstFree(Cellof(u),1);
            end
          end
        end;
      end
    end;
  end;
end; (* case *)
end; (* function ChannelChoice *)

function Power(x,y:real) : real;
(**)
(* Returns x to the y-th power. *)
(**)
begin
  if x > 0 then
    Power := exp(y*ln(x))
  else
    Power := 0.0000000001; (* to avoid overflow in procedure Update_G *)
  end;
end; (* function Power *)

procedure Update_G;
(**)
(* Updates the link-gain matrix. *)
(* This procedure is here to increase execution time to a desirable level *)
(**)
var
  m,n : integer; (* user var, ERB var *)
begin
  for m := 1 to POPULATION do
    for n := 1 to STATIONS do
      begin
        G[m,n] := 1/Power(Dist(m,n),ALFA);
      end;
    end;
  end;
end; (* procedure Update_G *)

```

```

procedure SeizeChn(chn,m,cell:integer);
(**)
(* Marks channel chn in cell as busy (=TRUE). *)
(**)
begin
  TextBackground(White);
  TextColor(Blue);
  GotoXY(14,24);
  Write(TIME:4,' : ');
  GotoXY(20,24);
  Write('Channel ');
  GotoXY(28,24);
  Write(chn:2);
  GotoXY(30,24);
  Write(' assigned to user ');
  GotoXY(48,24);
  Write(m:3);
  GotoXY(51,24);
  Write(' in cell ');
  GotoXY(60,24);
  Write(cell:2);
  TextColor(White);
  TextBackground(Black);
  CIU[cell,chn] := TRUE;
  if User[m].handoffchn = FALSE then      (* regular channel *)
    RCU[cell] := RCU[cell] + 1
  else
    HCU[cell] := HCU[cell] + 1;
end; (* procedure SeizeChn *)

procedure ReleaseChn(chn,m,cell:integer);
(**)
(* Marks channel chn in cell as free (=FALSE). *)
(**)
begin
  TextBackground(White);
  TextColor(Blue);
  GotoXY(14,22);
  Write(TIME:4,' : ');
  GotoXY(20,22);
  Write('Channel ');
  GotoXY(28,22);
  Write(chn:2);
  GotoXY(30,22);
  Write(' released by user ');
  GotoXY(48,22);
  Write(m:3);
  GotoXY(51,22);
  Write(' in cell ');
  GotoXY(60,22);
  Write(cell:2);
  TextColor(White);
  TextBackground(Black);
  CIU[cell,chn] := FALSE;
  if User[m].handoffchn = FALSE then      (* regular channel *)
    RCU[cell] := RCU[cell] - 1
  else
    begin
      User[m].handoffchn := FALSE;
      HCU[cell] := HCU[cell] - 1;
    end;
end; (* procedure ReleaseChn *)
(*===== End of Auxiliary Procedures & Functions =====*)

```



```

(*===== Main Procedures =====*)
procedure Initialize;
(**)
(* Sets up initial position & telecom status of all users. *)
(**)
var
  n : integer; (* loop *)
begin
  Randomize;
  TIME := 0;
  for n := 1 to POPULATION do with User[n] do
    begin
      posi := 12*Random;          (* uniform initial distribution *)
      oldposi := 0;               (* user was not in any place *)
      ChangeVelocity(n);          (* initial velocity *)
      rqst := FALSE;              (* no user wants to talk *)
      chnP.num := 0;              (* no user starts off talking *)
      ttgo := 0;                  (* time-to-go is zeroed. *)
      borrowtried := FALSE;       (* no channel borrowing was tried *)
      usertype := 2;              (* assume user can not borrow a channel *)
      timenewcall := Expnewcall; (* time for a new call request *)
      handoffchn := FALSE;        (* no handoff occurred *)
    end;
    FillChar(CIU,SizeOf(CIU),0); (* no channel in use at any cell *)
    FillChar(RCU,SizeOf(RCU),0);
    FillChar(HCU,SizeOf(HCU),0);
    Stats.call_tries := 0;        (* reset statistics *)
    Stats.call_fails := 0;
    Stats.inte_tries := 0;
    Stats.inte_fails := 0;
    Stats.intr_tries := 0;
    Stats.intr_fails := 0;
    NULLCHANNEL.num := 0;
    NULLCHANNEL.cel := 0;
    ClrScr;
  end; (* procedure Initialize *)

procedure MoveUsers;
(**)
(* Updates the position of each user according to the velocity; changes this *)
(* velocity randomly; updates each user's desire to speak (rqst). *)
(**)
const
  VeloChangeProb = 0.1; (* probability of changing velocity *)
var
  n : integer; (* loop *)
begin
  for n := 1 to POPULATION do with User[n] do
    begin
      oldposi := posi;          (* save previous position *)
      posi := posi + velo;      (* move user *)
      if (posi >= 0) then
        begin
          if (usertype = 1) and (Dist(n,Cellof(n)) > BRWRADIUS) then
            begin
              usertype := 2; (* user now can not use a borrowed channel *)
              if (chnP.num > 0) and (borrowtried) then
                (* if a borrowed channel is being used, then an *)
                (* intracellular handoff must be tried *)
                begin
                  borrowtried := FALSE;
                  chnP.cel := borrowcell; (* cell which lent the channel *)
                  borrowcell := 0;
                  ReleaseChn(chnP.num,n,chnP.cel);

```

