EGE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ

(YÜKSEK LİSANS TEZİ)

GSM'DE FREKANS PLANLAMA YÖNTEMLERİ VE HÜCRELERE FREKANS ATAMASI YAPACAK PLANLAMA YAZILIMI GERÇEKLEŞTİRİMİ

Serkan KAYACAN
Bilgisayar Mühendisliği Anabilim Dalı
Bilim Dalı Kodu : 619.01.00
Sunuş Tarihi : 21.09.2007

Tez Danışmanı : Prof. Dr. Levent TOKER

Bornova - İZMİR

Serkan KAYACAN tarafından yüksek lisans tezi olarak sunulan "**GSM'de Frekans Planlama Yöntemleri ve Hücrelere Frekans Ataması Yapacak Planlama Yazılımı Gerçekleştirimi**" başlıklı bu çalışma E.Ü. Lisansüstü Eğitim ve Öğretim Yönetmeliği ile E.Ü. Fen Bilimleri Enstitüsü Eğitim ve Öğretim Yönergesi'nin ilgili hükümleri uyarınca tarafımızdan değerlendirilerek savunmaya değer bulunmuş ve 21/09/2007 tarihinde yapılan tez savunma sınavında aday oybirliği/oyçokluğu ile başarılı bulunmuştur.

Jüri Üyeleri:	İmza
Jüri Başkanı : Prof. Dr. Levent TOKER	.....
Raportör Üye : Prof. Dr. Serdar KORUKOĞLU	.....
Üye :Yrd. Doç. Dr. Radosveta SOKULLU	.....

ÖZET

GSM'DE FREKANS PLANLAMA YÖNTEMLERİ VE HÜCRELERE FREKANS ATAMASI YAPACAK PLANLAMA YAZILIMI GERÇEKLEŞTİRİMİ

KAYACAN, Serkan

Yüksek Lisans Tezi, Bilgisayar Mühendisliği Bölümü

Tez Yöneticisi: Prof. Dr. Levent TOKER

Eylül 2007, 250 sayfa

Çok sayıda abonenin bulunduğu GSM şebekelerinde, mobil telefon servislerinin kullanımının artması ve bunun yanında ses ve görüntü içeren yeni servislerin de gelmesiyle birlikte operatörler bu trafiği karşılayabilmek için ağ kapasitesini arttırmak zorunda kalırlar. Kapasiteyi arttırmak için yeni hücreler eklenebilir ya da mevcut makro hücreler bölünebilir. Bunun yanında operatörler gelişmiş ağ özelliklerini kullanarak frekans tekrar kullanım faktörünü azaltırlar. Spektrumun limitli olmasından ve donanımsal kısıtlardan dolayı mevcut makro hücrelerde kapasite artırımının konuşma kalitesinde kötüleşmeye neden olması kaçınılmazdır. Kaliteyi korumak için makro hücrelerin mikro hücrelere bölünmesiyle aynı alana daha çok hücre ile servis verilecektir. Bu durumda frekans planlama önem kazanacaktır.

Bu tez çalışmasında, mobil ölçümlere dayalı frekans planlaması yapan bir yazılım gerçeklenmiştir.

Anahtar Sözcükler: GSM şebekesi, hücresel ağ, hücre planlama, frekans planlama, ICDM, frekans atama yazılımı.

ABSTRACT

FREQUENCY PLANNING METHODS IN GSM AND IMPLEMENTATION OF FREQUENCY ALLOCATION TO CELLS

KAYACAN, Serkan

MSc in Computer Engineering

Supervisor: Prof. Dr. Levent TOKER

September 2007, 250 pages

The large number of subscribers in the network and the increasing usage of existing mobile telephony services together with upcoming new services, such as video and audio streaming, will force operators to significantly increase capacity offered by the network. Sectorisation and cell splitting are two standard methods to increase capacity from macrocells. Often operators also utilise advanced network features to reduce the frequency reuse factor and thus further increase network capacity of macro cells. Owing to the limited spectrum and because of HW constraints regarding channel separation required on a cell or a site basis, providing additional capacity from existing macro cells is often impossible without degrading call quality. Therefore, network operators also have to use alternative cell types such as micro cells to meet capacity requirements. The reduced cell sizes also create new challenges. The main challenge is frequency planning.

In this thesis, a software is implemented for mobile measurement-based frequency planning.

Keywords: GSM network, cellular network, cell planning, frequency planning, ICDM, frequency allocation software.

TEŞEKKÜR

Bu çalışmam süresince her türlü desteği sağlayan çok değerli danışmanım Sayın Prof. Dr. Levent Toker'e teşekkürü bir borç bilirim.

Çalışmam sırasında desteklerini hiç bir zaman esirgemeyen aileme, çalışma arkadaşlarım Mehmet Kurutepe, Kamil Solmaz ve Ülgen Ünal'a, arkadaşlarım İnanç Seylan ve Simge Aksu'ya teşekkür ederim.

İÇİNDEKİLER

ŞEKİLLER DİZİNİ

ÇİZELGELER DİZİNİ

KISALTMALAR

ARQ	Automatic Repeat Request
AUC	Authentication Center
BCCH	Broadcast Control Channel
BER	Bit Error Rate
BSC	Base Station Controller
BSIC	Base Station Identity Code
BSS	Base Station System
BTS	Base Transceiver Station
CDD	Cell Design Data
CEPT	Conferance of European Posts and Telgraphs
CGI	Cell Global Identity
EIR	Equipment Identity Register
FDMA	Frequency Division Multiple Access
FLP	Fractional Load Planning
GMSC	Gateway Mobile Services Switching Center
GMSK	Gaussian Minimum Shift Keying
GoS	Grade of Service
GSM	Global System For Mobile Communications
HLR	Home Location Register
HSN	Hopping Sequence Number
ICDM	Inter Cell Dependency Matrix
IMEI	International Mobile Equipment Identity
IMSI	International Mobile Subscriber Number
ISI	Inter Symbol Interference
LA	Location Area
LAI	Location Area Identity

MAIO	Mobile Allocation Index Offset
MCC	Mobile Country Code
MNC	Mobile Network Code
MOC	Mobile Originated Call
MoU	Memorandum of Understanding
MRP	Multiple Reuse Pattern
MS	Mobile Station
MSC	Mobile Services Switching Center
MSIC	Mobile Subscriber Identification Number
MTC	Mobile Terminated Call
OMC	Operation and Maintenance Center
PCM	Pulse Code Modulation
PIN	Personal Identification Number
PUK	Personal Unblocking Key
SDCCH	Stand Alone Dedicated Control Channel
SIM	Subscriber Identity Module
SRES	Signed Response
SS	Switching System
TCH	Traffic Channel
TDMA	Time Division Multiple Access
TMSI	Temporary Mobile Subscriber Identity
TRU	Transceiver Unit
TS	Time Slot
VLR	Visitor Location Register

1 GSM'E GİRİŞ

GSM (Global System For Mobile Communications - Küresel Mobil İletişim Sistemi), son yirmi yılda hızla gelişmiş olan ve çok talep gören bir iletişim teknolojisidir. Küreselleşme sürecine giren dünyamızda bu süreci en çok hızlandıran etken iletişim teknolojileri olmuştur ve bunda GSM' in payı büyüktür.

1.1 GSM'in Tarihçesi

GSM'in tarihi 1982'de Nordic PTT'sinin Avrupa Posta ve Telgraf Konferansına (CEPT - Conferance of European Posts and Telgraphs) 900 Mhz'de Avrupa'yı kapsayan genel bir telekomünikasyon servisi kurulması teklifi ile başlamıştır. Daha sonra Mobil Uzmanlık Grubu (Group Special Mobile) adı altında bir çalışma grubu oluşturulmuştur. Konferans sırasında oluşturulan bu çalışma grubunun amacı, Avrupa çapında, 900 Mhz aralığında işleyecek olan, kamuya açık bir hücresel iletişim sistemi oluşturmak olmuştur (Ericsson, 1998c).

Hücresel telekomünikasyon, telekomünikasyon uygulamalarının en hızlı gelişeni ve en çok talep görenidir. Tüm dünyada yeni telefon aboneliklerindeki en büyük yüzde payı hücresel telekomünikasyon uygulamalarındadır ve sürekli artmaktadır. Birçok hususta hücresel çözümler, ticari kablo ağları ve kablosuz telefonlarla boy ölçüşebilmektedir.

Sayısal radyo transmisyonu kullanımı ve GSM şebekelerdeki ileri algoritmik iletişim, analog hücresel sistemlerden daha mükemmel bir frekans kullanımı sağlar. Dolayısıyla sürekli artan abonelere daha iyi hizmet sunulmuş olur. GSM genel bir standardı sağlamakla birlikte, hücresel aboneler, telefonlarını ya da veri iletişim cihazlarını bütün GSM servis alanı üzerinde kullanabileceklerdir. Bu haberleşme dolaşımı, GSM sistemleri içeren ülkelerde ve bu ülkeler arasında otomatik olarak gerçekleşir.

Uluslararası dolaşıma ek olarak, yüksek hızda veri haberleşmesi, kopyalama ve kısa mesaj hizmetleri gibi kullanıcı servisleri sağlar. GSM'in teknik standartları ISDN vb. diğer standartlarla uyum içinde çalışabilecek şekilde hazırlanmıştır (Harputluoğlu, 2000).

1982–1985 yıllarında sayısal veya analog sistem kurulması konusunda tartışmalar vardı. Görüşmeler sonrasında 1985'de sayısal sistemde karar kılınmıştır. Bundan sonraki basamak ise dar bant çözümü veya geniş bant çözümü seçme meselesi olmuştur. 1986'da Paris'te farklı şirketlerin farklı çözümlerle boy ölçüşebilecekleri bir ortam oluşmuştur. 1987'nin Mayıs ayında dar bant TDMA (Time Division Multiple Access) çözümünde ortak karara varılmıştır. Aynı zamanda 13 ülke, sistemin belli kaidelerine ve özelliklerine uyacakları bir protokol imzalamışlardır (MoU - Memorandum of Understanding). Dolayısı ile büyük bir potansiyel piyasa ortaya çıkmıştır. Bu ülkeler anlaştıkları şekilde bir GSM sistemi faaliyete geçirmek için 1 Temmuz 1991 tarihini belirlemişlerdir (Ericsson, 1998c).

1.2 GSM Sistem Standartları

GSM Şebekeler için sistemin standartları;

Frekans Bandı	Yukarı link: 890 Mhz - 915 Mhz Aşağı link: 935 Mhz - 960 Mhz
Duplex Mesafe	45 Mhz
Taşıyıcı Ayrıştırması	200 Khz
Modülasyon	GMSK
Transmisyon Oranı	270 kbit/sn
Erişim Metodu	TDMA
Konuşma Kodlayıcısı	RPE LPC 13 kbit/sn
Çeşitlilik	Kanal kodlama Araya yerleştirme Adaptasyon dengeleme Frekans atlaması

1.3 Erişim Metodu

Sayısal GSM sistemi, her taşıyıcının sekiz zaman aralığına bölündüğü Zaman Bölmeli Çoklu Erişim Metodu (TDMA) kullanır (Ericsson, 1998a). Hareketli istasyon (Mobile Station) aynı zaman aralığında gönderir ve alır. Bu demektir ki, aynı zamandaki sekiz konuşma aynı radyo kanalında yer alabilir.

1.4 Modülasyon

Kullanılan modülasyon GMSK (Gaussian Minimum Shift Keying) 'dir.

1.5 GSM'in Avantajları

GSM ile oluşturulmak istenen özellikler şu maddeler ile verilebilir (Ericsson, 1998a);

- Çok daha iyi konuşma kalitesi (eşit veya mevcut olan analog hücresel teknolojiden daha iyi)
- Düşük terminal, işletim ve hizmet ücretleri
- Yüksek seviyede güvenlik
- Uluslararası dolaşım
- Düşük güçte çalışan portatif veya mobil terminal desteği
- Yeni hizmetlere ve şebekelere uyumluluk

2 SİSTEME GENEL BİR BAKIŞ

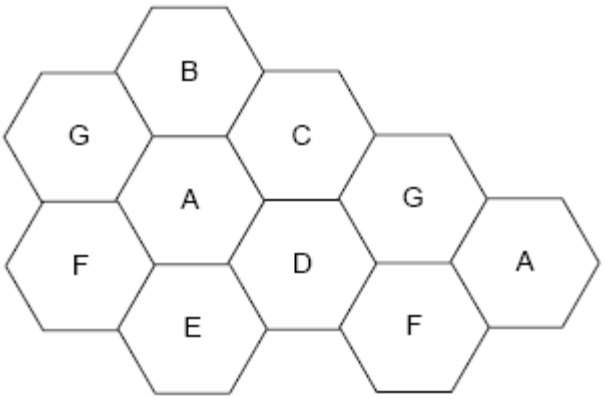
2.1 Hücresel Ağ

GSM, hücresel yapıya sahip bir iletişim sistemidir. GSM sistemindeki tüm hücreler, hücresel ağı oluşturmaktadır. Genelde bir hücre, anten sisteminden oluşan bir sektörün kapsama alanı olarak tanımlanabilir. Bir baz istasyonu birkaç hücreye sahip olabilir.

2.1.1 Hücresel Yapı

Radyo frekans haberleşmesinin ilk zamanlarında mühendisler alıcı ve verici arasında bir hat oluşturduklarında oldukça sevinmişlerdi. İlk hatlar iki yönlü iletişim için değildi. Bunlar tek-yön gönderme hatları olarak kaldılar ve mobilleri arayan insanlar hemen cevap alamazlardı. Hatta aramalarının mobil adreslere ulaşıp ulaşmadığını dahi hemen anlayamazlardı. Bundan sonraki basamak çift-yönlü, hemen cevap alınabilen bir transmisyon hattı kurmak oldu. Bu da mobil vericilerle sağlandı, fakat şebeke yapısı kolay kullanıma uygun değildi ve hizmet belli bir alan ile sınırlı idi. Bu alana, bir verici ile veya tek bir bölgede farklı kanallarda çalışan vericilerin küçük bir koleksiyonu ile ulaşmak mümkündü. Şebekenin tanıdığı bu alanlara “Hücre” denildi. Hücrenin veya şebekenin ebadı verici gücü ile ilgili idi. Hücredeki alıcı ve vericinin frekansını seçmek çok önemliydi. Çünkü diğer sistemlerden girişime çok müsaitti (Ericsson, 1998a).

Günümüz perspektifinden bakılırsa bu dezavantajlar açıkça kendini gösteriyor. Büyük bir bölge için küçük bir frekans grubu kullanılıyordu. Bütün bu problemlere çözüm arandı. Sonraları, frekans bandının ayrılıp birçok hücreler arasında bir hücreye tahsis edilmesi önerildi. Hücreler de birbirine bitişik ve bir arada olacaklardı. Böylece hücre yapısı oluştu (Şekil 2.1).



Şekil 2.1 - Aynı frekanslara sahip olamayan komşu hücreler

Bu projenin düzgün çalışması için bazı kısıtlamalar getirildi;

- Aynı kanalı kullanan farklı iki istasyon arasındaki girişimi azaltmak için frekanslar belli hücrelere tahsis edildi.

- Farklı istasyonlar arası girişimi azaltmak için, bir tek hücre içinde güç seviyeleri uygun bir şekilde ayarlandı. Bitişik hücrelerin birbirlerine yönelik girişime sebep olmaması için, güç sınırlı olmalıdır.

- Alıcı filtreleri geliştirildi. Günümüzde, bir mobil abone, mobil cihazı ile hücre kapsama alanı içinde her yerden arama gönderebilir ve arama alabilir.

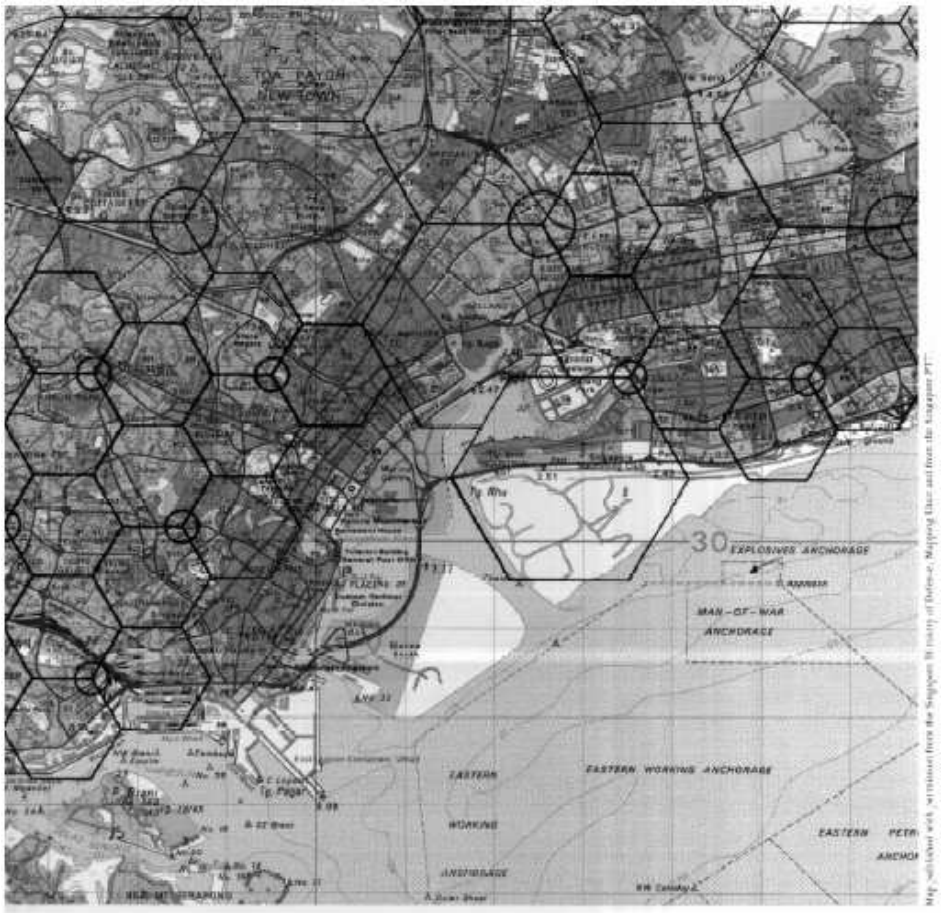
Hücre şekli teoride altıgen olarak gösterilir ve bu yapay bir gösterimdir. Baz istasyonu anteni tarafından yayılan sinyalin ideal kapsaması ise dairesel olarak gösterilir. Ancak gerçekte bazı alanlar çeşitli nedenlerle gerekli sinyal seviyesine sahip olamazlar. Bu sebeple hücreler pratikte geometrik olmayan şekillere sahiptirler (Mishra, 2004).

2.1.2 Ağ Planlaması

Eğer Amerika'daki gibi nüfusun farklı bölgelerde farklı yoğunluklarda olduğunu düşünürsek, her bölgede aynı büyüklükte hücre oluşturmanın mantıklı bir şey olmadığını görürüz. Operatör açısından olayı ele alalım: New York gibi büyük ve nüfusun yoğun olduğu bir bölge ile nüfusun seyrek olduğu Hawaii adasına, aynı işlevleri benzer bir ağ ile tedarik etmek mümkün değildir (Ericsson, 1998b). Bunun için ağ ve hücre planlamada farklı tasarımlar geliştirilmiştir.

2.1.2.1 Hücreleri Ayırma ve Mikro Hücre Uygulaması

Abone sayısı arttıkça şebeke içindeki yoğunluk da artmıştır. Operatörler ve radyo mühendisleri kapasiteyi arttırma yoluna gitmişlerdir. Oldukça temel bir fikir ortaya atılmıştır. Var olan alanı daha küçük parçalara ayırmak, böylelikle var olan kanal sayısını katlayarak, büyük hücreli eski duruma kıyasla, kapasiteyi çok daha yukarılara çekerek abone yoğunluğunu karşılamaktır (Şekil 2.2).

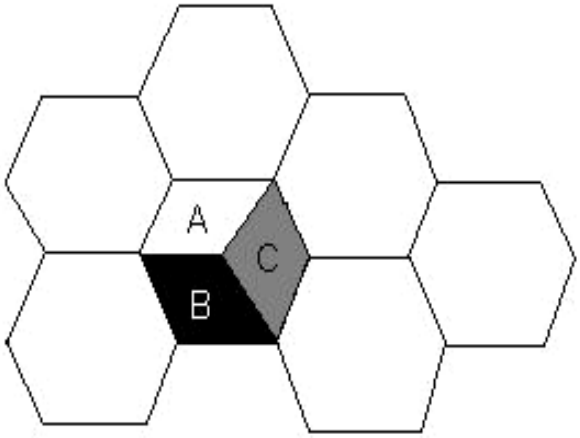


Şekil 2.2 - Hücreleri ayırma (Ericsson, 1998c)

Bu basit proje boyunca, hücrelerde kullanılan güç seviyeleri düşük tutulmuştur. Bundaki amaç ise mobil istasyonlar için gerekli olan pil büyüklüğünü azaltmak olmuştur. Mobiller için gerekli olan güç miktarı azalınca, mobillerin ebatları ve ağırlıkları da düşmüştür. Bu da şebekeleri kullanıcılar açısından daha çekici hale getirmiştir.