```

        Stats.intr_tries := Stats.intr_tries + 1;
        chnP.cel := Cellof(n);
        NEWCALL := FALSE; (* ongoing call *)
        chnP.num := ChannelChoice(n);
        if chnP.num = 0 then (* handoff failed *)
            Stats.intr_fails := Stats.intr_fails + 1
        else (* handoff succeeded *)
            SeizeChn(chnP.num,n,chnP.cel);
        end;
    end;
end;
if Trunc(posi) > Trunc(oldposi) then
    (* user moved to the right neighbor cell *)
    begin
        (* wrap-around, if necessary *)
        if posi >= STATIONS then posi := posi - STATIONS;
        if chnP.num > 0 then
            (* if user is talking, then an intercellular *)
            (* handoff must be tried *)
            begin
                ReleaseChn(chnP.num,n,chnP.cel);
                Stats.inte_tries := Stats.inte_tries + 1;
                chnP.cel := Cellof(n);
                NEWCALL := FALSE; (* ongoing call *)
                chnP.num := ChannelChoice(n);
                if chnP.num = 0 then (* handoff failed *)
                    Stats.inte_fails := Stats.inte_fails + 1
                else (* handoff succeeded *)
                    SeizeChn(chnP.num,n,chnP.cel);
                end;
            end
        end
    end
else
    if Trunc(posi) < Trunc(oldposi) then
        (* user moved to the left neighbor cell *)
        begin
            if chnP.num > 0 then
                (* if user is talking, then an intercellular *)
                (* handoff must be tried *)
                begin
                    ReleaseChn(chnP.num,n,chnP.cel);
                    Stats.inte_tries := Stats.inte_tries + 1;
                    chnP.cel := Cellof(n);
                    NEWCALL := FALSE; (* ongoing call *)
                    chnP.num := ChannelChoice(n);
                    if chnP.num = 0 then (* handoff failed *)
                        Stats.inte_fails := Stats.inte_fails + 1
                    else (* handoff succeeded *)
                        SeizeChn(chnP.num,n,chnP.cel);
                    end;
                end
            end
        end
    end;
end;
if posi < 0 then
    begin
        posi := posi + STATIONS; (* wrap-around *)
        if chnP.num > 0 then
            (* if user is talking, then an intercellular *)
            (* handoff must be tried *)
            begin
                ReleaseChn(chnP.num,n,chnP.cel);
                Stats.inte_tries := Stats.inte_tries + 1;
                chnP.cel := Cellof(n);
                NEWCALL := FALSE; (* ongoing call *)
                chnP.num := ChannelChoice(n);
                if chnP.num = 0 then (* handoff failed *)
                    Stats.inte_fails := Stats.inte_fails + 1
                end
            end
        end
    end
end

```

```

        else      (* handoff succeeded *)
            SeizeChn(chnP.num,n,chnP.cel);
        end;
    end;
    if Random < VeloChangeProb then (* eventually, change the velocity *)
        ChangeVelocity(n);
    if timenewcall > 0 then
        timenewcall := timenewcall - 1
    else      (* time for a new exponential generation *)
        begin
            timenewcall := Expnewcall;
            if chnP.num = 0 then rqst := TRUE; (* new request, if not talking *)
        end;
    end; (* for *)
    Update_G; (* update the link-gain matrix *)
    (* this procedure increases execution time to a desirable level *)
end; (* procedure MoveUsers *)

```

```

procedure ServeUsers;
(**)
(* Main telecom administration task. For each user there is either a channel *)
(* assignment trial (if the user's request state is TRUE), or a continuation *)
(* of his (her) countdown (if the user is already speaking). *)
(**)
var
    n : integer; (* users counter *)
begin
    n := 0;
    repeat
        n := n + 1;
        with User[n] do
            begin
                if Dist(n,Cellof(n)) <= BRWRADIUS then
                    usertype := 1          (* user can borrow a channel *)
                else
                    usertype := 2;          (* user can not borrow a channel *)
                if rqst then                (* if user wants to talk *)
                    begin
                        chnP.cel := Cellof(n);
                        NEWCALL := TRUE;
                        chnP.num := ChannelChoice(n);
                        Stats.call_tries := Stats.call_tries + 1;
                        if chnP.num = 0 then (* blocking occurred *)
                            begin
                                Stats.call_fails := Stats.call_fails + 1;
                                rqst := FALSE;
                                if (borrowtried) then borrowtried := FALSE;
                            end
                        else (* allocation succeeded *)
                            begin
                                if (Cellof(n) = 1) and (DISPLACEMENT = -1) then
                                    chnP.cel := STATIONS else
                                    if (Cellof(n) = STATIONS) and (DISPLACEMENT = 1) then
                                        chnP.cel := 1 else
                                        chnP.cel := Cellof(n) + DISPLACEMENT;
                                    if (borrowtried) then borrowcell := chnP.cel;
                                    SeizeChn(chnP.num,n,chnP.cel);
                                    rqst := FALSE;
                                    ttgo := Expttgo;
                                end;
                            end
                        end
                    end
                else (* if user does not want to talk: *)
                    if ttgo > 0 then (* see if he (she) IS talking *)
                        ttgo := ttgo - 1 (* if so, continue countdown *)
                    end
                end
            end
        end
    until n = MAXUSERS
end;

```

```

else
  begin
    if chnP.num > 0 then (* countdown ends: release channel *)
      begin
        if (borrowtried) then
          begin
            borrowtried := FALSE;
            chnP.cel := borrowcell;
            borrowcell := 0;
          end;
          ReleaseChn(chnP.num,n,chnP.cel);
          chnP := NULLCHANNEL;
        end; (* if *)
      end; (* else *)
    end;
  until (n = POPULATION);
end; (* procedure ServeUsers *)

procedure DisplayData;
(**)
(* Displays: elapsed simutime, accumulated number of new calls and handoff *)
(* requests, accumulated number of denials, rates of new call blocks and *)
(* handoff fails, and number of busy channels in each cell. *)
(**)
const
  NET = 15; (* The display line where the system is depicted. *)
var
  m, n : integer; (* scrap var, loop var *)
  call_block : real; (* rate of new calls blocking *)
  inte_block : real; (* rate of intercellular handoff fails *)
  intr_block : real; (* rate of intracellular handoff fails *)
begin
  TextColor(White);
  GotoXY(1,NET-14);
  WRITE('Channel Borrowing Without Locking');
  GotoXY(66,NET-14);
  WRITE('SIMUTIME');
  GotoXY(1,NET-13);
  WRITE('with ',HANDRESERV:1,' channel(s) reserved for handoff in each cell');
  GotoXY(68,NET-12);
  WRITE(TIME:4);
  GotoXY(1,NET-10);
  WRITE('CALLINIC',' ','CALLDENY',' ','CALLBLOCRATE');
  GotoXY(52,NET-9);
  WRITE('SIR threshold: ',10*ln(FLOORBLOC)/ln(10):4:1,' dB'); (* SIRb *)
  GotoXY(1,NET-8);
  if Stats.call_tries > 0 then (* to avoid overflow *)
    call_block := (Stats.call_fails / Stats.call_tries) * 100
  else
    call_block := 0;
  with Stats do
    WRITE(' ',call_tries:5,' ',call_fails:5,' ',call_block:6:2,' %');
    GotoXY(41,NET-8);
    WRITE('total number of channels: ',RESOURCES:4); (* C *)
    GotoXY(50,NET-7);
    WRITE('size of cluster: ',CLUSTER:4); (* N *)
    GotoXY(41,NET-6);
    WRITE('shared channels per cell: ',2*SHARED:4); (* 2*1 *)
    GotoXY(1,NET-5);
    WRITE('HANDINIC',' ','HANDDENY',' ','HANDFAILRATE');
    GotoXY(44,NET-5);
    WRITE('total number of users: ',POPULATION:4);
    GotoXY(42,NET-4);
    WRITE('mean time between calls: ',(CALLINTERV/60):4:1,' mi');

```