2.1.2.2 Sektörel Hücreler

Her zaman dairesel hücre oluşturmanın bir anlamı yoktur. Haberleşme mühendisleri hücreleri çok değişik şekillerde tasarlamışlardır. Bunda dikkate alınan kriterler yayımlanan gücün belirli bir bölge içinde sınırlandırılabilmesi ve komşu bölgelerden gelen gücün bu belirli bölgelere alınmaması olmuştur. Burada anten tasarımının önemi söz konusu olmuştur. Bu seçimli kapsama planının en genel olanı "bölgelere dilimlenmiş hücre" dir. Burada kapsama alanı tipik 360 dereceden ziyade 120 dereceyle sınırlandırılmıştır (Şekil 2.3). Böyle antenler genellikle tünel girişlerine, vadi kenarlarına ve gökdelenlerin aralarına yerleştirilirler (Harputluoğlu, 2000).

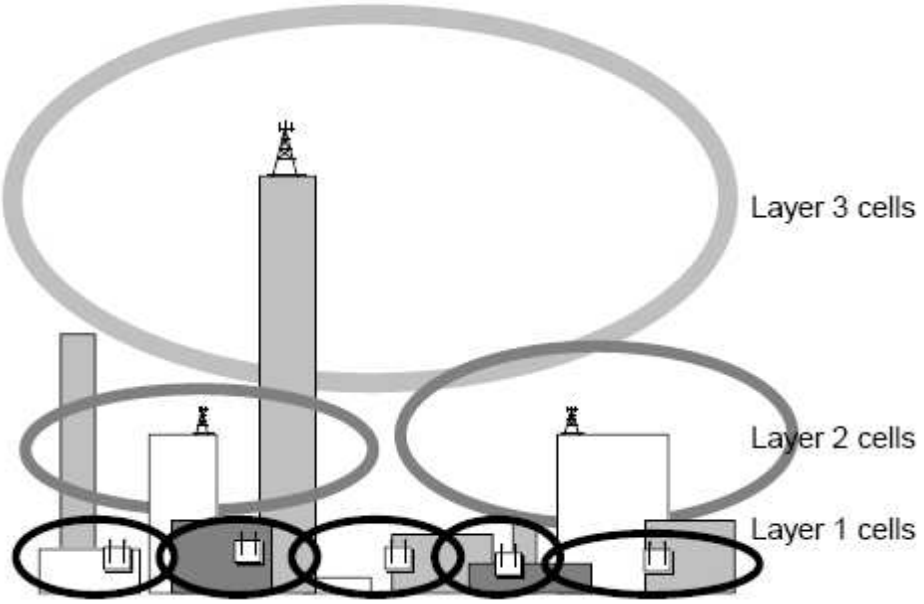


Şekil 2.3 - Sektörel Hücreler

2.1.2.3 Şemsiye Hücreler

Hücre ayırma tekniği ilk uygulandığında, operatörler çok küçük hücreler içinde bir transit geçişin farkına varmışlardır. Bu da farklı küçük hücreler arasında çok büyük sayıda hücreden hücreye işaret geçişine sebep olmuştur. Tabii ki bu istenmeyen bir durumdur. Bu daha çok ortalama hızın çok yüksek olduğu Avrupa'da görülmüştür. Bunu

engellemek için şemsiye hücreler oluşturulmuştur (Şekil 2.4). Bir şemsiye hücrede, altında bulunan mikro hücrelerin içinde yayımlanandan daha yüksek seviyede güç yayımlanır (Ericsson, 1998b). En önemlisi ise şemsiye hücrede yayımlanan gücün farklı frekans da olmasıdır. Böylece mobil, yüksek bir hızla seyahat ederken, sistem tarafından "hızlı hareket eden" olarak algılanacak ve şebeke içinde birçok kere farklı hücre tarafından ele alınmaktan kurtulacaktır. Bu duruma ise şemsiye hücre el atacaktır. Çünkü şemsiye hücre daha büyüktür ve mobil yüksek hızda olsa bile bu şemsiyenin dışına hemen çıkamaz. Böyle yüksek hızda seyreden bir mobil, yayılım karakteristiklerinden, çok sayıda farklı hücreler veya sistem tarafından ele alınma talebinden fark edilir. Bu hücrede mobil çok uzun bir zaman dilimi için kalabilir, böylelikle şebekenin iş yükü azaltılmış olur.

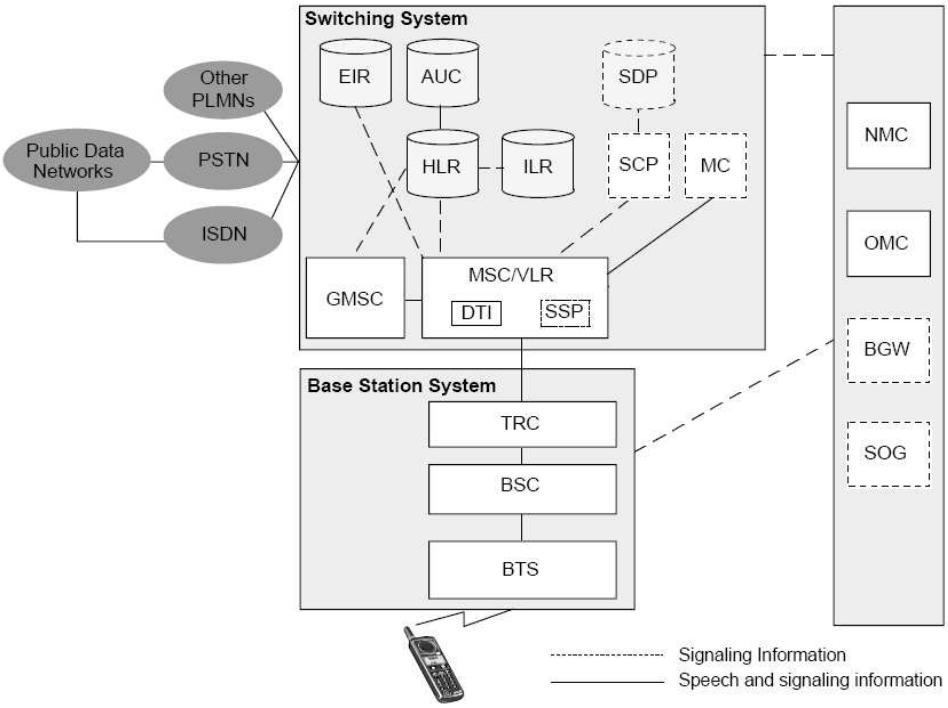


Şekil 2.4 - Şemsiye Hücreler (Ericsson, 1998b)

2.2 GSM Şebekesi

GSM şebekesi temel olarak, Anahtarlama Sistemi (SS – Switching System) ve Baz İstasyon Sistemi (BSS - Base Station System) olmak üzere iki bölüme ayrılır (Ericsson, 1998a).

Bunlardan her biri, bütün sistemin fonksiyonlarının gerçekleştirildiği bir takım fonksiyonel üniteler içerir. Bu fonksiyonel üniteler, değişik donanım gereçleri ile gerçeklenirler (Şekil 2.5).



Şekil 2.5 - GSM Şebekesi Sistem Modeli (Ericsson, 1998a)

Anahtarlama Sistemi (SS) şu fonksiyonel üniteleri içerir;

- Mobil Servisler Anahtarlama Merkezi (MSC - Mobile Services Switching Center)
- Ziyaretçi Yer Kaydedicisi (VLR - Visitor Location Register)
- Dahili Yer Kaydedicisi (HLR - Home Location Register)
- Doğrulama Merkezi (AUC - Authentication Center)
- Cihaz Kimlik Kaydedicisi (EIR - Equipment Identity Register)

Baz İstasyon Sistemi (BSS) ise şu üniteleri içerir;

- Baz Alıcı - Verici İstasyonu (BTS - Base Transceiver Station)
- Baz İstasyon Denetleyicisi (BSC - Base Station Controller)

2.2.1 Hücreler

Önceki bölümlerde belirtildiği gibi, sistem bitişik radyo hücreleri ağı şeklinde tasarlanır ve bu hücreler birlikte tüm servis alanını kapsarlar.

2.2.2 Mobil İstasyon Terminal Cihazı (MS - Mobile Station)

Hücresel şebekenin en çok bilinen ünitesi mobil istasyonlardır. Güç ve uygulama açısından dikkate alınırsa farklı tipte mobil istasyonlar mevcuttur. SIM ve mobil cihaz birlikte mobil istasyonu oluştururlar (Ericsson, 1998c).

Sabit mobil istasyonlar, arabanın içine kalıcı olarak yerleştirilir ve maksimum izin verilen RF çıkış gücü 20W'dır. Portatif üniteler (çanta telefonları) 8W ve elle taşınabilir üniteler 2W'a kadar güç çıkarırlar. 1993'den bu yana üretilen mobiller ile GSM sistem daha cazip hale gelmiştir. Elle taşınabilen üniteler, hacimce oldukça küçüktürler.

2.2.3 Abone Kimlik Modülü (SIM - Subscriber Identity Module)

SIM, mobil aboneye bir kimlik tedarik eder. SIM olmadan, acil aramalar hariç, mobil işlevini göremez. SIM, kredi kartı büyüklüğünde, içinde kurulmuş çipi olan plastik bir karttır. “Smart Card” olarak da adlandırılır. SIM kart, eğer mobil kullanılmak isteniyorsa mobil içine yerleştirilmelidir. Elle taşınabilir cep telefonları için, kredi kartı büyüklüğündeki SIM kartın yerine daha küçük olan “plug-in SIM“ geliştirilmiştir. Belirli abone parametreleri SIM kartta yüklüdür. Bunlarla beraber abone tarafından kullanılan kişisel veriler de kişisel telefon numaraları gibi bu kart içindeki çipte mevcuttur. SIM kart aboneyi tanıtır ve bir telefonu kişiselleştirdiğine göre, sadece SIM kartı alarak yurt dışına çıkmak mümkündür. Bu durumda gidilen yerde bir mobil telefon kiralayarak ve kişi kendi SIM kartını takarak, o cihazı kendi numarasından telefon ediyormuş gibi kullanabilir. Mali yükümlülük, kişinin bağlı olduğu numaraya ait olur. Aynı zamanda bu kişiye, kişinin abone numarası çevrilerek de ulaşılabilir (Harputluoğlu, 2000).

Şebekeden alınan kısa mesajlar da bu kartta saklanır. Kartın güvenliği için dört basamaklı bir şifre konulmuştur. Bu şifre PIN (Personal Identification Number) olarak adlandırılır. PIN kartta yüklüdür ve üç kere yanlış girilirse, kart kendini bloke eder. Bu durumda kart ancak sekiz basamaklı bir şifre ile çözülebilir. Buna da PUK (Personal Unblocking Key) denir ve PUK da kartta yüklüdür.

2.2.4 Baz Alıcı - Verici İstasyonu (BTS - Base Transceiver Station)

Her bir hücre bir grup radyo kanalını işleten Baz Alıcı - Verici istasyonuna (BTS) sahiptir. Bu kanallar girişimi önlemek amacıyla komşu hücrelerde kullanılan kanallardan farklı tasarlanmışlardır. BTS, mobilin şebekeye arayüzüdür. Bir BTS genellikle hücrenin ortasına yerleştirilir. BTS'den çıkan güç hücrenin gerçek boyutunu belirler. Bir baz istasyon, her biri ayrı RF kanalı temsil eden alıcı-vericilere sahiptir (Ericsson, 1998c).

2.2.5 Baz İstasyon Denetleyicisi (BSC - Base Station Controller)

Bir grup BTS, bir BSC ile kontrol edilir. Bu baz istasyonların sayısı üreticiye bağlıdır ve birkaç onlar veya birkaç yüzler mertebelerinde olabilir. Baz istasyon denetleyicisinin (BSC) en önemli görevleri arasında güç kontrolü, frekans idaresi ve BTS'lerin kontrolü sayılabilir. BSC donanımı, BTS gibi aynı bölgeye veya kendi başına bir bölgeye yerleştirilebileceği gibi Mobil Servisler Anahtarlama Merkezinin (MSC) bölgesine de yerleştirilebilir, BSC ve BTS, fonksiyonel olarak bir bütündür, buna da Baz İstasyon Sistemi veya Baz İstasyon Alt Sistemi (BSS - Base Station System) adı verilir (Harputluoğlu, 2000).

2.2.6 Mobil Servisler Anahtarlama Merkezi (MSC - Mobile Services Switching Center)

Belli bir sayıda temel istasyon denetleyicisine (BSC) bir Mobil Servisler Anahtarlama Merkezi (MSC) hizmet eder. MSC'ler PSTN, ISDN, PLMN ve birçok özel şebekelerle yapılan karşılıklı görüşmeleri kontrol ederler.

2.2.7 Geçit Mobil Servisler Anahtarlama Merkezi (GMSC - Gateway Mobile Services Switching Center)

GMSC, hücresel ağın PSTN'e arayüzüdür. Eğer sabit ağdaki bir abone herhangi bir GSM abonesini aramak isterse, PSTN aramayı "Gateway" diye adlandırılan giriş yerine bağlar. Gateway, genellikle bir MSC'de gerçeklenir ve bu MSC, GMSC - Gateway MSC olarak adlandırılır (Harputluoğlu, 2000).

GMSC, herhangi bir MSC olabilir. GMSC, araştırılan mobil istasyonun yerini bulacaktır. Sahip olduğu kayıtlarla sabit ağdan gelen aramayı BSC ve BTS yoluyla mobil istasyona gönderir. GMSC bu göndermeyi Dahili Yer Kaydedicisine (HLR) sorarak yapar. HLR, GMSC'ye adresi bildirir ve GMSC aramayı, aranılan mobil istasyonun bulunduğu MSC'ye yönlendirir. Arama, yönlendirilen MSC'ye ulaştığında, Ziyaretçi Yer Kaydedicisi (VLR) mobil istasyonun nerede olduğunu detayları ile bilecektir, çünkü VLR, HLR'dan gerekli verileri almıştır. Böylece aramanın gönderildiği MSC, aramayı o tarafa doğru anahtarlayacaktır.

Şebekenin boyutuna bağlı olarak, bir operatör, sabit şebekeye bir arayüz kullanabilir. Bu da birkaç GMSC veya sadece bir GMSC kullanarak olur. Eğer sabit şebekedeki trafik, GMSC'lerin tedarik edebileceğinden daha fazla mesaj değişimi (exchange) gerektiriyorsa, sabit şebekeye erişimi olmayan ek MSC'ler bir araya konuşlandırılmalıdır. Eğer daha fazla mesaj değişimi gerektirmezse, GMSC ile MSC aynıdır. Aralarındaki en önemli fark, MSC'nin HLR ile ilgisinin olmaması, yani GMSC'nin HLR ile ilgisinin bulunmasıdır (Ericsson, 1998a).

2.2.8 İşletme ve Bakım Merkezi (OMC - Operation and Maintenance Center)

OMC'nin hem (G)MSC'ye, hem de BSC'ye erişimi vardır. Şebekeden gelen hata mesajlarını ele alır. BTS ve BSC'nin trafik yükünü kontrol eder. OMC, BSC yoluyla BTS'i düzenler ve operatörün sisteme bağlı parçaları kontrol edebilmesini sağlar. Hücreler küçüldükçe ve baz istasyonlarının sayısı arttıkça, gelecekte bireysel istasyonları kontrol

etmek mümkün olmayacaktır. Bu olay alıcı-verici kalitesi dengesini bozabilecektir, çünkü bu kontroller belli bir düzende sürekli yapılmaktadır. Dolayısıyla uzaktan kontrollü, yerinde bakım olayını sağlayan sistemler kurmak maliyeti düşürmek açısından önemlidir, fakat sistem kalitesini de muhafaza etmelidir. Bu, BTS'deki "self-test" fonksiyonları ile desteklenir (Ericsson, 1998a). Bu özelliklerin sisteme sağlanması üreticiyle alakalıdır.

2.2.9 Dahili Yer Kaydedicisi (HLR - Home Location Register)

Yukarıda bahsedilen üniteler, mobil ile sabit bir ağ abonesi arasındaki konuşma bağlantılarının geçtiği ünitelerdir. Eğer bir mobil istasyona arama yapılması olayı söz konusu olmasaydı, daha fazla donanıma ve teçhizata ihtiyacımız olmayacaktı. Esas problem MS ile sonlandırılan bir görüşme yapılmak istendiğinde ortaya çıkmaktadır. Bu durumda mobil istasyonun yeri önemlidir ve mobil istasyonun izini takip edecek bir takım veri tabanlarına ihtiyaç vardır. Bu veri tabanlarından en önemli olanı Dahili Yer Kaydedicisi'dir (HLR). Herhangi bir kişi bir GSM operatöründen abonelik aldığında o operatörün HLR'ında kaydedilir. HLR, ilgili (G)MSC bölgesine ait bütün abonelerin kullanıcı veri ve kimlik bilgilerini saklar. Bunlar, bir kullanıcının Uluslararası Mobil Abone Numarası (IMSI - International Mobile Subscriber Number), doğrulama parametreleri ve abonenin kabul edilen bütünleyici servisleri gibi kalıcı veriler ve bazı geçici verilerdir. SIM'deki geçici veriler şunları içerirler;

- Abonenin o anda bulunduğu VLR
- Eğer abone yönlendirme seçerse gelen aramaların hangi numaraya aktarılacağı
- Güvenlik ve şifreleme için gerekli bazı parametreler

IMSI, SIM kartta kalıcıdır ve GSM sistemde aboneyi tanımlayan önemli bilgilerden biridir. IMSI'nin ilk üç basamağı Mobil Ülke Kodunu (MCC - Mobile Country Code), daha sonraki iki basamak Mobil Şebeke Kodunu (MNC - Mobile Network Code) tanımlar. On basamağa kadar olabilen Mobil Abone Kimlik Numarası (MSIC – Mobile Subscriber Identification Number), IMSI'yi tanımlar.

Örneğin IMSI: 262024542751010

Bu numara Almanya'dan bir aboneyi tanımlar (MCC=262), bu abonenin aylık fatura ödemesini 02 numaralı operatöre ödediği (MNC=02) anlaşılır. Abonenin şebeke kimlik numarası da 4542751010'dır (MSIC). Ancak PSTN'den ulaşılacak numara IMSI'den tümüyle farklıdır ve 0172 alan kodu ile başlar ve 7 basamaklı abone numarası ile devam eder.

2.2.10 Ziyaretçi Yer Kaydedicisi (VLR - Visitor Location Register)

Ziyaretçi Yer Kaydedicisi (VLR), MSC bölgesinde yerleştirilen bütün mobil istasyonlar hakkında bilgi içeren bir veri tabanıdır (Ericsson, 1998c). Mobil istasyon, yeni bir MSC bölgesine girer girmez o MSC'ye bağlı olan VLR, HLR'dan mobil istasyon hakkında bilgi ister. Aynı zamanda HLR, mobil istasyonun bulunduğu bölgenin hangi MSC olduğu hakkında bilgilendirilecektir. Eğer mobil istasyon bir arama yapmak isterse, VLR, her seferinde HLR'a sormadan aramanın sağlanması için gereken tüm bilgileri elde edecektir. VLR, dağıtılmış HLR gibi görülebilir. VLR, aynı zamanda MSC bölgesindeki mobil istasyonun yeri hakkında tam bilgiye sahiptir. VLR, bütün mobillerin ilişkide oldukları (G)MSC'ye yüklenen, konu ile ilgili verileri içerir. Kalıcı veri, HLR'da bulunan veri ile aynı olup, geçici veri biraz farklılık gösterir. Örneğin VLR, TMSI (Temporary Mobile Subscriber Identity - Geçici Mobil Abone Kimliği) bilgisini saklar. Bu bilgi sınırlı zaman aralıkları içindir ve hava arayüzü yoluyla IMSI'nin transmisyonunu önler.

VLR, arama olayının kurulumu esnasında doğrulama prosedürü süresince aboneye özel veriler tedarik ederek (G)MSC'yi destekler. VLR, belli bir abone için o abonenin HLR'ından veri alır. Abone verilerini VLR'a yerleştirme HLR'daki veri trafiğini azaltır. Çünkü her seferinde lazım olduğunda bu verilerin tekrar tekrar istenmesine gerek yoktur. Aşağı yukarı birbirinin aynı olan bu verilerin hem HLR, hem de VLR'a yüklenmesinin diğer sebebi her birinin farklı amaçlara hizmet etmesidir. HLR, PSTN'den bir arama geliyor iken gerekli abone bilgilerini GMSC'ye tedarik etmelidir. VLR, tam zıt bir işlevi yerine getirir, yani bir mobil istasyondan arama geliyor iken gerekli abone bilgilerini ev sahibi

GMSC'ye tedarik eder. Sonuçta hem HLR hem de VLR, GSM sisteminde ki haberleşme akışı açısından çok önemli yerlere sahiptirler.