```

GotoXY(1,NET-3);
if Stats.inte_tries > 0 then (* to avoid overflow *)
    inte_block := (Stats.inte_fails / Stats.inte_tries) * 100
else
    inte_block := 0;
with Stats do
    WRITE(' ',inte_tries:5,'      ',inte_fails:5,'      ',inte_block:6:2,' %');
GotoXY(33,NET-3);
WRITELN('intercell');
GotoXY(1,NET-2);
if Stats.intr_tries > 0 then (* to avoid overflow *)
    intr_block := (Stats.intr_fails / Stats.intr_tries) * 100
else
    intr_block := 0;
with Stats do
    WRITE(' ',intr_tries:5,'      ',intr_fails:5,'      ',intr_block:6:2,' %');
GotoXY(33,NET-2);
WRITELN('intracell');
WRITELN;
(**)
(* The next loop depicts the net: '+'=ERB, '|'=Frontier. *)
for n := 1 to 73 do
    begin
        if (n+2)/6 = Int((n+2)/6) then WRITE('+') else
        if (n+5)/6 = Int((n+5)/6) then WRITE('|') else
        WRITE(' ');
    end;
(**)
(* The next loop depicts the users. *)
for n := 1 to POPULATION do with User[n] do
    begin
        TextColor(LightBlue);(* BLUE users are not talking and do not want to*)
        if ttgo > 0 then
            TextColor(Green); (* GREEN users are talking now *)
            GotoXY(Trunc(6*posi)+1,NET);
            WRITE('!!');
        end; (* for *)
    TextColor(White);
    WRITELN;
    WRITELN;
    (**)
    (* The next loop depicts the cells numbers *)
    for n := 1 to 73 do
        begin
            case n of
                1: WRITE('.');
                4: WRITE('1');
                7: WRITE('.');
                10: WRITE('2');
                13: WRITE('.');
                16: WRITE('3');
                19: WRITE('.');
                22: WRITE('4');
                25: WRITE('.');
                28: WRITE('5');
                31: WRITE('.');
                34: WRITE('6');
                37: WRITE('.');
                40: WRITE('7');
                43: WRITE('.');
                46: WRITE('8');
                49: WRITE('.');
                52: WRITE('9');
                55: WRITE('.');

```

```

        57: WRITE('1 0');
        59: WRITE('.');
        61: WRITE('1 1');
        63: WRITE('.');
        65: WRITE('1 2');
        67: WRITE('.');
    else WRITE(' ');
    end; (* case *)
end; (* for *)
Writeln;
Writeln;
(**)
(* The next loop depicts the quantities of Regular Channels in Use *)
(* in each cell. *)
for n := 1 to 56 do
begin
    if (n < 47) and ((n+2)/4 = Int((n+2)/4)) then
    begin
        m := Trunc((n+2)/4);
        if RCU[m] = (RESOURCES/CLUSTER) - HANDRESERV then
            TextColor(Red) (* all regular channels of the cell are busy *)
        else
            TextColor(White);
            WRITE(RCU[m]:3);
        end
    else
        WRITE(' ');
    end;
    TextColor(White);
    (**)
    (* The next loop depicts the quantities of Handoff Channels in Use *)
    (* in each cell. *)
    for n := 1 to 73 do
    begin
        if (n < 47) and ((n+2)/4 = Int((n+2)/4)) then
        begin
            m := Trunc((n+2)/4);
            if HCU[m] = HANDRESERV then
                TextColor(Red) (* all handoff channels of the cell are busy *)
            else
                TextColor(White);
                WRITE(HCU[m]:3);
            end
        else
            WRITE(' ');
        end;
        TextColor(White);
    end; (* procedure DisplayData *)
    (*===== End of Main Procedures =====*)

    (*===== Main Routine =====*)
    BEGIN
        Initialize;
        repeat
            if TIME < STOPTIME then
            begin
                TIME := TIME + 1;          (* For each new instant in simutime: *)
                MoveUsers;                 (* 1. Move the users *)
                ServeUsers;                 (* 2. Try to serve requests for channels *)
            end;
            DisplayData;                   (* 3. Update system display *)
        until false;
    END.
    (*===== End of Main Routine =====*)

```

APÊNDICE B

O MELHOR CS EQUIVALENTE A $N = 3$

```

program SEGREGATION_SLIDING_WINDOW_WITH_HANDOFF_PRIORITY;
(**)
(* last revision: 1996/09/10 *)
(* programmers: Sergio Luiz Noronha Fraiha *)
(* Fabricio Mello Rodrigues da Silva *)
(**)
(*===== Declarations =====*)
Uses Crt, Graph;
const
  CALLINTERV = 180; (* Mean time between new call requests for each user. *)
                      (* Unity is equivalent to 1 second -- 486DX2-66 MHz *)
  ALFA = 3.5; (* Propagation exponent *)
  FLOORBLOC = 203.8; (* Minimum SIR for new calls (SIRb) -- 23.1 dB *)
  FLOORHAND = 64.6; (* Minimum SIR during a call (SIRf) -- 18.1 dB *)
  POPULATION = 384; (* Total number of users *)
  RESOURCES = 48; (* Total number of channels (C) *)
  STATIONS = 12; (* Total number of cells *)
  WINSIZE = 4; (* Window size for updating priority values (w) *)
  STOPTIME = 3600; (* Maximum time for simulation -- 60 MINUTES *)
  HANDRESERV = 5; (* Total number of channels reserved for handoff (h) *)

type
  (* A channel is defined by an integer number (its number in the sequence of *)
  (* existing channels) and one cell in which it is active. *)
  channel = record
    num, cel : integer;
  end;