2.2.11 Cihaz Kimlik Kaydedicisi (EIR - Equipment Identity Register)

Bilindiği gibi SIM ve mobil cihaz birlikte mobil istasyonu oluştururlar. SIM'in bulunmadığı durumda mobil istasyon GSM şebekesine acil haller dışında giriş alamaz. SIM kart, mobil cihazı şebekeye tanıttığından ve başka mobil cihazlarda da kullanılabileceğinden dolayı çalıntı mobil cihazların durumu ne olacak diye akla soru gelebilir. Bunun için cihazın kendine özel donanım kimliği içeren bir veri tabanına ihtiyaç duyulmuştur. Bu da cihaz kimlik kaydedicisidir (EIR).

EIR'ın varlığı GSM sistemine güvenlik sağlar. EIR'da çalıntı cihazlardan başka, donanımında bozukluk olan ve şebekede kullanılmayan mobil cihazların seri numaraları vardır .

Uluslararası Mobil Cihaz Kimliği (IMEI - International Mobile Equipment Identity), sadece belirli bir istasyonun seri numarası değildir. Aynı zamanda üreticiyi ve üretilen ülkeyi de gösterir. Burada amaç her kaydetme ve arama kurulumunda mobil istasyon kimliğinin kontrol edilmesi, sonra IMEI'ye dayanarak mobil istasyonun sisteme erişimini engellemek veya kabul etmektir. Örnek olarak şunu verebiliriz; herhangi bir firmanın ürettiği cihaz, RF kalitesi gibi tavsiye edilen özelliklere yeteri kadar uymuyorsa, bu mobil uyumsuz dalgalar üreteceğinden ve girişime sebep olacağından yasaklanmış demektir ve bu cihaz şebekeden reddedilir.

2.2.12 Doğrulama Merkezi (AUC - Authentication Center)

Doğrulama Merkezi (AUC), HLR ile ilişkilendirilmiştir. AUC, mobil istasyonun doğrulama prosedürü esnasında HLR'a bazı parametreler tedarik eder. AUC, belirli bir abone ile ilgili giriş değerlerini hesaplamak ve istenilen sonuçları HLR'a aktarmak için hangi algoritmayı kullanacağını önceden bilir. Doğrulama prosedürleri ile ilgili algoritmalar

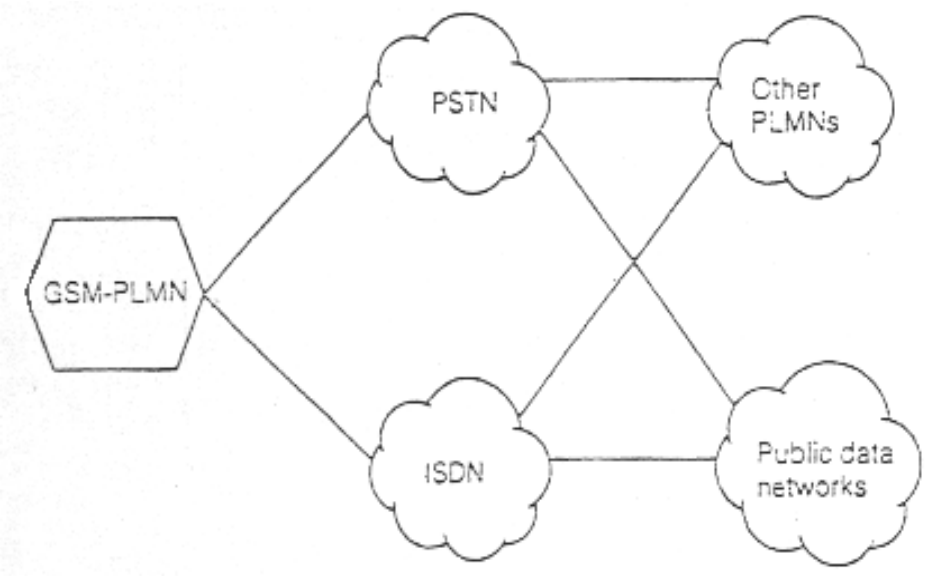
AUC'da yüklüdür ve herhangi bir andaki kötü kullanıma karşı korunurlar.

2.3 Coğrafi Ağ Yapısı

Her telefon ağı, gelen aramaları uygun bir şekilde doğru santrallere ve sonuçta aranılan aboneye ulaştıracak iyi bir yapıya sahip olmalıdır. Eğer bu bir mobil ağ ise, tüm abonelerin hareketliliğinden dolayı bu yapı çok daha fazla önem taşır. Bu açıdan ağ yapısının kurulumu ve işleyişi doğru olmalıdır.

2.3.1 Ağ Bölgesi ve GMSC

Bir GSM/PLMN ağı ile diğer PSTN, ISDN ve PLMN ağları arasındaki linkler, uluslararası veya ulusal transit santraller sayesinde olacaktır (Şekil 2.6). Bir GSM/PLMN ağına gereken aramalar bir veya daha fazla GMSC'ye gönderilir. Bu GMSC, GSM/PLMN için bir "giren transit santral" gibi çalışır. Bir GSM/PLMN ağında tüm mobil sonlandırmalı aramalar bir GMSC'ye iletilir (Ericsson, 1998a).

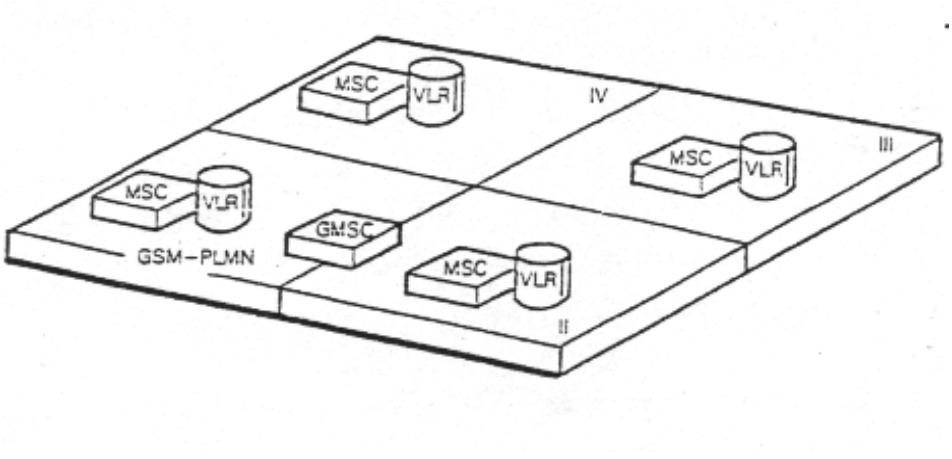


Şekil 2.6 - GSM/PLMN ağı ve diğer yerel ağlar arasındaki linkler

2.3.2 MSC/VLR Servis Bölgesi

Bir MSC bölgesi, tüm ağın tek bir MSC tarafından kaplanan parçasını temsil eder. Bir mobil aboneye aramayı göndermek için, abonenin bulunduğu alandaki MSC dikkate alınır (Harputluoğlu, 2000).

Servis bölgesi, MS'in bulunduğu ve MS'in kayıtlı olduğu VLR'ın bulunduğu bölgedir. VLR, HLR'dan MS hakkında bilgi alır. Ericsson firmasının ürettiği CME 20 sisteminde herhangi bir MSC bölgesi ile o MSC'ye karşılık gelen servis bölgesi aynı bölgeyi temsil eder (Şekil 2.7). Bir başka deyişle her bir MSC, bir VLR ile gerçeklenir (Ericsson, 1998c). Bu gerçekleme üretici firmaya göre değişir, verilen örnek Ericsson içindir.

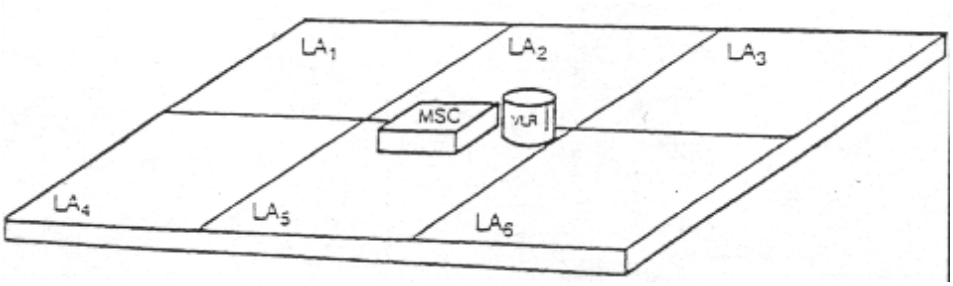


Şekil 2.7 - Ericsson MSC/VLR Servis Alanları

2.3.3 Yerleşim Bölgesi (LA – Location Area)

Her MSC/VLR servis bölgesi birkaç yerleşim bölgesine ayrılmıştır (Şekil 2.8). Yerleşim bölgesi (LA), MSC/VLR servis bölgesinin bir parçasıdır ve mobil istasyonun bu alanda serbestçe gezmesi yerleşim bölgesindeki MSC/VLR kontrolündeki yer bilgisini değiştirmez. Bir yerleşim bölgesi, abonenin bulunması için arama mesajının yayımlandığı bölgedir. Yerleşim bölgesi, birkaç hücreye sahip olabilir ve bir veya daha fazla BSC'ye bağlıdır. Fakat sadece bir MSC/VLR'a sahiptir. Yerleşim

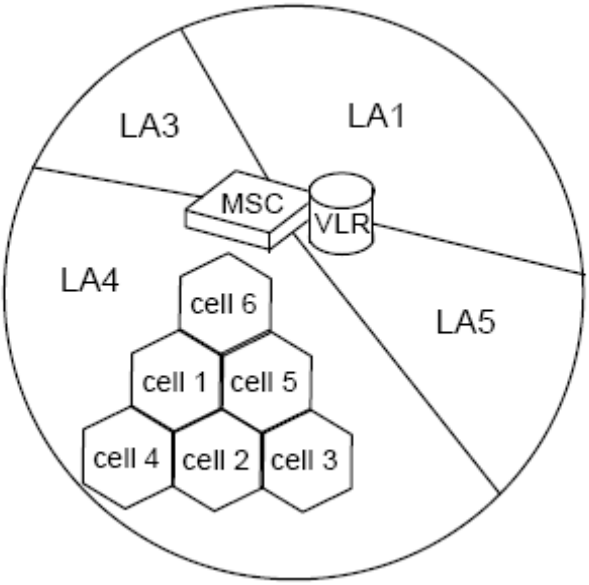
bölgesi, sistem tarafından Yerleşim Bölgesi Kimliği (LAI - Location Area Identity) kullanarak tanınır. Sistem aktif durumdaki aboneyi aramak için kullanır.



Şekil 2.8 - Bir MSC/VLR servis alanının yerleşim bölgelerine bölümü

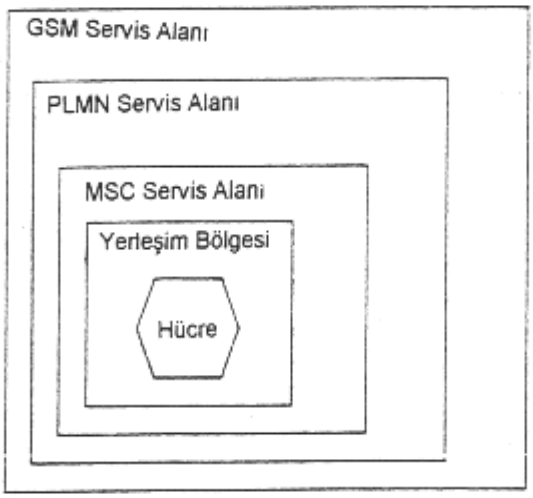
2.3.4 Hücre

Bir LA, belli sayıda hücreden oluşur (Şekil 2.9). Hücreler, şebekenin Hücre Küresel Kimliği (CGI - Cell Global Identity) ile tanıdığı bölgelerdir. MS, Baz İstasyon Kimlik Kodu (BSIC - Base Station Identity Code) vasıtasıyla, aynı taşıyıcı frekansları kullanan hücreleri ayırt eder (Ericsson, 1998b).



Şekil 2.9 - Bir MSC/VLR servis alanının yerleşim bölgelerine ve hücrelere bölümü

Özet olarak, coğrafi yapıyı Şekil 2.10'daki biçimde düşünebiliriz;



Şekil 2.10 - GSM'deki alanlar arası ilişki

2.4 Kaydetme

Mobil istasyon açıldıktan sonra, çok kısa bir zamanda, şebekenin varlığını fark etmek için belirli bir tarama algoritması kullanarak bütün GSM frekanslarını tarar. Şebeke fark edildiğinde, mobil istasyon ya ilerideki sistem enformasyonunu okur ya da temel kanalı okur (Şekil 2.11).

MS	BTS	BSC	(G)MSC	VLR	HLR	AKSİYON
→	→					Kanal talebi
	←					Kanalı aktif hale getirme komutu
	→					Bu komutun tanınması (kabulü)
←	←					Kanal tahsisi
→	→	→				Bölgenin uygun hale getirilmesi talebi
←	←	←				Doğrulama parametrelerinin talebi
→	→	→				Doğrulama parametrelerinin gönderilmesi
			↔			Doğrulama parametrelerinin karşılaştırılması
←	←	←				Yeni bölgenin ve TMSI'ın belirtilmesi
→	→	→				Yeni bölgenin ve TMSI'ın tanınması
			↔			Yeni bölge ve kimliğin VLR'a girilmesi
			←	↔		Yeni bölge ve kimliğin HLR'a girilmesi
←	←					Mobile kanal tahsisi

Şekil 2.11 - Şebekedeki kaydetme prosedürü

Bu enformasyonla, mobil istasyon şebekedeki o an ki pozisyonunu belirtme olanağına sahip olur. Eğer mobil istasyonun yeri en son kapatıldığı yer değilse, bir kaydetme prosedürü (registration procedure) başlar. Şekil 2.11'de kaydetme prosedürü esnasında yapılması gerekenler ve şebekedeki değişik cihazlar ile ilişkilerini tanımlar.

İlk olarak mobil istasyon, şebekeden baz istasyon tarafından belirtilecek bir kanal talep eder. Sonra BSC üzerinden BTS'deki bir kanal aktif hale getirilmeye çalışılır, boş kanal olup olmadığı araştırılır ve varsa bu BSC'ye bildirilir. Boş kanal varsa mobil istasyon alt yapıya bağlanmıştır ve sistemden bölgenin kendisi için uygun hale getirilmesini ister. Bu talep BSC'den (G)MSC'ye aktarılır ve bu arada herhangi bir işlev almadan önce mobil istasyonun doğrulanması istenir. Doğru parametrelerin alınması üzerine (G)MSC, mobili BSC ve BTS üzerinden ve yeni bölgede kabul eder. Daha sonra (G)MSC yeni bölgeyi ve geçici kimliği mobile belirtir (TMSI). Mobil istasyon da bunu tanımak zorundadır. Prosedür bittiğinde kanal BSC'den BTS yoluyla açılır. Eğer sistem hangi mobillerin sistemde mevcut olduğunu bilmek isterse, bu kaydetme işlemi de o kadar başarıyla tamamlanır. Mobil istasyonlar, kapatıldıklarında veya açıldıklarında şebekeyi haberdar ederler. Aslında kaydetme prosedürü şebeke içindeki bilgi akışını limitler ve şebekede sanal bir kontrol sağlar. HLR'ın bildiğini, GMSC zaten bilir ve mobil istasyonun kapalı veya açık olma durumu şebekede genel bilgi halini alır. Eğer bir kişi kapalı bir mobil istasyonu aramak isterse, GMSC hemen aranılan mobilin mevcut olmadığını gösteren bir mesaj sinyali gönderir. Böylelikle mobil istasyonu bölgede tarama olayı boşu boşuna yapılmamış olur. Dolayısıyla sistem daha az meşgul edilmiş ve sistemin dinamik enerjisinden tasarruf edilmiş olur (Harputluoğlu, 2000).

2.5 Arama Kurulumu

Aramanın kurulumundan (call establishment) önce, mobil istasyon açık olmalı ve sisteme kaydedilmiş olmalıdır. İki farklı prosedür vardır. Bunlardan biri "mobil çıkışlı arama" (MOC - Mobile Originated Call), diğeri ise "mobil sonlandırmalı arama" dır (MTC - Mobile Terminated Call). Burada sadece mobil çıkışlı aramalardan bahsedilecektir. Mobil sonlandırmalı arama işlemleri de bunun zıt yönünde benzer işlemler olacaktır.

Mobil çıkışlı arama, GSM sistemde kullanılan işaretleşme hakkında genel bir izlenim edindirecektir. Burada işaretleşmeden kastedilen mesaj değişimidir. Gerçek bir arama başlamadan önce, şebeke ve mobil istasyon arasında on dört farklı mesaj değişimi olur (Ericsson, 1998a).

Mevkiinin uygun hale getirilmesi prosedürüne (Location Update Procedure) benzer bir prosedür ile mobil bir kanal talebi ile başlar. Sistem tarafından kanal belirtilmesi yapılır. Mobil istasyon kanal isteme sebebini sisteme haber verir. Prosedürün devam etmesine izin verilmeden önce mobil kendi gerçekliğini tekrar kanıtlamak zorundadır. Şebeke bir mesaj göndererek gizli dinleyicilere karşı koruma amacıyla mobil istasyonun verilerini şifrelemesini ister. Şifreleme işi, mesajın sadece mobil istasyonun ve BTS'in anlayacağı bir şekilde gönderilmesi demektir. Sonra mobil aramak istediği numarayı gönderir. Arama devam ederken, BSC, BTS yolu ile kullanıcı verilerinin aktarılacağı bir trafik kanalı belirtir. Farklı tipteki mesajlar ve kullanıcı verileri farklı kanallardan giderler. Şekil 2.12 de mobil çıkışlı arama prosedürü görülmektedir.

Bazı kanallar sadece mesaj değişimi için, bazıları ise kullanıcı verilerinin ele alınması içindirler. Aranılan nokta meşgul değilse mobil işaretini (sinyalini) gönderir ve karşı taraf telefonu açtığında bağlantı kurulmuş olur.

MS BTS	Aksiyon
→	Kanal talebi
←	Kanal belirtilmesi
→	Arama kurulumu talebi
←	Doğrulama parametrelerinin talebi
→	Doğrulama parametrelerinin belirtilmesi
←	Şifreleme komutu
→	Şifreleme tamam
→	Arzulanan numarayı gösteren kurulum mesajı
←	Arama devam ediyor, şebeke aramayı arzu edilen numaraya yönlendirir
←	Kullanıcı verilerinin değişimi için kanal belirtilmesi
→	Kanal belirtilmesi tamam, mesajlar trafik kanalında
←	Aranılan numara meşgul değil ve telefon çalıyor
←	Aranılan numaraya bağlantı
→	Bağlantı tamam, arama iki tarafın konuşabilmesi içinde müsait
↔	Konuşma verilerinin değişimi

Şekil 2.12 - Mobil çıkışlı arama kurulumu prosedürü

2.6 Aktarma (Handover/Handoff)

Handover veya Handoff prosedürü, bir mobil istasyonun iki hücre arasında geçiş yaparken konuşmanın devamı için bir araçtır. Bir arama, hücre sınırı geçildiğinde veya mobil istasyon ile belirli bir baz istasyon arasındaki mesafe çok arttığında düşer.

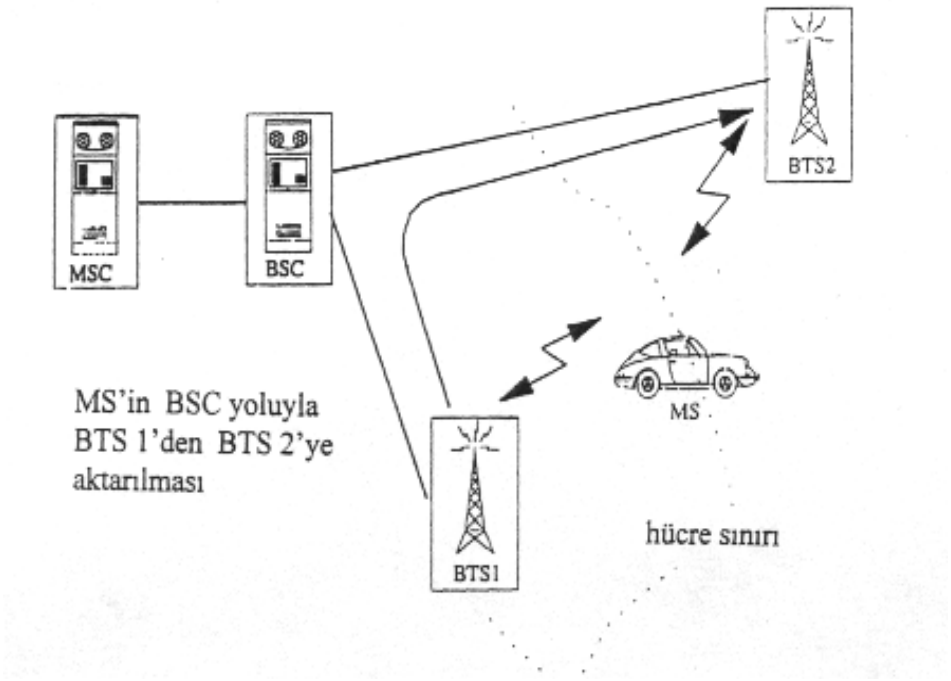
Hücresel bir şebekede, bir hücrenin komşu hücreleri vardır. Böylelikle sistem, mobil istasyonun hangi hücreye geçebileceğini saptayabilmektedir. Bir sonraki hücreyi saptayabilme metodu analog ve sayısal sistemlerde farklılık gösterir. Bu farklılık “Handoff” ve “Handover” sözcüklerinden tespit edilebilir. “Handoff” analog

sistemlerde kullanılmakta iken, “Handover„ ise GSM sisteminden bahsedilirken kullanılır.

Analog sistemlerde, baz istasyon, mobil istasyon ile kendi arasındaki bağlantı kalitesini gösterir. Baz istasyon, bağlantı kalitesinin düştüğünü ve mobil istasyon ile kendi arasındaki mesafenin arttığını fark ederse, komşu hücrelerden mobile olan güç seviyelerini rapor etmesini ister. Mantıklı olanı, mobil için rapor edilen en yüksek güç seviyesi, mobil istasyona en yakın hücrede tespit edilir. Daha sonra şebeke, baz istasyonun yeni hücrede hangi frekans kanalını kullanacağına ve mobil istasyonun hangi uygun frekansa senkron edileceğine karar verir. Son olarak mobil istasyon kanal değişikliği için şartlandırılır. Handoff prosedüründe mobil istasyon oldukça pasif kalır. Bütün ölçümler ve ölçümlerden sonra gelen işler baz istasyonlarda ve şebekede yapılır. Hücre bölgeleri, kullanımda olan değişik kanallardaki farklı mobil istasyonların güç seviyelerini ölçmek için ölçme alıcısı ile donatılmışlardır (Harputluoğlu, 2000).

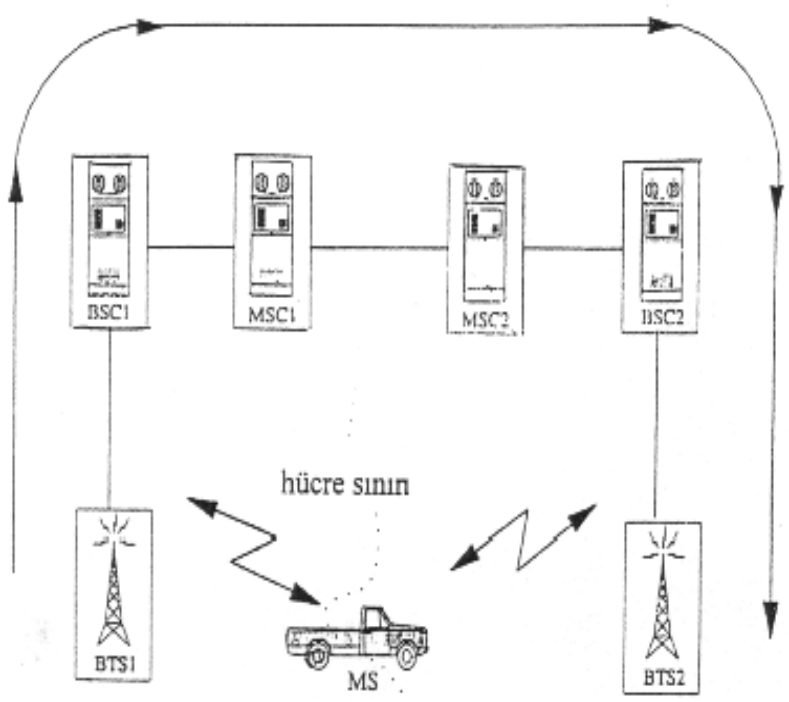
GSM sistemindeki durum farklıdır. Mobil istasyon sürekli komşu hücrelerde algılanan güç seviyelerini göstermelidir. Baz istasyon mobil istasyona güç ölçümlerini yapması için baz istasyonları kanallarının listesini verir. Bu liste temel kanalda gönderilir. Bu temel kanal, mobil açıldığında frekansın senkron olduğu kanaldır ve birinci kanaldır. Mobil istasyon, kalite için, içinde bulunulan hücrenin güç seviyesi ölçümlerine devam eder. Ayrıca bu ölçümler komşu hücrelerin güç seviyeleri için de aynı şekilde yapılır. Ölçüm sonuçları periyodik olarak ölçüm raporuna yerleştirilerek baz istasyona geri gönderilir. Baz istasyon, mobil istasyona olan bağlantının gücü ve kalitesi üzerinde ölçüm yapıyor da olabilir. Eğer ölçümler, bir aktarma yapılması gerektiğini gösteriyorsa, aktarma için en uygun baz istasyonu daha önceden tespit edilmiş olduğundan hiç gecikme olmadan bu aktarma gerçekleştirilerek sorun çözülür.

GSM sistemi farklı tipte aktarmalar seçer. Mobil istasyonun hangi tipte bir hücre sınırını geçtiğine bağlı olarak, yeni hücrede, mevcut bir kanal sağlamak için bu aktarma işinin kontrol edilmesi lazımdır. Eğer aktarma bir BSC alanı içinde gerçekleştirilecekse, aktarma MSC'ye başvurmadan BSC tarafından ele alınabilir. Bu şekilde bir aktarma BTS'ler arası basit aktarma olarak adlandırılır (Şekil 2.13).



Şekil 2.13 - BTS'ler arası aktarma (Harputluoğlu, 2000)

Eğer mobil istasyon bir BSC sınırından geçiyorsa, bu durumda konuşmada düzgün geçiş sağlanması için MSC bunu kontrol eder. Bu iki farklı MSC arasındaki aktarma için de devam edebilir. MSC'ler arası geçişteki tek fark, mobil ileride ikinci MSC tarafından ele alınsa da, ilk MSC hala arama yönetiminin kontrolünü muhafaza eder (Şekil 2.14).



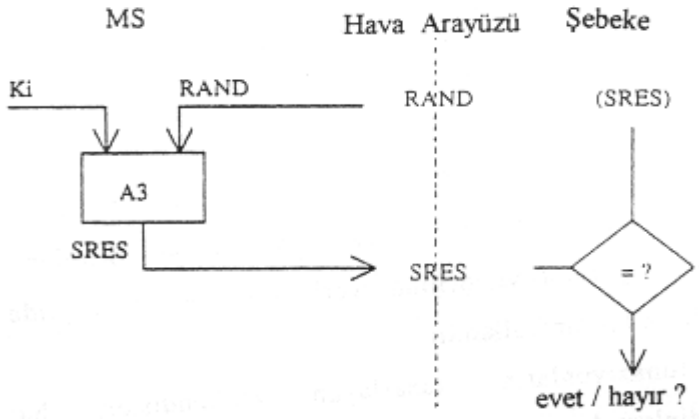
Şekil 2.14 - MSC'ler arası aktarma (Harputluoğlu, 2000)

Teorik olarak, iki ülkenin politik sınırları arasında aktarma yapmak mümkündür. Bu özellik için herhangi bir teknik kısıtlama yoktur. Farklı serbest dolaşım anlaşmalarından dolayı, hiç bir şekilde bir telefon aramasını başlatmak mümkün değildir. Örneğin Almanya'dan İsviçre'ye geçince aboneler yeni yabancı şebekede kayıtlarını yaptırmak zorundadırlar.

2.7 Güvenlik Parametreleri

2.7.1 Doğrulama (Authentication)

Doğrulama prosedürü abonelerin SIM kartlarının geçerliliğini kontrol eder. Doğrulama, “A3” diye adlandırılan ve SIM kart ile doğrulama merkezinde (AUC) yüklü olan bir doğrulama algoritmasına dayanır (Şekil 2.15).



Şekil 2.15 - Doğrulama İlkesi

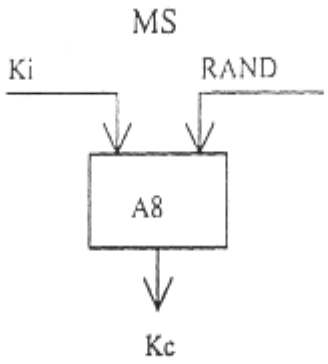
A3 algoritması iki giriş parametresi kullanır; biri sadece şebekede ve SIM kartta yüklü olan doğrulama anahtarıdır (Ki). İkinci değer ise, hava arayüzünde mobil istasyona gönderilen rastgele üretilmiş numaradır (RAND - Random Generated Number). Mobil istasyon, A3 algoritması için bir giriş değeri olan rastgele üretilmiş numarayı SIM'e geçer. Sonuç olan SRES (Signed Response), mobil istasyondan hava arayüzü yoluyla şebekeye gönderilir. Çünkü SRES değeri, doğrulama merkezinden hesaplanan değerle şebekede karşılaştırılır. Doğrulama parametreleri (RAND ve SRES), doğrulama merkezinin kullanımı için HLR ve VLR'da saklıdır. Eğer HLR veya VLR'da bu parametreler tüketilirse doğrulama merkezinden yenileri istenir. Tükenme durumu ise, her arama kurulumunda veya kaydetmede bu parametrelerin ıskartaya çıkma ihtimaline bağlıdır. Bu güvenlik özelliğinin bir önemli noktası, konu ile

ilgili bu parametrelerin (A3 ve Ki) güvenli yerlerde saklanması ve hava arayüzü ile asla gönderilmemesidir (Ericsson, 1998a).

2.7.2 Şifreleme

Sayısal transmisyon verinin şifrelenmesi için uygundur. Çünkü bit dizisi, hava arayüzünün iki tarafından da belirli bir metot ile gönderilmektedir. GSM sistem, işareti ve kullanıcı verisini korumak için böyle bir şifreleme metodu kullanır. Bir tarafta şifrelenen veri sadece diğer tarafta deşifre edilebilir. “A5” diye adlandırılan şifreleme algoritması veri dizisini şifreler ve orijinal veri dizisinin tekrar elde edilmesi için aynı algoritma tekrar kullanılır.

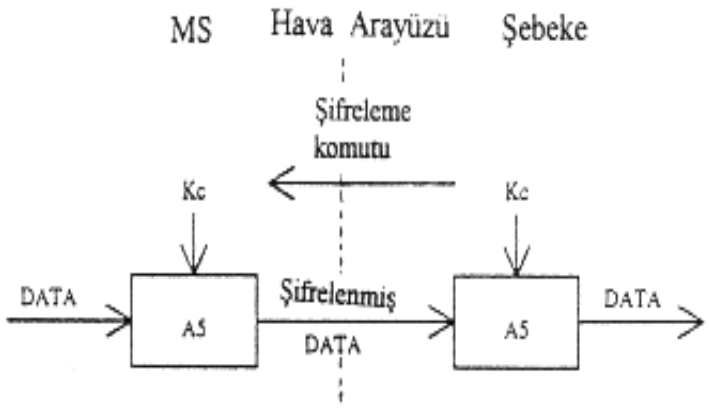
Şifreleme fonksiyonlarını tasarlayan mühendisler, bu algoritmanın gizli dinlemelere karşı çok iyi korunduğunu belirtmektedirler. Bu algoritma “Kc” diye adlandırılan özel bir anahtar istiyor. Bu anahtar ise şebeke tarafından verilen rastgele numaradan (RAND) hesaplanıyor. Rastgele numara doğrulama prosedürü için kullanılan numaradır. Bu hesaplama, RAND ile önceden bahsettiğimiz doğrulama anahtarı Ki'nin A8 kodu ile adlandırılan bir algoritmaya tabi tutulmasıyla oluyor (Şekil 2.16).



Şekil 2.16 - Şifreleme anahtarı Kc’nin hesabı

A8 algoritması SIM kartta yüklü olarak bulunmaktadır. Mobil cihaz A3 ve A8 hakkında hiçbir bilgiye sahip değildir. A8 algoritması ile hesaplanan Kc, A5 algoritmasında yani veriyi şifreleme veya deşifre etmede kullanılır (Şekil 2.17).

Şifreleme prosedürünün başlaması için, şebeke mobil istasyona şifrelemenin başlatılması komutunu verir. Artık bundan sonra mobilden şifrelenmiş olarak çıkan veri hava arayüzünden şebekeye iletilir ve şebeke bu verileri deşifre ederek uygun ünitelere yönlendirir.



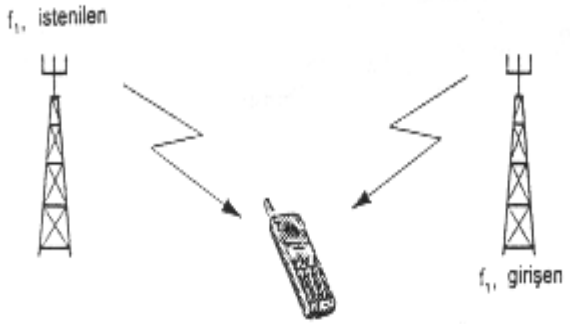
Şekil 2.17 - Şifrelemenin başlatılması ve yapılması

3 SAYISAL RADYO TRANSMİSYONU

Arabalarında seyahat eden herkes farkına varmıştır ki, seyahat esnasında bir radyo yayını dinlerken alınan sinyal kalitesi zaman zaman değişir. Örneğin bir tünele veya iki tepe arasına girerken olduğu gibi. Bu etkiye gölgelenme (shadowing) adı verilir ve kablosuz dünyada ilgilenilmesi gereken birçok can sıkıcı gerçeklerden biridir (Ericsson, 1998d).

Bu bölümde hücresel radyo ortamının temel problemleri ve bunlarla ilgili bazı ölçümler ele alınacaktır. Ek olarak, en genel biçimde sayısal haberleşme ilkeleri de anlatılacaktır.

Problemlerin çoğundaki en genel faktör, istenilen sinyalin çok zayıf olmasıdır. Bu zayıflık, rastgele (ısıl) gürültü veya girişim (interference) sinyalleri ile karşılaştırıldığında çok daha fazla önem kazanır. Böyle bir sinyal, istenilen sinyalin alındığı kanal üzerinde gelen, istenmeyen sinyal olarak tanımlanabilir. Örneğin bu sinyal, ilişkili olunan verici ile aynı frekansta çalışan ve ona çok uzak olmayan bir başka vericiden karışan sinyal olabilir (Şekil 3.1).



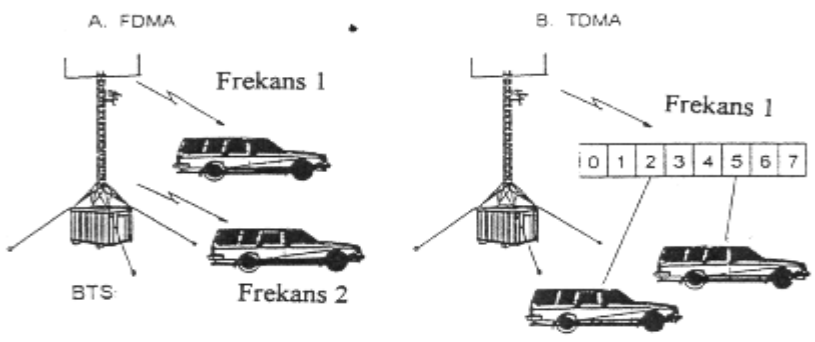
Şekil 3.1 - Girişen sinyal

Bu gerçeklere dayanarak denilebilir ki, bütün frekansların tekrar tekrar kullanıldığı bir sistem olan hücresel sistem, gürültü değil girişimden dolayı sınırlandırılır.

3.1 Zaman Bölmeli Çoklu Erişim (TDMA - Time Division Multiple Access)

Sıradan radyo yayınlarında FDMA (Frequency Division Multiple Access - Frekans Bölmeli Çoklu Erişim) metodu kullanılır. Böylelikle her kanala belirli bir frekans bandı tahsis edilir. Eğer başka bir kanalı dinlemek istiyorsanız alıcının frekansını başka kanala ayarlamalısınız. Bu teknik, analog hücresel sistemlerde kullanılır. Şöyle ki; bir hücredeki her arama bir frekans bandı kullanır. Eğer dubleks, yani iki yönlü arama ise iki bant kullanılır. Belirli bir arama için belirli bir bant kullanıldığından, bu frekans bandı başka bir arama için kullanılamayacaktır.

GSM'de TDMA tekniği kullanılmaktadır ve her frekans bandı için sekiz zaman aralığı bulunur (Ericsson, 1998d). TDMA ve FDMA arasındaki fark Şekil 3.2'de gösterilmiştir. (A)'da her konuşan mobile tahsis edilmiş bir frekans bandı (taşıyıcı frekansı) ilkesi ile uygulanan FDMA, (B)'de ise aynı frekans bandını kullanan sekiz zaman aralığında, sekiz farklı mobilin konuşabileceği TDMA sistemi görülmektedir.



Şekil 3.2 - (A). FDMA (B). TDMA

Dikkat edilmelidir ki, şekillerde tek yön gösterilmiştir. Zıt yönde ise buna uygun gelen frekanslar/zaman aralıkları olmalıdır.

3.2 Transmisyon Problemleri

3.2.1 Yol Kaybı

Yol kaybı, mobil istasyon ile baz istasyon arasındaki mesafe artışıyla artar ve işaretin zayıflamasıdır. Alıcı (Rx) ve verici (Tx) antenleri arasında engeller yoktur. Serbest uzay durumu için verilen bir anten ile ilgili olarak, alınan güç yoğunluğunun Tx ve Rx antenleri arasındaki "d" mesafesinin karesiyle ters orantılı olduğunu söylemek mümkündür. Bir de alınan güç, transmisyon frekansı "f" in karesiyle de ters orantılıdır. Uzay zayıflaması güç kaybı için sonuç olarak;

$$Ls \sim d^2 f^2$$

$$Ls\ [dB] = 33{,}4\ (dB) + 20 \log (f\ Mhz) + 20 \log (d\ km)$$

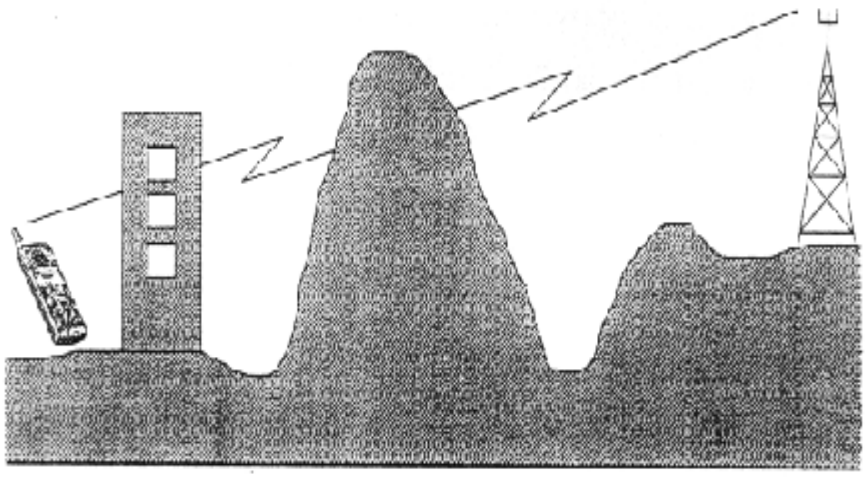
33,4 oranın bir sabiti olarak verilir.

Daha yüksek frekanslar daha fazla zayıflatmaya sebep olurlar. İdeal olmayan şartlarda bu zayıflama d 'nin dördüncü kuvveti ile gerçeklenir (Ericsson, 1998b).

3.2.2 Zayıflama

3.2.2.1 Log-Normal Zayıflama

Aslında mobil cihazlar nadiren engellerin olmadığı ortamlarda kullanılır. Çoğunlukla tepelerin ve binaların olduğu yerler iletim ortamı durumunda olur. Bu da sinyalin gücünü zayıflatan gölgeleme etkisini ortaya çıkarır. Yani sinyal yer yer zayıflayacaktır. Zayıflamadan dolayı sinyal gücünde değişiklik meydana gelir. Minimum noktalara "zayıflama dip noktaları„ denir. Gölgeleme etkilerinin meydana getirdiği bu zayıflamaya "log-normal zayıflama„ denir ve sinyal gücünün logaritmasını alırsak bu zayıflama ortalama değerin etrafında normal dağılım gösterir (Şekil 3.3). İki zayıflama dip noktaları arasındaki mesafe tipik olarak 10–20 metredir.