  (* The terminal type is the simulation's fundamental data structure. *)
  (* Each user is defined by: *)
  (* 1. A position, a real number between 0 and 12, the width of the system; *)
  (* 2. A velocity, distributed according to procedure ChangeVelocity; *)
  (* 3. A desire to speak, boolean; *)
  (* 4. A channel presently using; *)
  (* 5. A countdown stopwatch, showing how longer the present call (if any) *)
  (* will last. = Zero if no call is in progress; *)
  (* 6. A countdown stopwatch, showing the rest time for a new call request *)
  terminal = record
    posi : real; (* position *)
    velo : real; (* velocity *)
    rqst : boolean; (* request for channel *)
    chnP : channel; (* present channel *)
    ttgo : integer; (* time-to-go: countdown to end call *)
    timenewcall : integer; (* countdown for a new call request *)
  end;

var
  Time : integer; (* keeps track of elapsed simutime *)
  User : array[1..POPULATION] of terminal; (* collection of subscribers *)
  CIU : array[1..STATIONS,1..RESOURCES] of boolean;
      (* marks Channels In Use or not at each cell *)
  G : array[1..POPULATION,1..STATIONS] of real; (* link-gain matrix *)
  RCU : array[1..STATIONS] of integer; (* number of Regular Channels in Use *)
  HCU : array[1..STATIONS] of integer; (* number of Handoff Channels in Use *)
  RPriorit : array[1..STATIONS,1..(RESOURCES-HANDRESERV)] of real;
      (* priority value for each regular channel in each cell *)
  RNewSamp : array[1..STATIONS,1..(RESOURCES-HANDRESERV)] of integer;
      (* new sample of quality sense for regular channels per cell *)
  ROldSam1 : array[1..STATIONS,1..(RESOURCES-HANDRESERV)] of integer;
      (* 1st old sample of quality sense for regular channels per cell *)
  ROldSam2 : array[1..STATIONS,1..(RESOURCES-HANDRESERV)] of integer;
      (* 2nd old sample of quality sense for regular channels per cell *)
  ROldSam3 : array[1..STATIONS,1..(RESOURCES-HANDRESERV)] of integer;
      (* 3rd old sample of quality sense for regular channels per cell *)

```



```

ROldSam4 : array[1..STATIONS,1..(RESOURCES-HANDRESERV)] of integer;
(* 4th old sample of quality sense for regular channels per cell *)
HPriorit : array[1..STATIONS,1..HANDRESERV] of real;
(* priority value for each handoff channel in each cell *)
HNewSamp : array[1..STATIONS,1..HANDRESERV] of integer;
(* new sample of quality sense for handoff channels per cell*)
HOldSam1 : array[1..STATIONS,1..HANDRESERV] of integer;
(* 1st old sample of quality sense for handoff channels per cell *)
HOldSam2 : array[1..STATIONS,1..HANDRESERV] of integer;
(* 2nd old sample of quality sense for handoff channels per cell *)
HOldSam3 : array[1..STATIONS,1..HANDRESERV] of integer;
(* 3rd old sample of quality sense for handoff channels per cell *)
HOldSam4 : array[1..STATIONS,1..HANDRESERV] of integer;
(* 4th old sample of quality sense for handoff channels per cell *)
Stats : record
    call_tries: integer; (* total number of new call requests *)
    call_fails: integer; (* total number of new call denials *)
    hand_tries: integer; (* total number of handoffs tries *)
    hand_fails: integer; (* total number of handoffs fails *)
end;
NULLCHANNEL : channel; (* .num = .cel = 0 *)
NEWCALL : boolean; (* states whether the call is a new or an ongoing one *)
(*===== End of Declarations =====*)

(*===== Auxiliary Procedures & Functions =====*)
procedure ChangeVelocity(m:integer);
(**)
(* Sets a new velocity to User[m]. By altering this procedure, we can test *)
(* the assignment algorithm under different mobile behaviors. *)
(**)
(* Presently, the probability density functions are discrete: *)
(* - if user is talking: .10p x (-stp) + .80p x (0) + .10p x (stp) *)
(* - if user is not talking: .33p x (-stp) + .33p x (0) + .33p x (stp) *)
(* where stp is the user step length. The intent is to actually mimic *)
(* pedestrian users. When they speak, they will most probably stop walking, *)
(* hence the .80 probability on zero velocity. When they are disconnected, *)
(* they may walk either way or be still, all three velocities having *)
(* barely the same probability. *)
(**)
const
    stp = 0.0015432; (* user step length *)
(* velocity of user is equal to 5 kilometers per hour *)
(* diameter of cell is 900 meters, and this is equivalent to the space unit *)
var
    r, v : real; (* random num between 0 and 1; new velocity *)
begin
    r := Random;
    if User[m].chnP.num > 0 then (* if user is talking... *)
        begin
            if r <= 0.1 then v := -stp else
            if r < 0.9 then v := 0 else
            v := stp;
        end
    else (* if user is not talking... *)
        begin
            if r <= 0.33 then v := -stp else
            if r < 0.67 then v := 0 else
            v := stp;
        end;
    User[m].velo := v;
end; (* procedure ChangeVelocity(m) *)

function Expttgo : integer;
(**)

```

```

(* Exponential Distributed Random Integer Number. When an user is given a *)
(* channel, his time-to-go (ttgo) variable is set to such a number.      *)
(* It measures the total time he (she) will hold the channel.            *)
(**)
const
  mean = 120;          (* mean time of call duration -- 2 MINUTES *)
var
  numel, nume2 : real; (* scrap vars *)
begin
  numel := Random;      (* random number between 0 and 1 *)
  if numel = 0 then
    nume2 := 0.0000000001 (* to avoid overflow *)
  else
    nume2 := numel;
  Expttgo := Round(-mean*ln(nume2));
end; (* function Expttgo *)

function Expnewcall : integer;
(**)
(* Exponential Distributed Random Integer Number.      *)
(* It measures the total time for a new call request. *)
(**)
var
  numel, nume2 : real; (* scrap vars *)
begin
  numel := Random;      (* random number between 0 and 1 *)
  if numel = 0 then
    nume2 := 0.0000000001 (* to avoid overflow *)
  else
    nume2 := numel;
  Expnewcall := Round(-CALLINTERV*ln(nume2));
end; (* function Expnewcall *)

function Cellof(m:integer) : integer;
(**)
(* Returns the cell number (1-12) where User[m] is. *)
(**)
begin
  (* wrap-around the user, if necessary *)
  if User[m].posi >= STATIONS then User[m].posi := User[m].posi - STATIONS;
  if User[m].posi < 0 then User[m].posi := User[m].posi + STATIONS;
  Cellof := Trunc(User[m].posi) + 1;
end; (* function Cellof *)

function CalcSIR(m,c:integer) : real;
(**)
(* Returns the Signal-to-Interference Ratio for user m when using channel c *)
(**)
var
  i, j : integer; (* loop var, scrap var *)
  k, sum : real;   (* scrap var, amount of interference *)
begin
  sum := 0;
  for i := 1 to STATIONS do
    if CIU[i,c] = TRUE then sum := sum + G[m,i];
    (* include gain between user m and station i when *)
    (* channel c is being used in that station      *)
  if NEWCALL = TRUE then
    begin
      if sum > 0 then
        CalcSIR := G[m,Cellof(m)]/sum
      else (* any other cell is using channel c *)
        CalcSIR := FLOORBLOC; (* to avoid overflow *)
    end
end

```