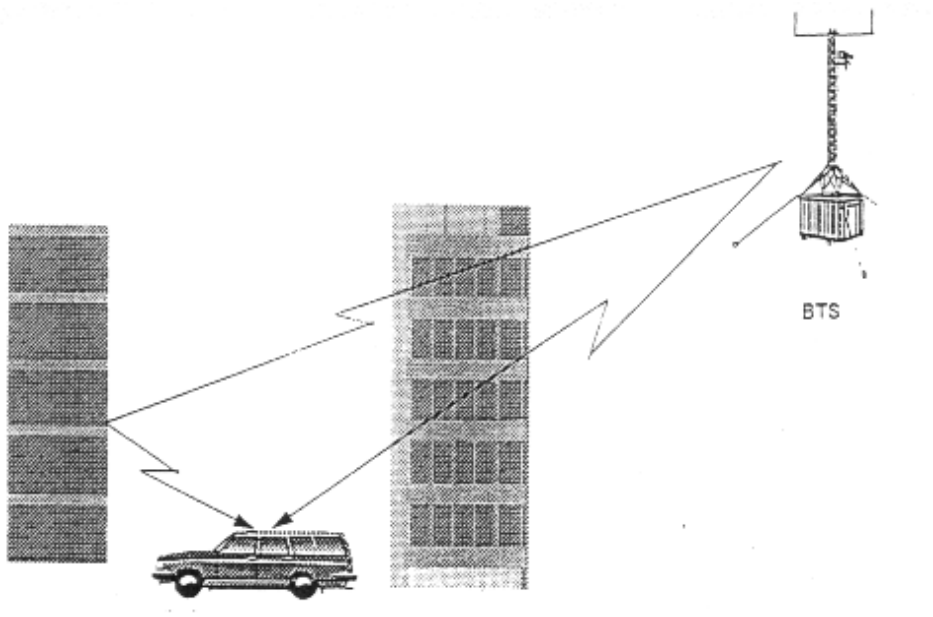


Şekil 3.3 - Log-normal zayıflama

3.2.2.2 Rayleigh Zayıflaması

Mobil telefonların güncelliğinin her geçen gün artması dolayısıyla nüfusun çok olduğu yerlerde abone sayısının yüksek olduğunu ve sürekli artacağını kestirmek hiç de zor değildir. Mobillerin şehirlerde kullanılması bozucu bir etki olan "çoklu yol" veya "Rayleigh zayıflaması" olarak adlandırılır. Bu durum sinyalin Tx anteninden çıktıktan sonra Rx antenine ulaşırken birden fazla yol almasıyla oluşur. İşaret sadece Tx anteninden çıktığı doğrultudan alınmaz, çıktığı noktadan birçok farklı noktalara gider. Antenler arasında görüş hattı yoktur. Sinyal birçok engellerden, örneğin binalardan, yansıyarak mobil istasyona ulaşır (Şekil 3.4).

Bu da demektir ki, alınan sinyal, sadece fazı farklı ve biraz da genliği farklı aynı sinyallerin toplamı olacaktır. Eğer sinyaller vektör olarak toplanırsa sinyal gücü sıfıra düşecektir. İki zayıflama dip noktası arasında geçen zaman, hem transmisyon hızına hem de mobil hızına bağlıdır.



Şekil 3.4 - Rayleigh zayıflaması

Rayleigh zayıflamasına göre iki dip nokta arasındaki mesafe dalga boyunun yarısı kadardır. 900 MHz için bu mesafe 17 cm civarında hesaplanır. Böylece, eğer 50 km/h hızla hareket eden bir mobil için iki dip nokta arası zaman şöyle olacaktır;

V = 50 km/h = 13,89 m/s ≈ 14 m/s

λ = c / f = 3 x 10^8 / 900 x 10^6 = 0,3m

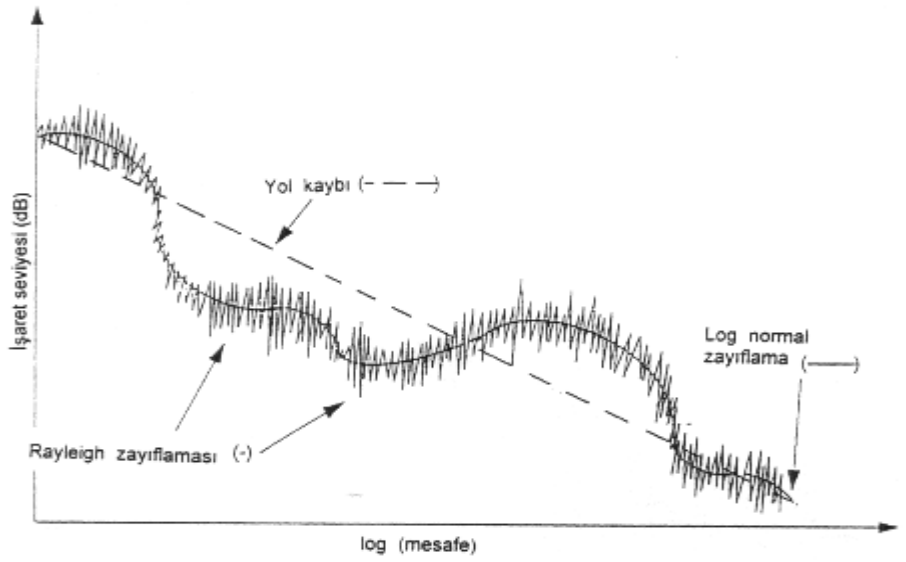
(λ / V) / 2 = 10,7 ms

V: hız λ : dalga boyu

1800 MHz için bu hesaplanan zaman yarıya düşer.

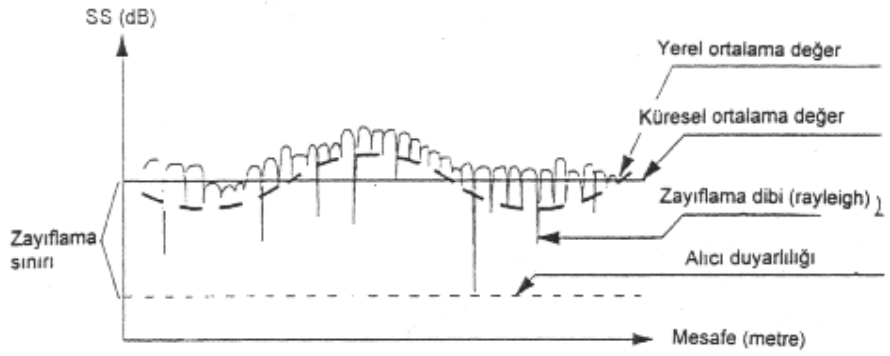
3.2.2.3 Toplam Zayıflayan Sinyal

BTS Tx anteninden uzaklaşıldığında mobil istasyonun Rx anteninde muhtemel sinyal gücü gösterimi Şekil 3.5 ile verilmiştir.



Şekil 3.5 - Mesafe ile Rx sinyal gücü ilişkisi

Tx anteninden belirli bir d kadar mesafe uzaklıkta alınan sinyal Şekil 3.6 'daki gibi olacaktır.

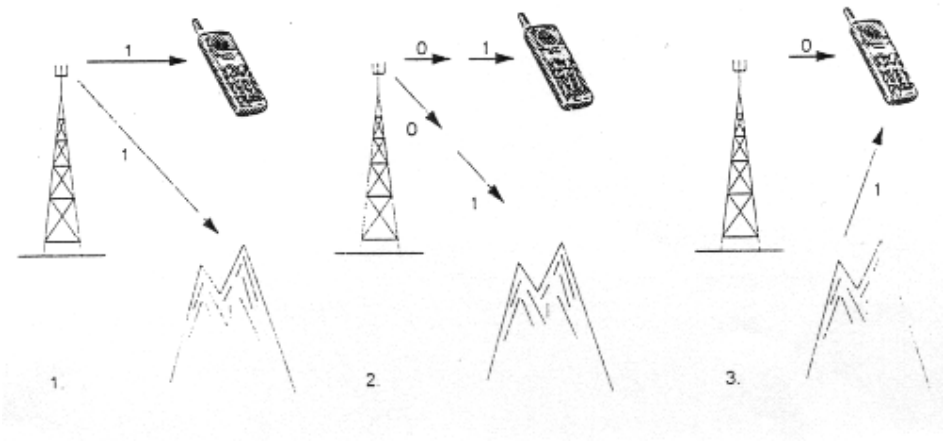


Şekil 3.6 - Rx sinyal gücü

Belirli bir çıkış için istenilen en küçük sinyal gücü değeri alıcının duyarlılığı anlamındadır. Diyelim ki, Tx anteninden gönderilen bilginin sezilebilmesi için X watt'lık bir güç almamız gereksin. Bu durumda sinyal gücü X watt'ın altına düşerse bilgi kaybolacaktır. Dolayısıyla sistemin sadece sinyal gücünün küresel ortalama değerine göre planlanamayacağı çok açıktır. Zayıflamaya karşı önlemlerin alınması gerekir ve "zayıflama aralığı" diye bir terim tanımlanır. Eğer kesintisiz bir transmisyon yoluna sahip olmak istiyorsak, küresel ortalama değerin alıcı duyarlılığının üzerinde olması gerekir. Şekil 3.6'dan da görülebileceği gibi alıcı duyarlılığının seviyesi, en derin zayıflama dip seviyesinin biraz altındadır. Bu durumda zayıflama aralığını küresel ortalama değer ile alıcı duyarlılığı arasındaki fark olarak tanımlayabiliriz.

3.2.3 Zaman Ayrılması

Sayısal transmisyon zaman ayrılması olarak adlandırılan bir başka problemi de beraberinde getirir. Bu problemin merkezinde, çoklu yol zayıflamasına zıt olarak, Rx anteninden çok uzaktaki bir nesneden gelen yansıyan işaretler vardır (Şekil 3.7).



Şekil 3.7 - Zaman ayrılması

Zaman ayrılması, simgeler arası girişim (ISI - Inter Symbol Interference) olayına sebep olur. ISI, sonuç simgelerin birbirine girişmesi ve alıcı tarafın hangi gerçek simgeyi sezeceğine karar vermesinin zorlaşması anlamına gelir. Buna bir örnek Şekil 3.7'de gösterilmiştir.

Eğer yansıyan sinyal direkt giden sinyalden tam olarak bir bit geç giderse, bu durumda alıcı direk giden dalgadan “0” algıladığı gibi yansıyan dalgadan da “1” algılar. “1” simgesi “0” simgesi ile girişir.

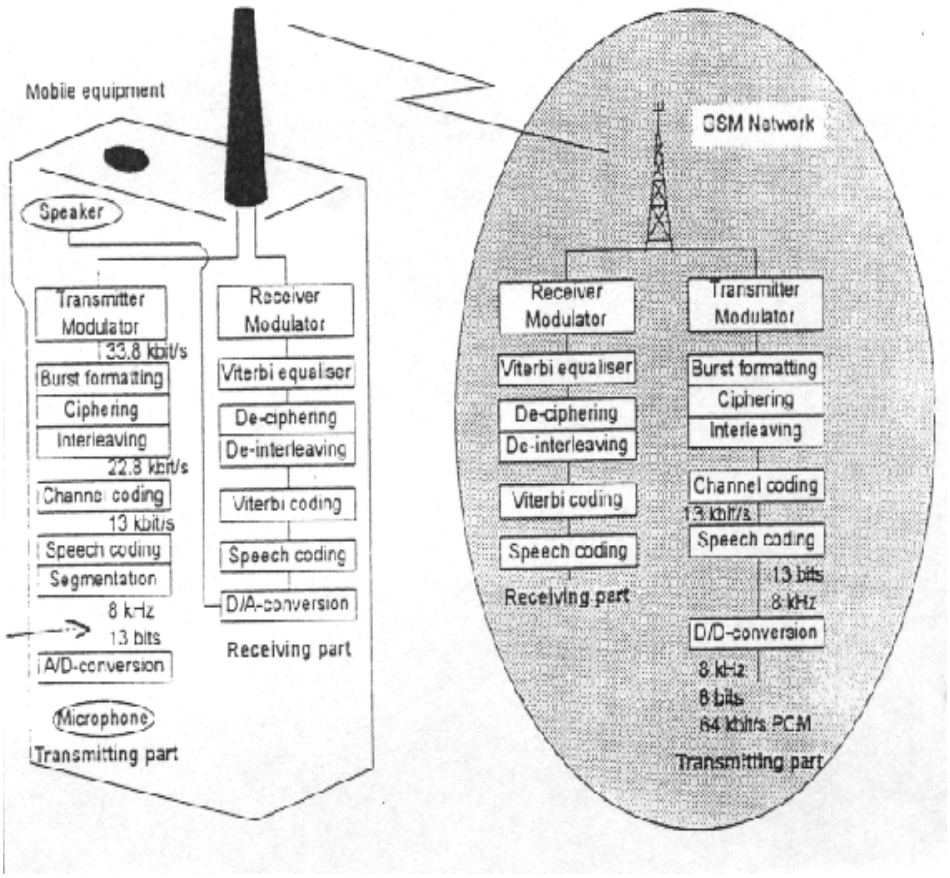
GSM'de hava arayüzünde net bit oranı 270 kbit/s 'dir (Ericsson, 1998a). Dolayısıyla bit zamanı 3,7µs'dir. Bir bit 1,1 km'ye karşılık geldiğinden, eğer mobil istasyonun arkasında 1 km'den bir yansıma varsa yansıyan sinyal direk giden sinyalden 2 km daha uzun yol alır. Bu da istenilen sinyal ile istenilen sinyalden iki bit zamanı geç gelen bir sinyalin karışması anlamına gelir.

3.2.4 Zaman Ayarlaması

TDMA kullanma, mobilin sadece tahsis edilen zaman aralığı süresince sinyal göndermesi, diğer zamanlarda göndermemesi anlamına gelir. Aksi takdirde diğer mobillerden yapılan aramalar, aynı taşıyıcı üzerinde farklı zaman aralıklarında olduğundan karışacaktır. Örneğin mobil, baz istasyonuna çok yakın olsun. Zaman dilimi 3 (TS3 - Time Slot 3) tahsis edilir ve arama için sadece bu zaman dilimi kullanılır. Arama süresince mobil, baz istasyonundan uzaklaşır, böylelikle baz istasyonundan gönderilenler mobile zaman geçtikçe daha geç ulaşmaya başlar, dolayısıyla mobilden çıkan cevap da baz istasyona her zaman geç ulaşır. Eğer bir şey yapılmazsa gecikme ileride daha da artacak ve mobilin TS3 'de gönderdiği mesaj bilgisi ile baz istasyonun TS4' de aldığı mesaj bilgisi üst üste çakışacaktır. Bu ise kesinlikle istenmeyen bir durumdur.

3.3 Transmisyon Problemlerine Çözümler

Şekil 3.8 şematik olarak sinyal işleme bloklarını göstermektedir.



Şekil 3.8 - Sinyal işleme blokları (Harputluoğlu, 2000)

Sinyal işleme, mobil haberleşme sistemindeki en önemli noktalardan biridir. Bu olay mobil cihazda ve şebeke kısmında gerçekleşir.

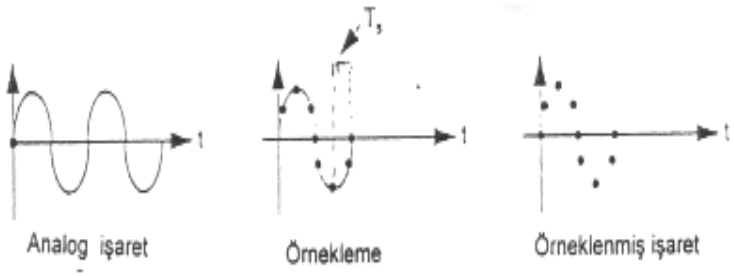
İlk olarak analog konuşma, A/D dönüştürücü (Analog/Sayısal Dönüştürücü) ile sayısal hale getirilir. Daha sonra bit oranının azaltılması için konuşma kodlayıcıya girmek üzere 20 ms'lik parçalara bölünür. Daha sonraki basamak, kanal kodlama ve araya yerleştirme işlemleridir. Konuşmanın şifrelenmesi ve burst formatlama (başlangıç ve bitiş bitlerinin eklenmesi) işlemleri de gerçekleştirildikten sonra son basamak olarak bit dizisinin bir taşıyıcı üzerine modüle edilmesi ve sinyalin gönderilmesi işlemleri gerçekleştirilir. Alıcı tarafta da buna uygun işlemler gerçekleşir. Mobil cihaz tarafı ile şebeke tarafındaki fark, konuşmanın şebeke tarafında A/D veya D/A dönüşüme uğramamasıdır. Eğer konuşma yerine veri gönderilmek isteniyorsa tabii ki mobil tarafında A/D veya D/A dönüşüme gerek kalmayacaktır. Ayrıca verinin konuşma kodlayıcısına da aktarılmasına gerek olmayacaktır. Veri haberleşmesinde transmisyon hataları olma ihtimalleri çok olduğundan kanal kodlama başka bir şekilde yapılacaktır.

3.3.1 Analog Sinyaller ve Sayısal Transmisyon İlkeleri

Sayısal transmisyon temel olarak, birler ve sıfırlardan oluşan simge serilerinin bir noktadan diğerine gönderilmesi anlamına gelir. Konuşma analog, yani sürekli dalga olduğundan dolayı analog işaretin mümkün olduğu kadar sayısal terimlerle ifade edilmesi gerekir. Bu PCM adı verilen bir yöntem kullanılarak yapılır. PCM (Pulse Code Modulation), darbe kod modülasyonudur ve telekomünikasyon sistemlerinde kullanılan genel bir ilkedir. PCM üç aşamada gerçekleştirilir.

1. Örnekleme:

Analog işaretin örneklenmesi işaretin belirli zamanlarda ölçülmesi anlamına gelir. Her değer örnek olarak adlandırılır ve ölçümler tanımlanan belli zaman aralıklarında tekrarlanır (Şekil 3.9). Örnekleme zamanı ise “Ts” dir.



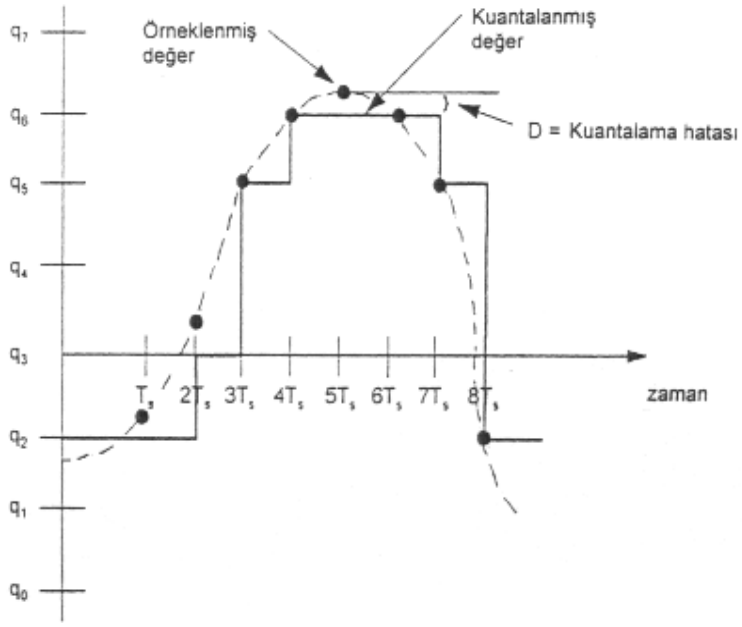
Şekil 3.9 - Analog bir işaretin örneklenmesi

İşaret ne kadar sık aralıklarla örneklenirse, sayısal olarak o kadar iyi modellenir. Örnekleme teoremine göre kayıpsız olarak bir analog işareti tekrar elde edebilmek için, örnekleme frekansının en az olması gereken değer, analog sinyalin en yüksek frekanslı bileşeninin frekansının iki katıdır.

Normal konuşma, 3 KHz'den düşük frekans bileşenlerini içerir. Daha yüksek frekanslı bileşenlerin enerjileri düşük olduğundan konuşma kalitesindeki etkileri ihmal edilebilir. Örnekleme teoremine göre, analog konuşma işaretinin örnekleme frekansı “fs”, en az 2 x 3 KHz = 6 KHz olmalıdır. Telekomünikasyon sistemlerinde örnekleme frekansı 8 KHz’dir.

2. Kuantalama:

Gönderilecek değerlerin sayısını kısıtlamak için genlik seviyesi sonlu değerde seviye gruplarına bölünür. Belirli bir aralıktaki her örnek, seviyelerden biri ile temsil edilir. Şekil 3.10'da analog işaretin GSM sistemde kullanılan “düzgün kuantalama” ilkesi görülmektedir. İki seviye arasındaki mesafeler sabittir. PSTN, seviyeler arası mesafenin değiştiği “A-yöntemi kuantalama” metodunu kullanır. Dolayısıyla farklı genlik seviyelerinin doğruluğu optimize edilir.



Şekil 3.10 - Düzgün kuantalama

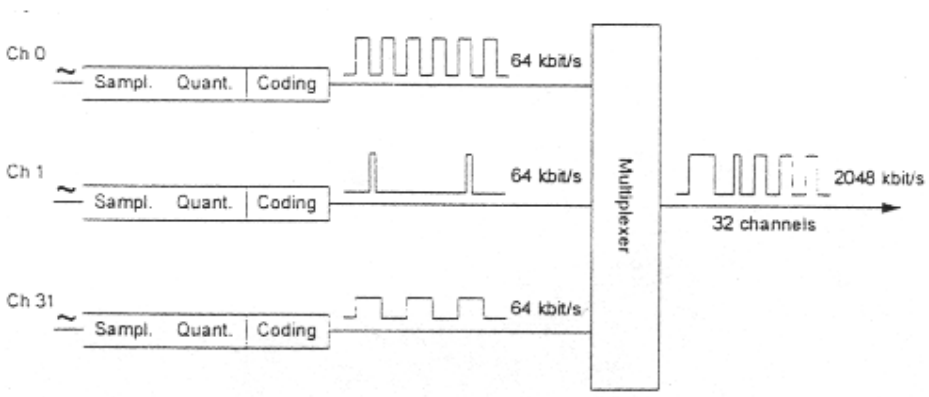
Doğruluğun derecesi kullanılan seviye sayısına bağlıdır. Normal telefonda 256 seviye kullanılırken, GSM'de analog sinyal 8192 seviyede kuantalanır. Gerçekte, sonlu sayıda seviyelerle sürekli analog işareti tam olarak temsil edemeyiz. Şekil 3.10'da işaretlendiği gibi birçok durumda örneklenmiş değerle kuantalanmış değer arasında fark olacaktır. Ayrık seviyelerin sayısı arttırılarak kuantalama hatalarının boyutları azaltılabilir ama hiçbir zaman tam olarak ortadan kaldırılamaz.

3. Kodlama:

Her kuantalanmış değer bir ikili (binary) kod ile temsil edilir. 256 seviye gerçekleştirmek için 8 bit kullanılır (2^8 = 256). GSM' de ise 13 bit 8192 seviyeye tekabül eder (2^13 = 8192).

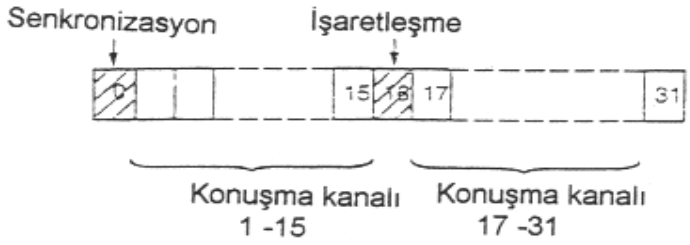
PCM sistemde 8 KHz'de örnekleme yapılıp, kuantalama gerçekleştirilip ve 8 bit ile kodlanırsa 8000 x 8 = 64 kbit/s 'lik bir bit oranı oluşturulmuş olur.

Bu bitlerin gönderilmesi için kullanılan hatta “PCM hat” adı verilir. Hattı daha verimli kullanabilmek için birçok kanal aynı hatta çoklanır. Kullanılan tekniğe Zaman Bölmeli Çoklu Erişim (TDMA) adı verilir ve birçok kanal aynı hattı paylaşır. Her kanal, hattı belirli bir zaman aralığında kullanır (Time slot). Şekil 3.11'de 32 kanalın bir PCM hatta nasıl çoklandığı görülmektedir. Bu hatta bit oranı 32x8x8000=2,048 Mbit/s olur.



Şekil 3.11 - Bir PCM hatta 32 kanalın çoklanması

32 kanal şekil 3.12' de görülen bir çerçeveyi (Frame) oluşturur. 0 no'lu kanal senkronizasyon için, 16 no'lu kanal işaretleşme için ve diğer 30 kanal konuşma veya veri trafiği için kullanılır.



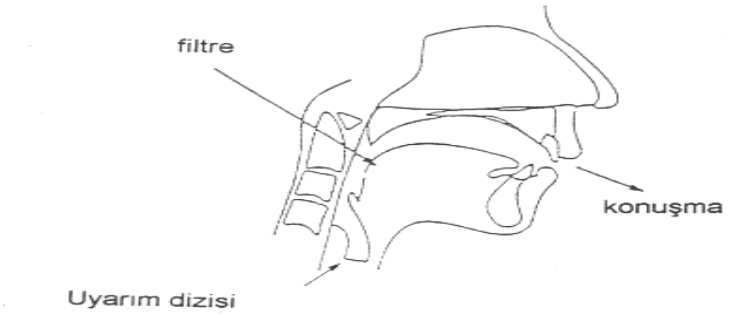
Şekil 3.12 - 32 zaman aralıklı bir çerçeve

3.3.2 Konuşma Kodlama

Sayısal transmisyon kullanımının faydalarından biri de daha fazla aboneye hizmet edebilmektir. Şekil 3.2 'den de görülmüş olduğu gibi bir frekans bandı için bir yerine sekiz kanal mevcuttur. Burada dikkate alınması gereken durum bir frekans bandında aynı zamanda sekiz konuşmanın gerçekleşmesi değil, kullanılan gerçek bandın ne kadar geniş olduğudur. Bazı FDMA hücresel mobil sistemlerde 25 KHz 'lik bir kanal ayarlaması kullanılır (NMT, TACS). GSM'de belirtilen ayrılma ise 200 KHz 'dir. 200 KHz TDMA başına sekiz eş zamanlı kullanıcı, 25 KHz FDMA 'in sekiz kullanıcısı ile aynıdır. Kazanç, analog sistemle karşılaştırıldığında farklı bir frekans planlamasından kaynaklanmaktadır.

PCM olarak kodlanmış konuşma şekline bir göz atarsak görülür ki her konuşma kanalı 64 kbit/s 'dır. Bunun gibi sekiz kanal hava arayüzüne çıktığında 512 kbit/s 'lık bit oranı verir. Bu değer, transmisyon doğruluğu eklenmemiş değerdir. İzin verilen frekans bandı içinde kalmak şartı ile her konuşma kanalı için bit oranını azaltmamız gerekir. Bu ise konuşma kodlama ile sağlanır. Yapılan şey konuşma yerine bilgi gönderilmesidir.

Konuşmanın oluşumunu bildiğimizden konuşmanın tam bir parametrik modelini oluşturabiliriz. Şekil 3.13’e bakıp gerçek yapıdan modele bir geçiş yapabiliriz.



Şekil 3.13 - İnsanda konuşma sistemi

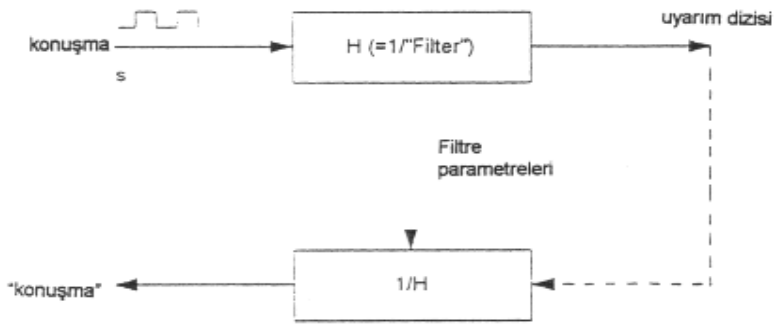
Uyarma, tonu olmayan sessiz harflerin üretildiği konuşma organlarına uygun olan darbeleri içerir.

Filtre, hayali şeyleri gerçekleştiren konuşma organlarının parçalarına uyar. Konuşma organlarının adapte oluşu yavaştır ve hatta denilebilir ki yaklaşık 20ms için sabittirler. Konuşma beyaz gürültünün (white noise) bir dizisi halindedir. Verici tarafta yapmak istediğimiz, konuşmanın ters modelini elde etmektir. Verici tarafta sayısal hale getirilmiş konuşma giriş, uyarma dizileri ise çıkış durumundadır.

Konuşma işareti, konuşmanın oluşturulduğu filtre modeli ile karşılaştırılırsa ters bir karakteristik veren “H” filtresinde filtrelenir. Doğru akortlama ile orijinal uyarma, beyaz gürültü (white noise) olarak tekrar üretilebilir.

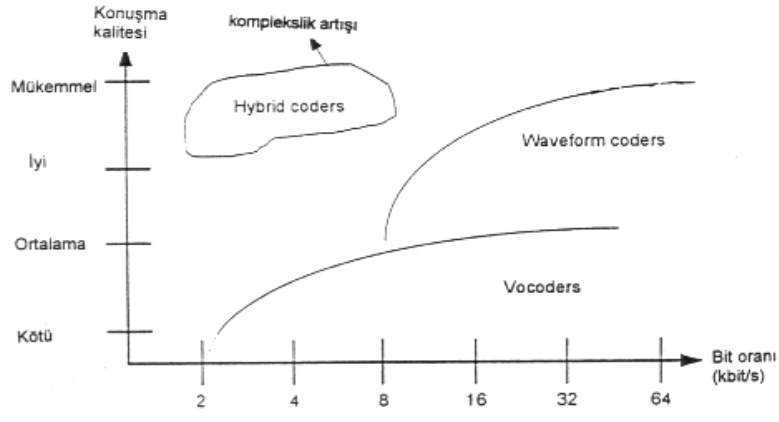
Konuşma kodlayıcısı için bir analiz fonksiyonu, “H” için filtre parametrelerini hesaplar, böylece çıkış işareti beyaz gürültüye mümkün olduğunca yakın olacaktır. Konuşma kodlayıcısı aynı zamanda, ses tellerinin frekansının ve tonlu-tonsuz gürültünün takdiri ile uyarımın olasılığının analizini yapar.

Daha sonra filtre parametreleri hava arayüzü ile gönderilir. Böylece bir “1/H” ters filtre kurulabilir. Aynı zamanda uyarım dizisi hakkındaki bilgi de gönderilir. Bu uyarım dizisi (excitation sequence) “H” filtresinin çıkışıdır (Şekil 3.14).



Şekil 3.14 - Konuşma transmisyon modeli

Üç tip kodlayıcı mevcuttur. Bunlardan biri dalga şekli kodlayıcısıdır. Buna 64 kbit/s PCM kodlayıcı örnek verilebilir. Diğeri vocoder (ses kodlayıcısı) kodlayıcıdır. Konuşma, üretim yönteminin çok basitleştirilmiş bir modelinin temeli üzerinde çalışır. En sonuncusu ise hibrid kodlayıcısıdır. Bunlar çok yüksek ses kalitesi sağlamak için tasarlanmışlardır (Şekil 3.15).



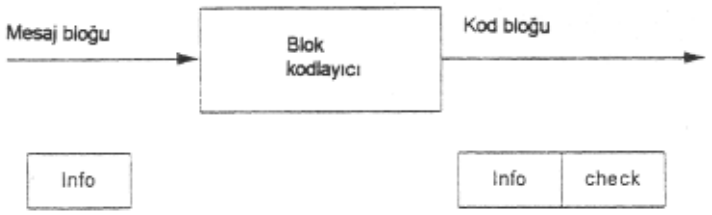
Şekil 3.15 - Konuşma kalitesi bit oranı ilişkisi

3.3.3 Kanal Kodlama

Sayısal transmisyon ile gönderilen işaretin kalitesi genellikle alınan bitlerin ne kadarının doğru olduğunu ifade eden Bit Hata Oranı (BER - Bit Error Rate) ile kısaca ifade edilir (Ericsson, 1998c). BER, toplam bitlerin yüzde kaçının yanlış sezildiğini gösterir. Bu oranın mümkün olduğunca düşük olması istenir.

Kanal kodlama ile alınan bit dizisi içindeki hatalar sezilebilir ve düzeltilebilir. Çünkü bitlerin içinde bazı fazlalık bitler vardır, bilgi bir kaç bitten daha fazla bitlere yayılır.

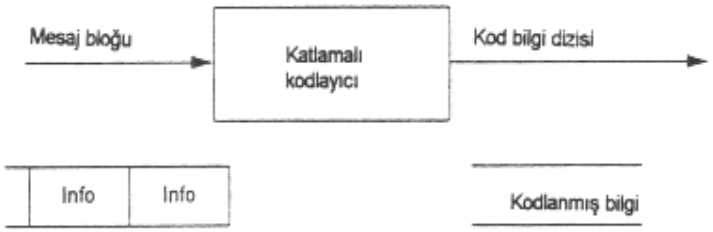
Hata kontrol kodları, blok kodlayıcıları ve katlamalı kodlayıcılar olarak iki kısımda incelenebilir. Blok kodlayıcılarda bazı kontrol bitleri ve eklediğimiz belli sayıda bilgi bitleri vardır. Kontrol bitleri bilgi bitleri ile ilişkilidir. Blok kodlama Şekil 3.16'da gösterilmiştir.



Şekil 3.16 - Blok kodlama

Kod bloğundaki (Code block) kontrol bitleri sadece mesaj bloğundaki (Message block) bilgi bitlerine bağlıdır.

Katlamalı kodlamada, kodlayıcı tarafından üretilen kodlanmış sayısal bloklar sadece kodlayıcıya taşınmış mesaj bloklarına bağlı değildir, ayrıca daha önceden gelen mesaj bloklarındaki bitlere bağlıdır (Şekil 3.17).



Şekil 3.17 - Katlamalı kodlama

Katlamalı kodlayıcıya aktarılan her yeni bit için çıkış iki bit olacaktır. Blok kodlar, bloklar olduğunda sıklıkla kullanılır. Genellikle ARQ (Automatic Repeat Request) yerine getirildiğinde hatayı sezmek için kullanılırlar. ARQ, hata sezilince tekrar iletim isteğidir. Katlamalı kodlama hata düzeltmede, ARQ kolaylığı bulunmadığında, daha çok yardımcı olur (Ericsson, 1998a).

GSM 'de iki metot daha kullanılır. İlk olarak bilgi bitlerinin bir kısmı kontrol bitlerini oluşturacak şekilde blok kodlanır. Sonra bütün bitler katlamalı kodlanır. Kodlama düzeni biraz farklı olmasına karşın hem konuşmaya hem de veriye bu iki basamak uygulanır. Burada çift yönlü kodlama yapılmasının sebebi, eğer düzeltilebiliyorsa hataların düzeltilmesi (katlamalı kodlama) ve eğer bilgi kullanılmayacak kadar çok bozulduysa bunun sezilmesidir (blok kodlama).

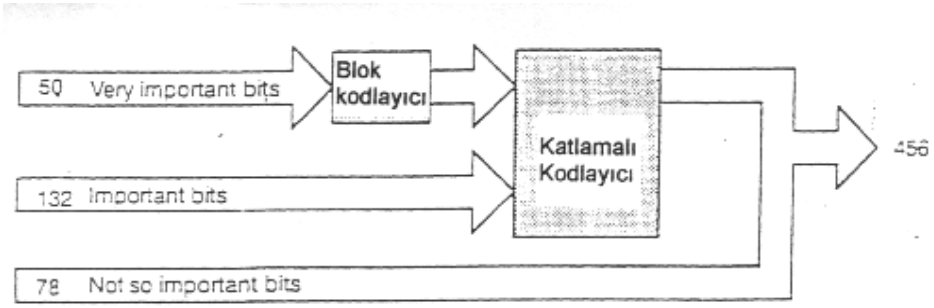
Konuşma 20 ms 'lik parçalara ayrılır. Bu 20 ms 'lik konuşma parçaları sayısal hale getirilir ve konuşma kodlaması yapılır. Her 20 ms 'lik konuşma için 260 kbit/s 'lık bir oran vardır ve konuşma kodlayıcı tarafından sağlanır. Bunlar şöyle ayrılırlar;

50 bit → Çok önemli bitler

132 bit → Önemli bitler

78 bit → Çok önemli olmayan bitler

50 bite üç kontrol biti eklenir (Blok kodlama). Bu 53 bit, 132 önemli bit ve 4 kuyruk biti ile birlikte 378 bite kodlanır. Kalan bitler korunmaz. Bu durum, şekil 3.18 'de gösterilmiştir.

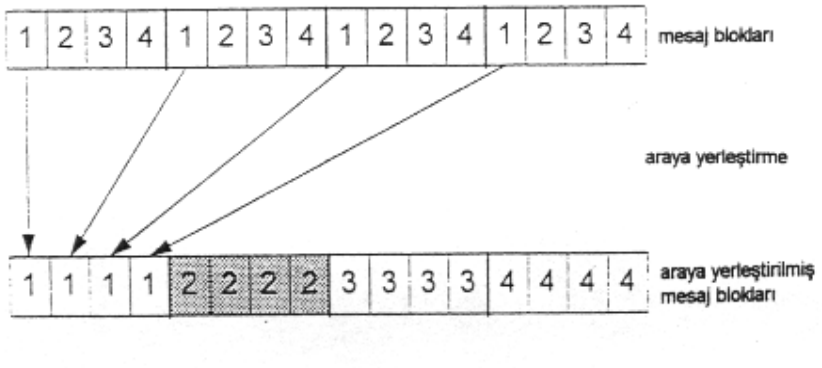


Şekil 3.18 - GSM 'de kanal kodlama

3.3.4 Araya Yerleştirme (Interleaving)

Gerçek hayatta bit hataları burst'lerde ortaya çıkar ve bu da birçok sonuç bitlerinin uzun zayıflama dip noktalarından (fading dips) etkilenmesi ile ilgilidir. Kanal kodlama ancak çok uzun olmayan burst hatalarının ve basit hataların sezilmesinde ve düzeltilmesinde etkilidir. Bu durumda araya yerleştirme işlemi kullanılırsa ilgili mesajın sonuç bitlerinin ayrıştırılması sağlanır. Böylece blok yapıda olmayan bitler birlikte gönderilmiş olur.

Örnek olarak, mesaj bloğu dört bit içeriyor. Dört bloğunun ilk bitlerini çerçeve (frame) diye adlandırılan dört bitlik yeni bloklara koyalım. 2 'den 4 'e kadar olan bitlere de aynı işlemi uygulayalım. Daha sonra çerçeveleri, 1 - numaralı bitler, 2 - numaralı bitler v.b. şekilde gönderelim (Şekil 3.19).



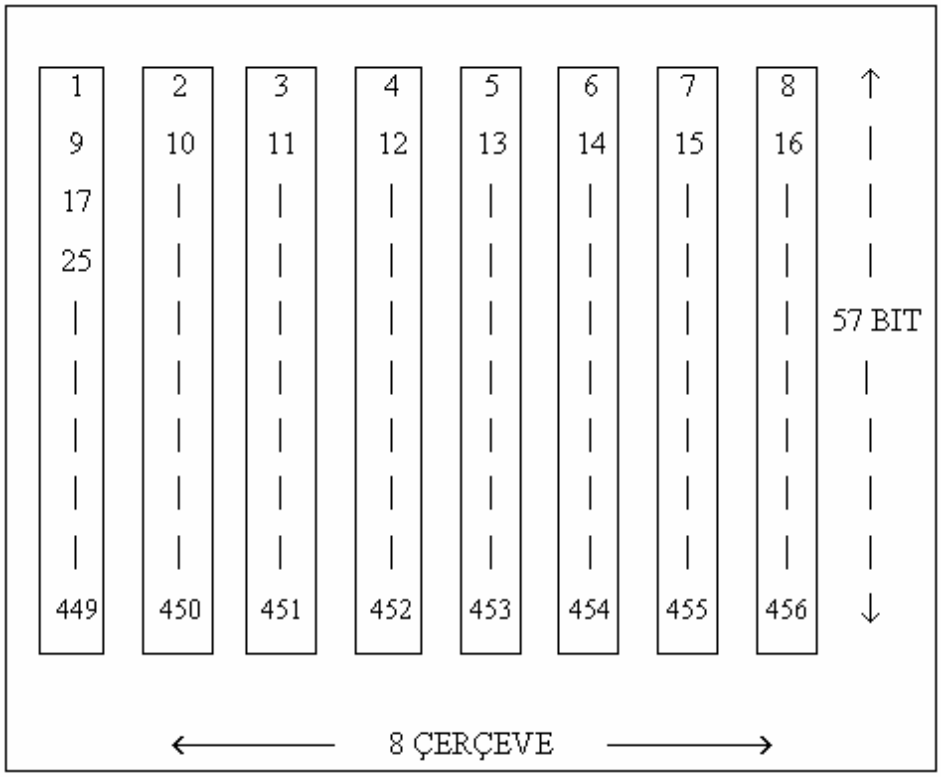
Şekil 3.19 - Araya yerleştirme

İletim sırasında, 2 no'lu çerçevede hata oluşursa araya yerleştirme (interleaving) işlemi yapıldığından, bütün mesaj bloğu kaybolmaz. Sadece her mesaj bloğunun ikinci bitinde hata oluşur (Şekil 3.20).