```

else      (* call is ongoing *)
begin
  j := User[m].chnP.cel;
  k := G[m,j];
  if (sum-k) > 0 then
    CalcSIR := k/(sum-k)
  else (* any other cell is using channel c *)
    CalcSIR := FLOORHAND; (* to avoid overflow *)
  end;
end; (* function CalcSIR *)

function Dist(i,j:integer) : real;
(**)
(* Returns the distance between user i and ERB j. *)
(**)
var
  d : real; (* scrap var *)
begin
  d := Abs(User[i].posi - (j-0.5)); (* ERB is at the center of the cell *)
  if d > STATIONS/2 then
    Dist := STATIONS - d (* because wrap-around is used *)
  else
    Dist := d;
  end; (* function Dist *)

function ChannelChoice(u:integer) : integer;
(**)
(* Channel assignment module. This is the main intracellular channel *)
(* assignment algorithm, which means: this is the algorithm that each base *)
(* station uses to allocate channels to requests by users inside its cell. *)
(* To test the effectiveness of a new assignment algorithm, try it here. *)
(**)
var
  RPriValAux : array [1..(RESOURCES-HANDRESERV)] of real;
    (* auxiliar vector for priority values of regular channels *)
  RChnNumAux : array [1..(RESOURCES-HANDRESERV)] of integer;
    (* auxiliar vector for the regular channels numbers *)
  HPriValAux : array [1..HANDRESERV] of real;
    (* auxiliar vector for priority values of handoff channels *)
  HChnNumAux : array [1..HANDRESERV] of integer;
    (* auxiliar vector for the handoff channels numbers *)
  a : integer; (* scrap var *)
(**)
Procedure RSortPrior(cell:integer);
(**)
(* For a given cell, this procedure performs a sorting on the regular *)
(* channels according to their priority values. The sorting order is from *)
(* the greatest to the smallest value. "Bubble sort" algorithm is used. *)
(**)
var
  i, j : integer; (* loops *)
  logicsize : integer; (* current logical size of auxiliar vectors *)
  privalscrap : real; (* sort var for the priority values *)
  chnnumscrap : integer; (* sort var for the channels numbers *)
  changed : boolean; (* condition for "bubble sort" algorithm *)
begin
  for i := 1 to (RESOURCES-HANDRESERV) do
    begin
      (* copy priority values to the corresponding auxiliar vector *)
      RPriValAux[i] := RPriorit[cell,i];
      (* copy channels numbers to the corresponding auxiliar vector *)
      RChnNumAux[i] := i;
    end;
  logicsize := (RESOURCES-HANDRESERV);

```

```

repeat      (* "bubble sort" algorithm *)
  changed := FALSE;
  for j := 1 to (logicsize - 1) do
    begin
      if (RPriValAux[j] < RPriValAux[j+1]) then
        begin (* swap priority values and channels numbers *)
          privalscrap := RPriValAux[j+1];
          chnnumscrap := RChnNumAux[j+1];
          RPriValAux[j+1] := RPriValAux[j];
          RChnNumAux[j+1] := RChnNumAux[j];
          RPriValAux[j] := privalscrap;
          RChnNumAux[j] := chnnumscrap;
          changed := TRUE;
        end;
      end;
    logicsize := logicsize - 1;
  until (NOT changed) or (logicsize = 1);
end; (* Procedure RSortPrior *)
(**)
Function RChnFree(cell:integer) : integer;
(**)
(* Returns the number of the first free regular channel at cell *)
(**)
var
  chn, i, j, k : integer; (* channel number, loop var, scrap vars *)
  foundfree : boolean; (* test *)
begin
  foundfree := FALSE;      (* assume free chn will not be found *)
  i := 0;
  repeat
    i := i + 1;
    chn := RChnNumAux[i];
    if CIU[cell,chn] = FALSE then (* chn is not being used in cell *)
      begin
        ROldSam4[cell,chn] := ROldSam3[cell,chn]; (* discard the      *)
        ROldSam3[cell,chn] := ROldSam2[cell,chn]; (* current oldest  *)
        ROldSam2[cell,chn] := ROldSam1[cell,chn]; (* saved sample.   *)
        ROldSam1[cell,chn] := RNewSamp[cell,chn];
        NEWCALL := TRUE;
        if CalcSIR(u,chn) >= FLOORBLOC then (* measure the quality *)
          begin
            foundfree := TRUE;
            RNewSamp[cell,chn] := 1;      (* quality was sensed good *)
          end
        else
          RNewSamp[cell,chn] := -1;      (* quality was sensed bad  *)
          j := ROldSam4[cell,chn];
          k := RNewSamp[cell,chn];
          RPriorit[cell,chn] := ((RPriorit[cell,chn]*WINSIZE)-j+k)/WINSIZE;
          (* update priority value for chn in cell *)
        end;
      until (foundfree) or (i = (RESOURCES-HANDRESERV));
      if foundfree then
        RChnFree := chn  (* allocation will be possible *)
      else
        RChnFree := 0;
      end;
  end; (* Function RChnFree *)
(**)
Procedure HSortPrior(cell:integer);
(**)
(* For a given cell, this procedure performs a sorting on the handoff *)
(* channels according to their priority values. The sorting order is from *)
(* the greatest to the smallest value. "Bubble sort" algorithm is used. *)
(**)

```