1	X	3	4	1	X	3	4	1	X	3	4	1	X	3	4

Şekil 3.20 - Alınmış, tekrar elde edilmiş mesaj blokları

Kanal kodlama ile bütün bloklardaki bilgi tekrar elde edilebilir. GSM 'de kanal kodlayıcı her 20 ms konuşma için 456 bit tedarik eder. Bunlar her biri 57 bitten oluşmuş sekiz blok halinde araya yerleştirme işlemine tabi tutulur (Şekil 3.21).



Şekil 3.21 - Kodlanmış konuşmanın 20 ms araya yerleştirilmesi

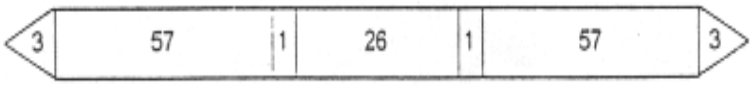
3.3.5 İkinci Seviye Araya Yerleştirme

57 bitlik sekiz grup halinde araya yerleştirilmiş 20 ms 'lik konuşma, 456 bitlik konuşma çerçevesini oluşturur (Şekil 3.22).

A	B	C	D
20 ms konuşma 456 bits = 8 x 57	20 ms konuşma 456 bits = 8 x 57	20 ms konuşma 456 bits = 8 x 57	20 ms konuşma 456 bits = 8 x 57

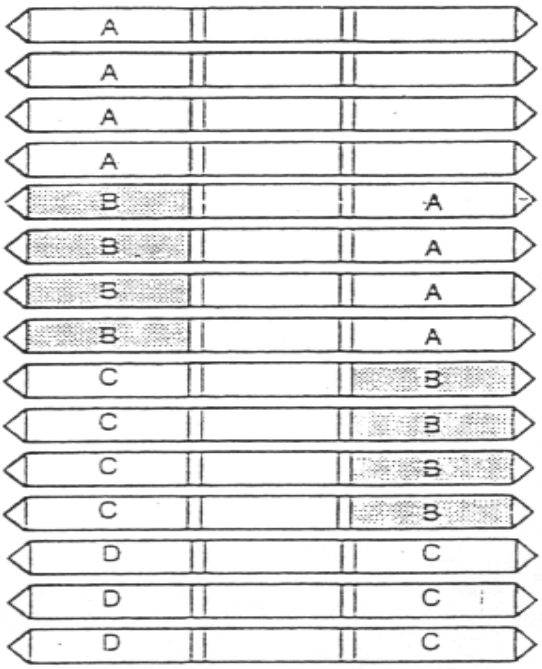
Şekil 3.22 - Konuşma çerçevesi

Eğer aynı konuşma çerçevesinden 2x57 bit alırsak ve bunları aynı burst'e sokarsak, bu durumda burst kaybı %25 bit kaybına sebep olur ki bu da kanal kodlama için çok fazladır. Dolayısıyla araya yerleştirmenin başka bir seviyesini iki konuşma çerçevesi arasına eklememiz gerekir (Şekil 3.23).



Şekil 3.23 - Normal burst

Bu durum, sistemde küçük bir geciktirme oluşturur. Kayıp her konuşma çerçevesinin bitlerinin sadece %12,5 kadarını etkilediğine göre, bir bütün burst kaybı göze alınabilir ve kanal kodlama ile düzeltilebilir (Şekil 3.24).



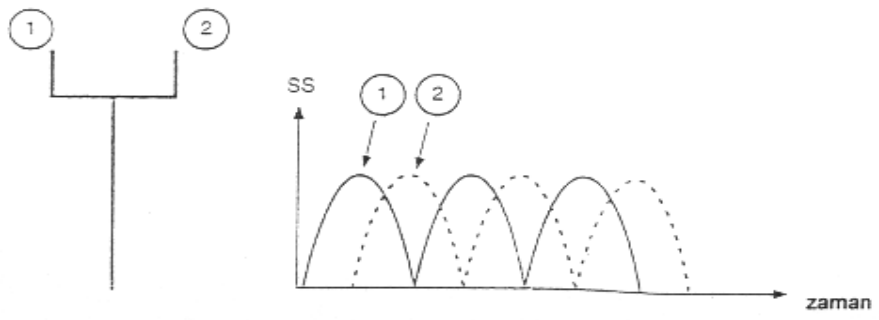
Şekil 3.24 - İkinci seviye araya yerleştirme

3.3.6 Modülasyon

GSM'de kullanılan modülasyon metodu GMSK'dir (Ericsson, 1998a). GMSK bir sayısal modülasyon şeklidir ve gönderilecek bilgi sayısaldır. Bu bilgi bir veri veya sayısal hale getirilmiş konuşma olabilir. Modülatör, bir faz modülatörü olarak düşünülebilir. Taşıyıcı, modülatöre gönderilen bilgi bitlerine bağlı olarak fazı değiştirir. GMSK, bir burst içinde arzu edilen sabit zarf modülasyonunu içerir. Fazı değiştirirken düzgün eğri şekilleri elde etmek için, temel bant işareti bir Gauss geçiş bandı ile filtrelenir. GMSK, alışılmış MSK ile karşılaştırıldığında daha dar bir bant genişliği elde edilir.

3.3.7 Anten (Uzay) Farklılığı

Farklılığı sona erdirmenin bir yolu zayıflamadan bağımsız etkilenen iki kabul kanalı kullanmaktır. İkisinin de aynı anda çok derin bir zayıflama dip noktasından etkilenme riski çok küçüktür. Bu da iki Rx anteninin aynı işareti bağımsız olarak almasının faydalı olacağı demektir, böylece işaret zayıflama zarflarından az etkilenecektir. İki işaretin en iyisini seçmekle zayıflama derecesi azaltılmış olur. Antenler arasındaki mesafe iki antendeki işaret ilişkisi (correlation) ile ilgilidir. İlişki, işaretlerin neye benzediğini gösteren istatistiksel bir terimdir. Pratik olarak bir kaç metredir. 900 MHz'de, antenler arasında 5–6 metre mesafe ile 3 dB kazanç sağlamak mümkündür. 1800 MHz'de ise dalga boyunun düşmesiyle mesafe de kısalacak, böylece daha az mesafe ile aynı kazanç elde edilebilecektir (Ericsson, 1998b). Şekil 3.25 ’te iki ayrı antene ait farklılık görülmektedir.



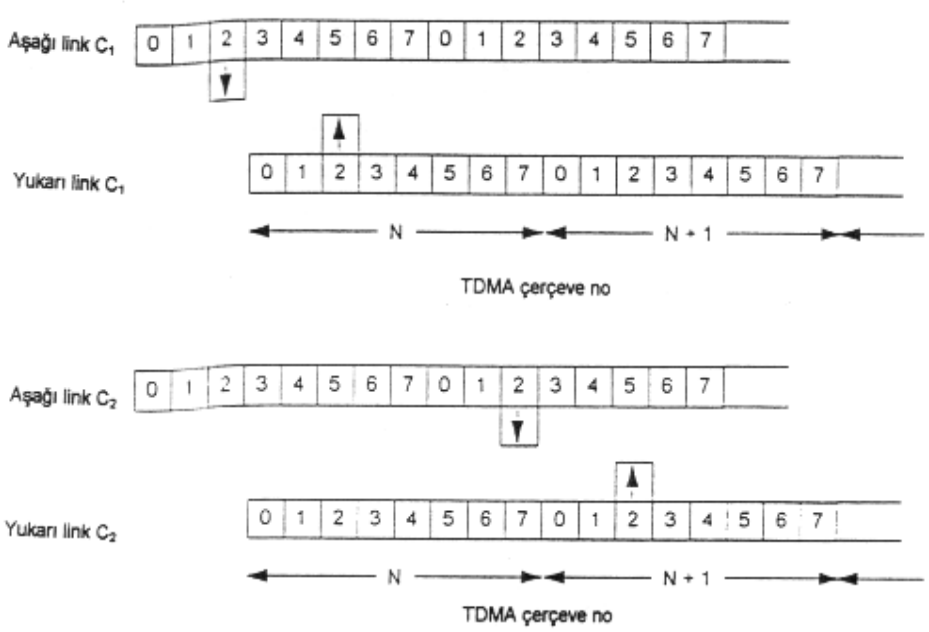
Şekil 3.25 - Anten farklılığı

3.3.8 Frekans Atlaması (Frequency Hopping)

Daha önce Rayleigh zayıflamasında belirtildiği gibi, zayıflama deseni frekansa bağımlıdır. Bu da farklı frekanslar için farklı yerlerde zayıflama dip noktalarının oluşması anlamına gelir.

Kazanç sağlamak için, arama süresince belli sayıda frekansın arasında taşıyıcı frekans değiştirilir ve bunlardan birinde zayıflama dip noktası mevcut ise bilginin sadece küçük bir bölümü kaybedilir.

Kompleks işaret işleme ile işaret tekrar onarılabilir ve eski haline getirilebilir. Aynı arama için, N no'lu TDMA çerçevesi süresince C_1 frekansı, N+1 no'lu TDMA çerçevesi süresince C_2 frekansı kullanılır ve bu olay değişim halinde devam eder (Şekil 3.26).



Şekil 3.26 - C_1 ve C_2 frekansları arasında frekans atlaması

3.3.9 Dengeleyici (Equalizer)

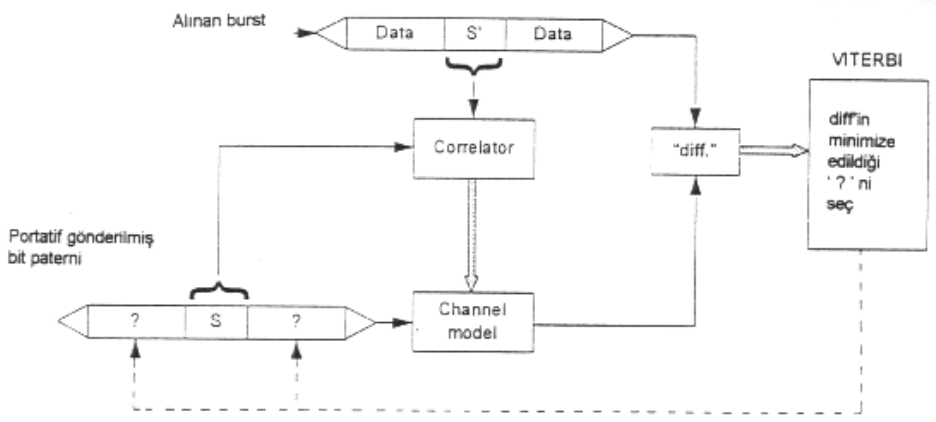
Daha önce zaman ayarlamasında bahsedildiği gibi Rx anteninden çok uzaktaki yansımalar ISI'yı (Ara Simge Girişimi) oluştururlar. İşaretler yayılırlar ve komşu simgeler birbirleri ile girişirler. Bu da alıcıda, hangi bilginin gönderildiğine karar vermeye çalışmada problemler oluşturur.

Kanal; kablo, optik kablo veya bir mikrodalga hat olabilir. Her çeşit kanal; kendi bant genişliği, zayıflaması ve buna benzer özelliklere sahiptir. Optimum bir alıcı, transmisyon için belirli bir kanala uydurulur.

Bu durumda hava arayüzünün matematiksel bir modeli oluşturulmak ve alıcıyı da bu modele ayarlamak istenir. Eğer yansımaların ne kadar uzun ve kuvvetli olduğunu biliyorsak, bunu alınan burst sezildiğinde hesaba katabiliriz. Bu da transmisyon kanalının bir modelini oluşturan dengeleyici ile yapılır ve buna ek olarak en muhtemel gönderilen dizi hesaplanır (Şekil 3.27).

Veri zaman aralıkları içinde yer alan burst'lerde gönderilir. Bilinen bir desenin burst dizisinin ortasında oto korelasyon ile özellikler yerleştirilir. Bu model bütün zamanları değiştirir fakat bir bit süresince sabit olarak dikkate alınır.

GSM şartnamesi, dört bite kadar (yaklaşık 15µs 'ye tekabül eder) gecikmiş, yansıyan bir işareti ele alabilen veya direkt ve yansımış işaret arasındaki yol farkının yaklaşık 4,5 km olduğu bir işareti ele alabilen bir dengeleyicinin sistemde var olmasını önerir (Ericsson, 1998a).



Şekil 3.27 - Viterbi dengeleyici

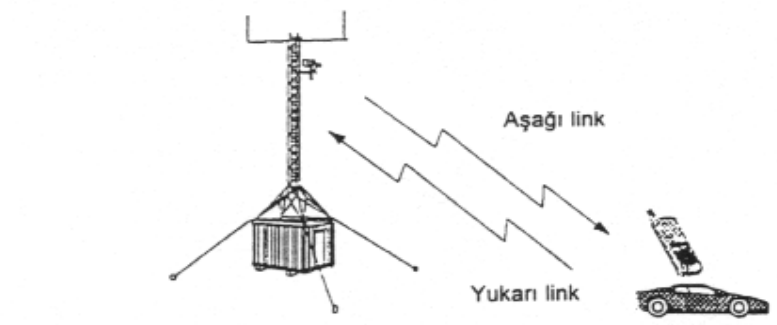
3.4 Sayısal Transmisyon Probleminin Özet Çözümü

İletimdeki sorunların çözümünde ilk olarak konuşma sayısal hale dönüştürülür ve 20 ms 'lik parçalara bölünür. Ondan sonra bit oranını azaltmak için konuşma kodlama işlemi ve hata kontrol için kodlama işlemleri yapılır. Araya yerleştirme işlemi küçük bir gecikmeye sebep olur. Şifreleme 1:1 ilişkisi (giriş-çıkış) ile yapılır ve bitler, her 20 ms'lik konuşma için yarım burst'ler halinde şekillenir. Bunlar daha sonra yaklaşık 270 kbit/s 'lık bir hızla uygun zaman aralıklarında gönderilir.

Alıcıda ise burst'ler alınır ve bit dizilerinin hesaplandığı dengeleyicide bir kanal modeli oluşturulur. Hepsinden sonra sekiz yarım burst alınır ve deşifre edilir. Bunlar 456 bit mesaj olarak kurulur. Bu dizi üzerinde, iletim süresince hata düzeltme ve sezme işlemi için kod çözme işlemi gerçeklenir. Kod çözücü, hata düzeltme işleminin daha iyi olması için dengeleyiciden "soft information" kullanır. "Soft information", bitin doğru olduğuna dair bir olasılıktır. Son olarak, bit dizisi konuşma kodlama işlemine tabi tutulur ve analog konuşmaya dönüştürülür.

4 SAYISAL RADYO (HAVA) ARAYÜZÜ

Radyo arayüzü MS ve BTS arasındaki bağlantının genel adıdır. Her taşıyıcı frekans için bir TDMA çerçeve kullanılır. Her çerçeve sekiz TS (zaman aralığı) içerir. BTS'den MS'e olan yön aşağı link (down link), MS'den BTS'e olan yön de yukarı link (up link) olarak tanımlanır (Şekil 4.1). Bir hücrede, kullanılmak istenen taşıyıcı frekans sayısı kadar alıcı-verici birim (TRU – Transceiver Unit) bulunmalıdır. Her TRU sadece bir frekansta yayın yapabilir.



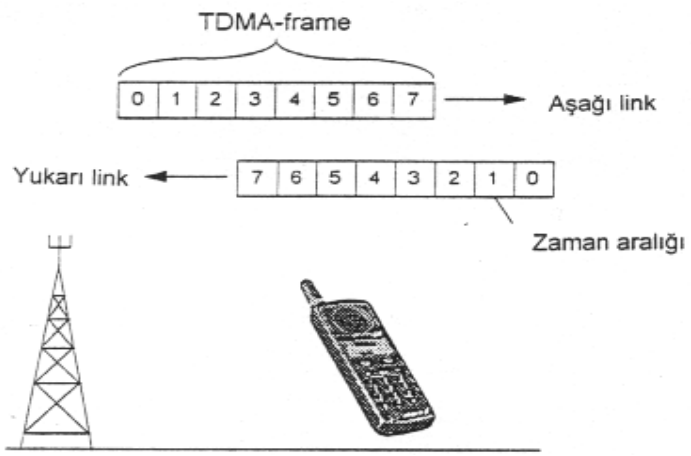
Şekil 4.1 - Bir radyo kanalı üzerindeki yukarı ve aşağı link

4.1 Kanal Kavramı

GSM sistemi, 900 MHz bandında 1'den 124'e kadar numaralandırılmış taşıyıcılar içerir. GSM, her taşıyıcının sekiz zaman aralığına bölündüğü Zaman Bölmeli Çoklu Erişim metodu (TDMA – Time Division Multiple Access) kullanır (Ericsson, 1998d). Mobil cihaz aynı zaman aralığında gönderir ve alır. Bu demektir ki, aynı zamandaki sekiz konuşma, aynı radyo kanalında yer alabilir.

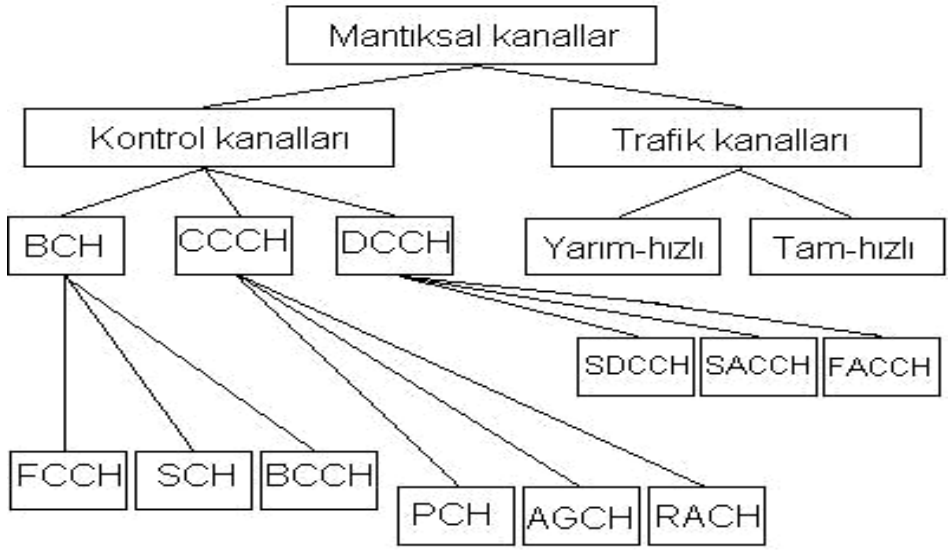
Taşıyıcı üzerindeki TDMA çerçevedeki bir zaman aralığı (TS), bir fiziksel kanalı ifade eder (Ericsson, 1998a). Eğer, her kullanıcının belli sayıda frekanstan biri yoluyla bir sisteme bağlandığı FDMA sistemi ile karşılaştırma yaparsak, netice olarak görülür ki, GSM'de her taşıyıcı için

sekiz fiziksel kanal mevcuttur (Şekil 4.2). Bir TS süresince gönderilen bilgiye “burst” adı verilir.



Şekil 4.2 - TDMA kanal kavramı

Birçok bilgi türü, BTS ve MS arasında gönderilir. Farklı bilgi türleri, fiziksel kanallar üzerinde farklı bir düzen ve dizide gönderilirler. Gönderilen bilginin türüne bağlı olarak, bu düzen ve diziler mantıksal kanalları ifade eder. Mantıksal kanallar fiziksel kanallar üzerinde bilgi türlerine göre planlanmıştır (Ericsson, 1998a). Örneğin konuşma, “trafik kanalı” olarak adlandırılan mantıksal kanaldan gönderilir. Mantıksal kanallar “kontrol” ve “trafik” kanalları olmak üzere ikiye ayrılır. Şekil 4.3 ’te mantıksal kanalların alt bölümleri görülmektedir.



Şekil 4.3 - Mantıksal kanallar

4.1.1 Kontrol Kanalları

İlk olarak, MS açıldığında bir radyo baz istasyonunu sezmeye çalışır. Bu, tüm frekans bandını tarayarak veya kullanılan operatör için ayrılmış Yayın Kontrol Kanalı (BCCH - Broadcast Control Channel) taşıyıcısını içeren bir prosedür kullanarak yapılır. MS, en kuvvetli taşıyıcıyı bulduğunda bunun BCCH taşıyıcısı olup olmadığını tespit etmelidir. Bir BCCH taşıyıcısı, kontrol kanallarının taşınmasında kullanılan bir frekanstır (Ericsson, 1998a).

4.1.1.1 Yayın Kanalları (BCH)

- Frekans Düzeltme Kanalı (FCCH - Frequency Correction Channel):

FCCH'de bir sinüs işareti gönderilir. Amaçlarından biri BCCH taşıyıcısına emin olmak, diğeri de MS'in frekansa senkron olmasını sağlamaktır. FCCH, aşağı linkten ve tek noktadan çok noktaya gönderilir.