```

var
  i, j : integer; (* loops *)
  logicsize : integer; (* current logical size of auxiliar vectors *)
  privalscrap : real; (* sort var for the priority values *)
  chnnumscrap : integer; (* sort var for the channels numbers *)
  changed : boolean; (* condition for "bubble sort" algorithm *)
begin
  for i := 1 to HANDRESERV do
    begin
      (* copy priority values to the corresponding auxiliar vector *)
      HPriValAux[i] := HPriorit[cell,i];
      (* copy channels numbers to the corresponding auxiliar vector *)
      HChnNumAux[i] := i;
    end;
  logicsize := HANDRESERV;
  repeat (* "bubble sort" algorithm *)
    changed := FALSE;
    for j := 1 to (logicsize - 1) do
      begin
        if (HPriValAux[j] < HPriValAux[j+1]) then
          begin (* swap priority values and channels numbers *)
            privalscrap := HPriValAux[j+1];
            chnnumscrap := HChnNumAux[j+1];
            HPriValAux[j+1] := HPriValAux[j];
            HChnNumAux[j+1] := HChnNumAux[j];
            HPriValAux[j] := privalscrap;
            HChnNumAux[j] := chnnumscrap;
            changed := TRUE;
          end;
        end;
      logicsize := logicsize - 1;
    until (NOT changed) or (logicsize = 1);
  end; (* Procedure HSortPrior *)
  (**)
Function HChnFree(cell:integer) : integer;
  (**)
  (* Returns the number of the first free handoff channel at cell *)
  (**)
var
  chn, i, j, k, z : integer; (* channel number, loop var, scrap vars *)
  foundfree : boolean; (* test *)
begin
  foundfree := FALSE; (* assume free chn will not be found *)
  i := 0;
  repeat
    i := i + 1;
    z := HChnNumAux[i];
    chn := z + RESOURCES - HANDRESERV;
    if CIU[cell,chn] = FALSE then (* chn is not being used in cell *)
      begin
        HOldSam4[cell,z] := HOldSam3[cell,z]; (* discard the *)
        HOldSam3[cell,z] := HOldSam2[cell,z]; (* current oldest *)
        HOldSam2[cell,z] := HOldSam1[cell,z]; (* saved sample. *)
        HOldSam1[cell,z] := HNewSamp[cell,z];
        NEWCALL := TRUE;
        if CalcSIR(u,chn) >= FLOORBLOC then (* measure the quality *)
          begin
            foundfree := TRUE;
            HNewSamp[cell,z] := 1; (* quality was sensed good *)
          end
        else
          HNewSamp[cell,z] := -1; (* quality was sensed bad *)
          j := HOldSam4[cell,z];
          k := HNewSamp[cell,z];
        end
      end
    end
  until foundfree;
  foundfree := TRUE;
  HNewSamp[cell,z] := 1; (* quality was sensed good *)
  end
end
  j := HOldSam4[cell,z];
  k := HNewSamp[cell,z];
end

```

```

        HPriorit[cell,z] := ((HPriorit[cell,z]*WINSIZE)-j+k)/WINSIZE;
        (* update priority value for channel z in cell *)
    end;
until (foundfree) or (i = HANDRESERV);
if foundfree then
    HChnFree := chn    (* allocation will be possible *)
else
    HChnFree := 0;
end; (* Function HChnFree *)
(**)
begin
    ChannelChoice := 0;    (* assume failure *)
    a := 0;
    if NEWCALL = FALSE then    (* handoff case *)
        begin
            HSortPrior(Cellof(u));    (* sort priority values for handoff channels *)
            if HChnFree(Cellof(u)) > 0 then    (* search succeeded *)
                a := HChnFree(Cellof(u));
            end;
        end;
    if a = 0 then    (* search not succeeded OR new call case *)
        begin
            RSortPrior(Cellof(u));    (* sort priority values for regular channels *)
            if RChnFree(Cellof(u)) > 0 then    (* search succeeded *)
                a := RChnFree(Cellof(u));
            end;
        end;
    ChannelChoice := a;
end; (* function ChannelChoice *)

function Power(x,y:real) : real;
(**)
(* Returns x to the y-th power. *)
(**)
begin
    if x > 0 then
        Power := exp(y*ln(x))
    else
        Power := 0.0000000001;    (* to avoid overflow in procedure Update_G *)
    end; (* function Power *)

procedure Update_G;
(**)
(* Updates the link-gain matrix *)
(* This procedure is here to increase execution time to a desirable level *)
(**)
var
    x,y : integer;    (* user var, ERB var *)
begin
    for x := 1 to POPULATION do
        for y := 1 to STATIONS do
            begin
                G[x,y] := 1/Power(Dist(x,y),ALFA);
            end;
        end;
    end; (* procedure Update_G *)

procedure SeizeChn(chn,m,cell:integer);
(**)
(* Marks channel chn in cell as busy (=TRUE). *)
(**)
begin
    CIU[cell,chn] := TRUE;
    if chn <= (RESOURCES-HANDRESERV) then    (* regular channel *)
        RCU[cell] := RCU[cell] + 1
    else
        HCU[cell] := HCU[cell] + 1;

```

```

end; (* procedure SeizeChn *)

procedure ReleaseChn(chn,m,cell:integer);
(**)
(* Marks channel chn in cell as free (=FALSE). *)
begin
  CIU[cell,chn] := FALSE;
  if chn <= (RESOURCES-HANDRESERV) then (* regular channel *)
    RCU[cell] := RCU[cell] - 1
  else
    HCU[cell] := HCU[cell] - 1;
end; (* procedure ReleaseChn *)
(*===== End of Auxiliary Procedures & Functions =====*)

(*===== Main Procedures =====*)
procedure Initialize;
(**)
(* Sets up initial position & telecom status of all users. *)
(**)
var
  m, n : integer; (* loops *)
begin
  Randomize;
  TIME := 0;
  for n := 1 to POPULATION do with User[n] do
    begin
      posi := 12*Random; (* uniform initial distribution *)
      ChangeVelocity(n); (* initial velocity *)
      rqst := FALSE; (* no user wants to talk *)
      chnP.num := 0; (* no user starts off talking *)
      ttgo := 0; (* time-to-go is zeroed. *)
      timenewcall := Expnewcall; (* time for a new call request *)
    end;
    FillChar(CIU,SizeOf(CIU),0); (* no channel in use at any cell *)
    FillChar(RCU,SizeOf(RCU),0);
    FillChar(HCU,SizeOf(HCU),0);
    for m := 1 to STATIONS do
      for n := 1 to (RESOURCES-HANDRESERV) do
        begin
          RPriorit[m,n] := Random; (* set initial priority values *)
        end; (* for regular channels. *)
      end;
    for m := 1 to STATIONS do
      for n := 1 to HANDRESERV do
        begin
          HPriorit[m,n] := Random; (* set initial priority values *)
        end; (* for handoff channels. *)
      end;
    FillChar(RNewSamp,SizeOf(RNewSamp),0); (* no channel was sensed yet *)
    FillChar(ROldSam1,SizeOf(ROldSam1),0);
    FillChar(ROldSam2,SizeOf(ROldSam2),0);
    FillChar(ROldSam3,SizeOf(ROldSam3),0);
    FillChar(ROldSam4,SizeOf(ROldSam4),0);
    FillChar(HNewSamp,SizeOf(HNewSamp),0);
    FillChar(HOldSam1,SizeOf(HOldSam1),0);
    FillChar(HOldSam2,SizeOf(HOldSam2),0);
    FillChar(HOldSam3,SizeOf(HOldSam3),0);
    FillChar(HOldSam4,SizeOf(HOldSam4),0);
    Stats.call_tries := 0; (* reset statistics *)
    Stats.call_fails := 0;
    Stats.hand_tries := 0;
    Stats.hand_fails := 0;
    NULLCHANNEL.num := 0;
    NULLCHANNEL.cel := 0;
    ClrScr;
  end; (* procedure Initialize *)

```

```

procedure MoveUsers;
(**)
(* Updates the position of each user according to the velocity; changes this *)
(* velocity randomly; updates each user's desire to speak (rqst). *)
(**)
const
  VeloChangeProb = 0.1; (* probability of changing velocity *)
var
  n : integer; (* loop *)
  sir : real; (* SIR scrap var *)
begin
  for n := 1 to POPULATION do with User[n] do
    begin
      posi := posi + velo; (* move user *)
      (* wrap-around, if necessary *)
      if posi >= STATIONS then posi := posi - STATIONS;
      if posi <= 0 then posi := posi + STATIONS;
      if chnP.num > 0 then (* if user is talking... *)
        begin
          NEWCALL := FALSE; (* ongoing call *)
          sir := CalcSIR (n,chnP.num); (* measure the quality of the link *)
          if sir < FLOORHAND then (* quality sensed bad *)
            begin (* a handoff must be tried *)
              ReleaseChn(chnP.num,n,chnP.cel);
              Stats.hand_tries := Stats.hand_tries + 1;
              chnP.cel := Cellof(n);
              chnP.num := ChannelChoice(n);
              if chnP.num = 0 then (* handoff failed *)
                Stats.hand_fails := Stats.hand_fails + 1
              else (* handoff succeeded *)
                begin
                  SeizeChn(chnP.num,n,chnP.cel);
                  G[n,chnP.cel] := 1/Power(Dist(n,chnP.cel),ALFA);
                  (* update link-gain value *)
                end;
              end;
            end;
          end;
          if Random < VeloChangeProb then (* eventually, change the velocity *)
            ChangeVelocity(n);
          if timenewcall > 0 then
            timenewcall := timenewcall - 1
          else (* time for a new exponential generation *)
            begin
              timenewcall := Expnewcall;
              if chnP.num = 0 then rqst := TRUE; (* new request, if not talking *)
            end;
          end; (* for *)
          Update_G; (* update the link-gain matrix *)
          (* this procedure increases execution time to a desirable level *)
        end; (* procedure MoveUsers *)
      end;
    end;
  end;

```

```

procedure ServeUsers;
(**)
(* Main telecom administration task. For each user there is either a channel *)
(* assignment trial (if the user's request state is TRUE), or a continuation *)
(* of his (her) countdown (if the user is already speaking). *)
(**)
var
  n : integer; (* users counter *)
begin
  n := 0;
  repeat
    n := n + 1;
  until

```



```

with User[n] do
begin
  if rqst then                                (* if user wants to talk *)
  begin
    chnP.cel := Cellof(n);
    NEWCALL := TRUE;
    chnP.num := ChannelChoice(n);
    Stats.call_tries := Stats.call_tries + 1;
    if chnP.num = 0 then (* blocking occurred *)
    begin
      Stats.call_fails := Stats.call_fails + 1;
      rqst := FALSE;
    end
    else (* allocation succeeded *)
    begin
      chnP.cel := Cellof(n);
      SeizeChn(chnP.num,n,chnP.cel);
      G[n,chnP.cel] := 1/Power(Dist(n,chnP.cel),ALFA);
      (* update link-gain value *)
      rqst := FALSE;
      ttgo := Expttgo;
    end;
  end
else
  (* if user does not want to talk: *)
  if ttgo > 0 then
    (* see if he (she) IS talking *)
    ttgo := ttgo - 1
    (* if so, continue countdown *)
  else
  begin
    if chnP.num > 0 then (* countdown ends: release channel *)
    begin
      ReleaseChn(chnP.num,n,chnP.cel);
      chnP := NULLCHANNEL;
    end; (* if *)
  end; (* else *)
end;
until (n = POPULATION);
end; (* procedure ServeUsers *)

procedure DisplayData;
(**)
(* Displays: elapsed simutime, accumulated number of new calls and handoff *)
(* requests, accumulated number of denials, rates of new call blocks and *)
(* handoff fails, number of busy channels in each cell, and priority values *)
(* of five sample channels in each cell. *)
const
  NET = 14; (* The display line where the system is depicted. *)
var
  n, m, x, y, z : integer; (* loop var, scrap vars *)
  call_block : real; (* rate of new calls blocking *)
  hand_block : real; (* rate of handoffs fails *)
begin
  TextColor(White);
  GotoXY(1,NET-13);
  WRITE('Channel Segregation');
  GotoXY(1,NET-12);
  WRITE('using sliding window criterion with window size = ',WINSIZE:1);
  GotoXY(66,NET-12);
  WRITE('SIMUTIME');
  GotoXY(1,NET-11);
  WRITE('and reserving ',HANDRESERV:1,' channel(s) for handoff purposes');
  GotoXY(68,NET-10);
  WRITE(TIME:4);
  GotoXY(1,NET-9);
  WRITE('CALLINIC',' ', 'CALLDENY',' ', 'CALLBLOCRATE');

```

```

GotoXY(1,NET-7);
if Stats.call_tries > 0 then (* to avoid overflow *)
    call_block := (Stats.call_fails / Stats.call_tries) * 100
else
    call_block := 0;
with Stats do
    WRITE(' ',call_tries:5,' ',call_fails:5,' ',call_block:6:2,' %');
GotoXY(38,NET-7); (* SIRb *)
WRITE('SIR threshold for new calls: ',10*ln(FLOORBLOC)/ln(10):4:1,' dB');
GotoXY(34,NET-6); (* SIRf *)
WRITE('SIR threshold for handoff calls: ',10*ln(FLOORHAND)/ln(10):4:1,' dB');
GotoXY(41,NET-5);
WRITE('total number of channels: ',RESOURCES:4); (* C *)
GotoXY(1,NET-4);
WRITE('HANDINIC',' ', ' ', 'HANDDENY',' ', ' ', 'HANDFAILRATE');
GotoXY(44,NET-4);
WRITE('total number of users: ',POPULATION:4);
GotoXY(42,NET-3);
WRITE('mean time between calls: ',(CALLINTERV/60):4:1,' mi');
GotoXY(1,NET-2);
if Stats.hand_tries > 0 then (* to avoid overflow *)
    hand_block := (Stats.hand_fails / Stats.hand_tries) * 100
else
    hand_block := 0;
with Stats do
    WRITE(' ',hand_tries:5,' ',hand_fails:5,' ',hand_block:6:2,' %');
WRITELN;
WRITELN;
(**)
(* The next loop depicts the net: '+'=ERB, '|'=Frontier. *)
for n := 1 to 73 do
    begin
        if (n+2)/6 = Int((n+2)/6) then WRITE('+') else
        if (n+5)/6 = Int((n+5)/6) then WRITE('|') else
        WRITE(' ');
    end;
(**)
(* The next loop depicts the users. *)
for n := 1 to POPULATION do with User[n] do
    begin
        TextColor(LightBlue);(* BLUE users are not talking and do not want to*)
        if ttgo > 0 then
            TextColor(Green); (* GREEN users are talking now *)
            GotoXY(Trunc(6*posi)+1,NET);
            WRITE('!');
        end; (* for *)
    TextColor(White);
    WRITELN;
    WRITELN;
    (**)
    (* The next loop depicts the cells numbers *)
    for n := 1 to 73 do
        begin
            case n of
                1: WRITE('.');
                4: WRITE('1');
                7: WRITE('.');
                10: WRITE('2');
                13: WRITE('.');
                16: WRITE('3');
                19: WRITE('.');
                22: WRITE('4');
                25: WRITE('.');
                28: WRITE('5');

```

```

31: WRITE(' ');
34: WRITE('6');
37: WRITE(' ');
40: WRITE('7');
43: WRITE(' ');
46: WRITE('8');
49: WRITE(' ');
52: WRITE('9');
55: WRITE(' ');
57: WRITE('1 0');
59: WRITE(' ');
61: WRITE('1 1');
63: WRITE(' ');
65: WRITE('1 2');
67: WRITE(' ');
else WRITE(' ');
end; (* case *)
end; (* for *)
Writeln;
(**)
(* The next loop depicts the quantities of Regular Channels in Use *)
(* in each cell. *)
for n := 1 to 56 do
begin
if (n < 47) and ((n+2)/4 = Int((n+2)/4)) then
begin
m := Trunc((n+2)/4);
if RCU[m] = RESOURCES-HANDRESERV then
TextColor(Red) (* all regular channels used in this cell *)
else
TextColor(White);
WRITE(RCU[m]:3);
end
else
WRITE(' ');
end;
TextColor(White);
(**)
(* The next loop depicts the quantities of Handoff Channels in Use *)
(* in each cell. *)
for n := 1 to 73 do
begin
if (n < 47) and ((n+2)/4 = Int((n+2)/4)) then
begin
m := Trunc((n+2)/4);
if HCU[m] = HANDRESERV then
TextColor(Red) (* all handoff channels used in this cell *)
else
TextColor(White);
WRITE(HCU[m]:3);
end
else
WRITE(' ');
end;
TextColor(White);
Writeln;
(**)
(* The next five loops depict the priority values of five channels in *)
(* each cell. *)
y := 8; (* regular channel number 8 *)
for n := 1 to 73 do
begin
if (n < 25) and (n/2 = Int(n/2)) then
begin

```

```

    x := Trunc(n/2);
    if RPriorit[x,y] > 1 then
        TextColor(Blue)          (* high priority value is BLUE depicted *)
    else
        begin
            if RPriorit[x,y] < 0 then
                TextColor(Red)      (* low priority value is RED depicted *)
            else
                TextColor(White); (* neither high nor low priority value *)
            end;
            WRITE(RPriorit[x,y]:5:2);
        end
    else
        begin
            if n = 27 then
                begin
                    TextColor(White);
                    WRITE(y:2);
                end
            else
                WRITE(' ');
            end;
        end;
    end;
GotoXY(1,NET+7);
y := 18; (* regular channel number 18 *)
for n := 1 to 73 do
    begin
        if (n < 25) and (n/2 = Int(n/2)) then
            begin
                x := Trunc(n/2);
                if RPriorit[x,y] > 1 then
                    TextColor(Blue)          (* high priority value is BLUE depicted *)
                else
                    begin
                        if RPriorit[x,y] < 0 then
                            TextColor(Red)      (* low priority value is RED depicted *)
                        else
                            TextColor(White); (* neither high nor low priority value *)
                        end;
                        WRITE(RPriorit[x,y]:5:2);
                    end
                end
            else
                begin
                    if n = 27 then
                        begin
                            TextColor(White);
                            WRITE(y:2);
                        end
                    else
                        WRITE(' ');
                    end;
                end;
            end;
        end;
GotoXY(1,NET+8);
y := 28; (* regular channel number 28 *)
for n := 1 to 73 do
    begin
        if (n < 25) and (n/2 = Int(n/2)) then
            begin
                x := Trunc(n/2);
                if RPriorit[x,y] > 1 then
                    TextColor(Blue)          (* high priority value is BLUE depicted *)
                else
                    begin
                        if RPriorit[x,y] < 0 then

```



```

        end
    else
        begin
            if n = 27 then
                begin
                    TextColor(White);
                    WRITE(y:2);
                end
            else
                WRITE(' ');
            end;
        end;
    end;
end; (* procedure DisplayData *)
(*===== End of Main Procedures =====*)

(*===== Main Routine =====*)
BEGIN
    Initialize;
    repeat
        if TIME < STOPTIME then
            begin
                TIME := TIME + 1;          (* For each new instant in simutime: *)
                MoveUsers;                 (* 1. Move the users *)
                ServeUsers;                (* 2. Try to serve requests for channels *)
            end;
            DisplayData;                  (* 3. Update system display *)
        until false;
    END.
    (*===== End of Main Routine =====*)

```

**“UMA ANÁLISE DO PROBLEMA DA ALOCAÇÃO DOS CANAIS
DE TRÁFEGO EM SISTEMAS MÓVEIS CELULARES”**

Dissertação de Mestrado apresentada por SERGIO LUIZ NORONHA FRAIHA em
13 de setembro de 1996 ao departamento de Engenharia Elétrica da PUC/Rio, e
aprovada pela Comissão Julgadora formada pelos seguintes professores:

JOSÉ ROBERTO BOISSON DE MARCA
ORIENTADOR - CETUC - PUC/RIO

GLAUCIO LIMA SIQUEIRA
CETUC - PUC/RIO

JOÃO CELIO BARROS BRANDÃO
CETUC - PUC/RIO

RAIMUNDO SAMPAIO NETO
CETUC - PUC/RIO

Visto e permitida a impressão:
Rio de Janeiro, 24/09/1996

MARIA AUGUSTA DAVIDOVICH
COORDENADORA DOS PROGRAMAS
DE PÓS-GRADUAÇÃO DO CENTRO
TÉCNICO-CIENTÍFICO DA PUC/RIO