- Senkronizasyon Kanalı (SCH - Synchronisation Channel):

MS için bundan sonra ki olay, belirli bir hücre içindeki yapıya senkronize olmak ve seçilen baz istasyonunun bir GSM baz istasyonu olduğuna emin olmaktır. Senkronizasyon kanalını dinleyerek; MS, seçilen baz istasyonun bu hücresindeki TDMA çerçeve yapısındaki bilgiyi alır. Bu bilgi TDMA çerçeve no'sudur. Ayrıca Temel İstasyon Kimlik Kodu da alınır (BSIC). BSIC, baz istasyonu bir GSM şebekeye aitse kod çözme işlemine tabi tutulur. SCH aşağı linkten ve tek noktadan çok noktaya gönderilir.

- Yayın Kontrol Kanalı (BCCH - Broadcast Control Channel):

MS'in, dolaşımı başlatmak yani gelen aramaları beklemek veya istendiğinde arama yapmak için alması gereken son bilgi, hücre ile ilgili bazı genel enformasyonlardır. Bu, üzerinde MS'in ölçümleri dikkate alacağı BCCH içinde gönderilir. Bu ölçümler, hücrede izin verilen maksimum çıkış gücü ve komşu hücreler için BCCH taşıyıcısı ölçümleridir. BCCH, aşağı linkten ve tek noktadan çok noktaya gönderilir. Artık MS bir baz istasyonuna kilitlenmiştir ve hücredeki çerçeve yapısına senkronizedir. Baz istasyonları birbirine senkron değildir, yani MS hücre değiştirdiğinde, her defasında FCCH, SCH ve BCCH'in baştan okunması gereklidir.

4.1.1.2 Ortak Kontrol Kanalları (CCCH)

- Çağırma Kanalı (PCH - Paging Channel):

Belirli zaman aralıkları içinde MS, şebekenin kendisi ile bağlantı kurmak isteyip istemediğini görmek için PCH'yi dinler. Sebep, gelen bir arama veya gelen bir kısa mesaj olabilir. PCH'de gelen bilgi, MS'in kimlik numarası (IMSI) veya geçici kimlik numarasıdır (TMSI). PCH, aşağı linkten ve tek noktadan tek noktaya gönderilir.

- Rastgele Erişimli Kanal (RACH - Random Access Channel):

MS, numaraları aldıktan sonra, RACH'dan numaraları aldığını bildirir. MS, ayrıca bu kanalı, şebekeye bağlanmak istendiğinde de kullanır. RACH, yukarı linkten ve tek noktadan tek noktaya gönderilir.

- Erişim Verme Kanalı (AGCH - Access Grant Channel):

Şebeke bir işaretleşme kanalı belirler (SDCCH). Bu işaretleşme kanalının belirlenmesi olayı AGCH'da gerçekleşir. AGCH, aşağı linkten ve tek noktadan tek noktaya gönderilir.

4.1.1.3 Tahsis Edilmiş Kontrol Kanalları

- Tek Başına Tahsis Edilmiş Kontrol Kanalı (SDCCH - Stand Alone Dedicated Control Channel):

MS, belirtilen işaretleşme kanalı olan SDCCH'e anahtarlama yapar. Arama kurulumu prosedürü bu kanal üzerinden yapılır. SDCCH, hem aşağı hem de yukarı linkten ve tek noktadan tek noktaya gönderilir. Arama kurulumu gerçekleştirildiğinde MS, kendine atanan taşıyıcı üzerindeki zaman aralığında (TS) tanımlı trafik kanalına (TCH) anahtarlanır.

- Yavaş Birleştirilmiş Kontrol Kanalı (SACCH - Slow Associated Control Channel):

Yukarı linkte MS, kendi baz istasyonu ve komşu baz istasyonlarıyla ilgili ortalama ölçümleri SACCH üzerinden gönderir. Bu ölçümlerden kendi baz istasyonu ile ilgili ölçüm sinyal gücü ve kalitesi, komşu baz istasyonlarıyla ilgili ölçüm sinyal gücü üzerinedir. SACCH, tek noktadan tek noktaya gönderilir.

- Hızlı Birleştirilmiş Kontrol Kanalı (FACCH - Fast Associated Control Channel):

Eğer konuşma anında, aniden hücreler arası aktarma (handover) gerekirse FACCH kullanılır.

4.1.2 Trafik Kanalları (TCH)

Trafik kanalları tam-hızlı (full-rate) ve yarı-hızlı (half-rate) olmak üzere ikiye ayrılırlar. Bir tam-hızlı TCH, bir TS yani bir fiziksel kanalı işgal eder. İki yarı-hızlı TCH ise bir fiziksel kanalı paylaşırlar böylece hücrenin kapasitesi iki katına çıkmış olur, ancak konuşma kalitesi azalır.

5 HÜCRE PLANLAMA

Hücre Planlama kapsamında yapılan işler şunlardır:

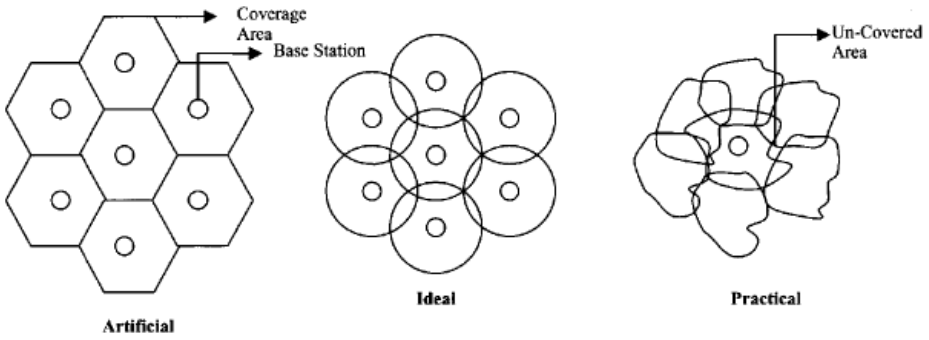
- Radyo ekipmanları için istasyon belirlemek
- Radyo ekipmanlarını belirlemek
- Radyo ekipmanlarını ayarlamak

Her hücresel ağın yeterli kapsama alanı ve konuşma kalitesi sağlayabilmesi için hücre planlamaya ihtiyacı vardır.

5.1 Hücreler

Bir hücre, bir baz istasyonu anten sisteminin kapsama alanı olarak tanımlanabilir. Bazı özel durumlarda bir hücre birkaç anten sisteminin kapsama alanları birleşiminden oluşabilir. Hücre, bir mobil ağın en küçük yapı taşıdır. Mobil ağların, hücresel ağlar (cellular networks) olarak da anılmasının sebebi budur.

Hücre şekli teoride altıgen olarak gösterilir ancak bu yapay bir gösterimdir. Baz istasyonu anteni tarafından yayılan sinyalin ideal kapsaması ise dairesel olarak gösterilir; ancak gerçekte bazı alanlar çeşitli nedenlerle gerekli sinyal seviyesine sahip olamazlar (Mishra, 2004). Bu sebeple hücreler pratikte geometrik olmayan şekillere sahiptirler (Şekil 5.1).



Şekil 5.1 - Hücre şekilleri (Mishra, 2004)

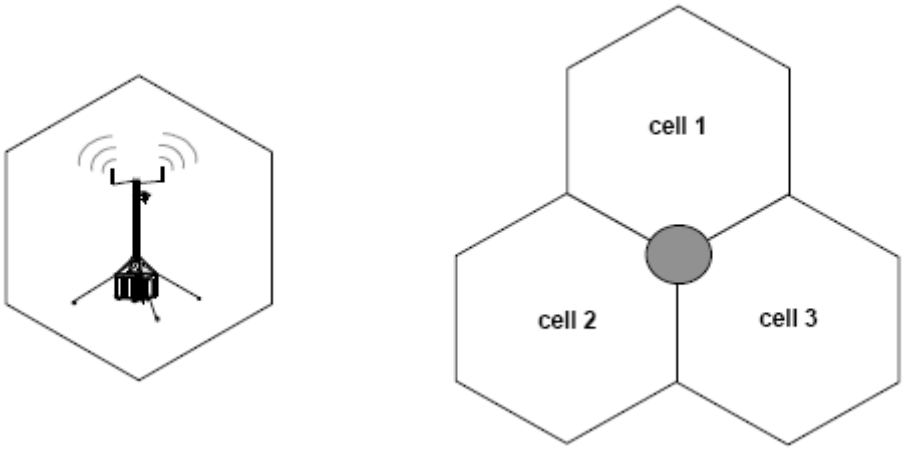
Hücre temel tipleri ikiye ayrılır (Şekil 5.2). Bunlar:

- Dairesel hücre (Omni directional cell):

Bir dairesel hücre, her yöne (360 derece) eşit miktarda iletim yapan antene sahip bir baz istasyonu tarafından sağlanır. Genellikle bu hücreler kapsama amaçlı kullanılır.

- Sektörel hücre (Sector cell):

Bir sektörel hücre, sadece belirtilen yönde iletim yapan bir anten tarafından kapsanan alandan oluşur. Örnek olarak bir baz istasyonu 120 derecelik antenler kullanan üç hücre içerebilir. Genellikle bu tip hücreler kapasite amaçlı kullanılır.



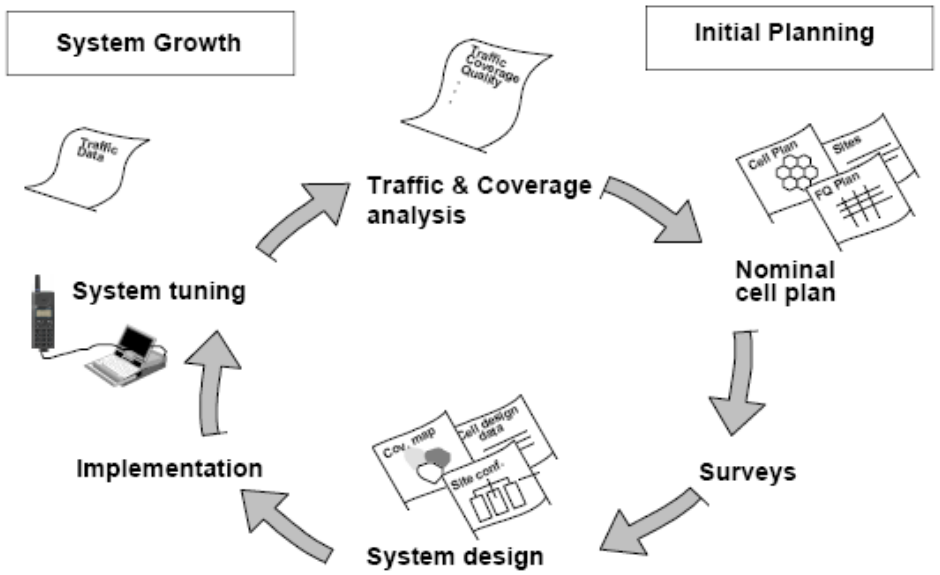
Şekil 5.2 - Dairesel ve sektörel hücreler (Ericsson, 1998c)

5.2 Hücre Planlama Adımları

Başlıca hücre planlama adımları sırasıyla şunlardır:

- Trafik ve kapsama analizi
- Nominal hücre planı
- Saha incelemeleri
- Sistem tasarımı
- Sistem gerçekleştirimi
- Sistemin düzenlenmesi

Gerçekleştirilen sistemin trafik artışı sebebiyle büyümesi ya da yeni binalar yapılması sonucu değişmesi durumunda hücre planlama süreci, yeni bir trafik ve kapsama analizi yapılarak en baştan ele alınmalıdır (Şekil 5.3).



Şekil 5.3 - Hücre planlama adımları (Ericsson, 1998d)

5.2.1 Trafik ve Kapsama Analizi

Hücre planlama süreci, trafik ve kapsama analizi ile başlar. Bu analiz sonucu bölgenin coğrafi yapısı ve beklenen kapasite ihtiyacı hakkında bilgi sahibi olunur. Araştırılan veri tipleri şunlardır:

- Maliyet
- Kapasite
- Kapsama
- Servis derecesi (GoS - Grade of Service)
- Kullanılabilen frekanslar
- Bit hata oranı (BER)
- Sistemin büyümeye yatkınlığı

Trafik talebi, hücresel ağ mühendisliğinin temelini oluşturur. Trafik talebi, sisteme giriş yapan abone sayısını ve yaratacakları trafik miktarını içeren bir kavramdır. Trafik talebinin coğrafi dağılışı, aşağıdaki demografik veriler kullanılarak hesaplanabilir:

- Nüfus dağılımı
- Araba kullanım dağılımı
- Gelir dağılımı
- Arazi kullanım verileri
- Telefon kullanım istatistiği
- Abonelik ücreti, konuşma ücretleri ve cep telefonu fiyatları gibi diğer faktörler

5.2.2 Nominal Hücre Planı

Trafik ve kapsama analizinden gelen verilerin derlenmesinden sonra bir nominal hücre planı oluşturulur. Nominal hücre planı şebekenin grafiksel bir gösterimidir. Nominal hücre planları ilk hücre planlarıdır ve ileriki planlama için temel oluşturur.

Bu adımda, kapsama ve girişim tahmin edilmesine başlanır. Bu planlamada radyo yayılımının incelenmesi için bilgisayar destekli analiz araçlarına ihtiyaç vardır (Ericsson, 1998b).

5.2.3 Saha İncelemeleri

Nominal hücre planı oluşturulup kapsama ve girişim tahminleri kabaca doğrulandıktan sonra istasyonlarda radyo ölçümleri gerçekleştirilir. Bu kritik bir adımdır çünkü hücresel bir ağ planlaması yapılırken istasyon yerlerinin uygun olup olmadığının belirlenmesinde sahanın yani gerçek ortamın değerlendirilmesi gereklidir. Böylece cep telefonlarının bulunacağı gerçek ortamdan alınan sinyal seviyesi ölçümleri ile daha iyi tahminler yapılabilir.

5.2.4 Sistem Tasarımı

Planlama aracının ürettiği tahminlerin güvenilir olduğu anlaşıldıktan sonra baz istasyonu ekipmanı, BSC ve MSC boyutlandırması yapılır. Böylece son hücre planı oluşturulmuş olur. Bu plan daha sonra sistem kurulumunda kullanılır. Bunlara ek olarak her hücre için tüm hücresel parametreleri içeren ve Hücre Tasarım Verisi (CDD - Cell Design Data) adı verilen bir doküman oluşturulur (Ericsson, 1998d).

5.2.5 Sistem Gerçekleştirimi

Son hücre planı oluşturulması ve sistem tasarımından sonra sistem gerçekleştirimi, görev ve yetki dağılımı, test etme işlemleri gerçekleştirilir.

5.2.6 Sistemin Düzenlenmesi

Sistem kurulumu tamamlandıktan sonra sistemin talepleri karşılayıp karşılamadığı devamlı olarak değerlendirilir. Bu işleme sistemin düzenlenmesi (system tuning) adı verilir ve şu adımları içerir:

- Son hücre planının başarılı bir şekilde gerçekleştirildiği kontrol edilir
- Müşteri şikayetleri değerlendirilir
- Şebeke performansının kabul edilebilir olduğu kontrol edilir
- İhtiyaç durumunda gerekli parametreler değiştirilir ve ilgili ölçümler yapılır

Sürekli artan abone sayısı ve trafik miktarından dolayı sistemin devamlı olarak düzenlenmesi gerekir. Sonunda sistem artan yük ve trafiği yönetemeyecek bir noktaya ulaşır. Bu noktada kapsama analizi gerçekleştirilerek hücre planlama adımlarına en baştan başlanır ve bu döngü bu şekilde devam eder (Ericsson, 1998b).

5.3 Trafik Kavramı

Hücresel sistem kapasitesi şu faktörlere bağlıdır:

- Ses ve veri için ayrılan kanal sayısı
- Aboneler için belirlenen servis derecesi (GoS)

Trafik kavramı ile bir hücrede olması gereken kanal sayısı tahmin edilir. Bu tahmin, seçilen sisteme ve abonelerin gerçek ya da kabullenilen davranışlarına dayanır (Ericsson, 1998b).

Trafik, kanalların kullanımı ve tutulma zamanı ile ilgilidir ve Erlang (E) birimi ile ölçülür. Eğer tek bir abone bir saat boyunca konuşmasını sürdürürse 1 E'lık trafik yaratmış olur (Mishra, 2004).

Trafik, şu formül ile hesaplanır:

$$A = \frac{n \times T}{3600}$$

Burada birimi Erlang olan A, sistemdeki bir ya da daha fazla kullanıcının oluşturduğu trafiktir. n, saatte yapılan arama sayısıdır. T ise saniye cinsinden ortalama konuşma süresidir (Ericsson, 1998a).

Bir hücrenin taşıyabileceği trafik, sahip olduğu trafik kanalları (TCH) ile müşteri ve operatör tarafından kabul edilebilir tıkanıklık (congestion) miktarına bağlıdır. Bu tıkanıklık miktarına servis derecesi (GoS - Grade of Service) adı verilir (Ericsson, 1998b).

Kabullenilen abone davranışları için Erlang B tablosu kullanılır (Şekil 5.4).

n	.007	.008	.009	.01	.02	.03	.05	.1	.2	.4	n
1	.00705	.00806	.00908	.01010	.02041	.03093	.05263	.11111	.25000	.66667	1
2	.12600	.13532	.14416	.15259	.22347	.28155	.38132	.59543	1.0000	2.0000	2
3	.39664	.41757	.43711	.45549	.60221	.71513	.89940	1.2708	1.9299	3.4798	3
4	.77729	.81029	.84085	.86942	1.0923	1.2589	1.5246	2.0454	2.9452	5.0210	4
5	1.2362	1.2810	1.3223	1.3608	1.6571	1.8752	2.2185	2.8811	4.0104	6.5955	5
6	1.7531	1.8093	1.8610	1.9090	2.2759	2.5431	2.9603	3.7584	5.1086	8.1907	6
7	2.3149	2.3820	2.4437	2.5009	2.9354	3.2497	3.7378	4.6662	6.2302	9.7998	7
8	2.9125	2.9902	3.0615	3.1276	3.6271	3.9865	4.5430	5.5971	7.3692	11.419	8
9	3.5395	3.6274	3.7080	3.7825	4.3447	4.7479	5.3702	6.5464	8.5217	13.045	9
10	4.1911	4.2889	4.3784	4.4612	5.0840	5.5294	6.2157	7.5106	9.6850	14.677	10
11	4.8637	4.9709	5.0691	5.1599	5.8415	6.3280	7.0764	8.4871	10.857	16.314	11
12	5.5543	5.6708	5.7774	5.8760	6.6147	7.1410	7.9501	9.4740	12.036	17.954	12
13	6.2607	6.3863	6.5011	6.6072	7.4015	7.9667	8.8349	10.470	13.222	19.598	13
14	6.9811	7.1154	7.2382	7.3517	8.2003	8.8035	9.7295	11.473	14.413	21.243	14
15	7.7139	7.8568	7.9874	8.1080	9.0096	9.6500	10.633	12.484	15.608	22.891	15
16	8.4579	8.6092	8.7474	8.8750	9.8284	10.505	11.544	13.500	16.807	24.541	16
17	9.2119	9.3714	9.6171	9.6516	10.656	11.368	12.461	14.522	18.010	26.192	17
18	9.9751	10.143	10.296	10.437	11.491	12.238	13.385	15.548	19.216	27.844	18
19	10.747	10.922	11.082	11.230	12.333	13.115	14.315	16.579	20.424	29.498	19
20	11.526	11.709	11.876	12.031	13.182	13.997	15.249	17.613	21.635	31.152	20
21	12.312	12.503	12.677	12.838	14.036	14.885	16.189	18.651	22.848	32.808	21
22	13.105	13.303	13.484	13.651	14.896	15.778	17.132	19.692	24.064	34.464	22
23	13.904	14.110	14.297	14.470	15.761	16.675	18.080	20.737	25.281	36.121	23
24	14.709	14.922	15.116	15.295	16.631	17.577	19.031	21.784	26.499	37.779	24
25	15.519	15.739	15.939	16.125	17.505	18.483	19.985	22.833	27.720	39.437	25
26	16.334	16.561	16.768	16.959	18.383	19.392	20.943	23.885	28.941	41.096	26
27	17.153	17.387	17.601	17.797	19.265	20.305	21.904	24.939	30.164	42.755	27
28	17.977	18.218	18.438	18.640	20.150	21.221	22.867	25.995	31.388	44.414	28
29	18.805	19.053	19.279	19.487	21.039	22.140	23.833	27.053	32.614	46.074	29
30	19.637	19.891	20.123	20.337	21.932	23.062	24.802	28.113	33.840	47.735	30
31	20.473	20.734	20.972	21.191	22.827	23.987	25.773	29.174	35.067	49.395	31
32	21.312	21.580	21.823	22.048	23.725	24.914	26.746	30.237	36.295	51.056	32

Şekil 5.4 - Erlang B Tablosunun bir bölümü (Ericsson, 1998d)

Kabullenmeler şunlardır:

- Kuyruk olmaması
- Trafik kanallarından daha fazla sayıda abone olması
- Tahsis edilmiş trafik kanallarının olmaması
- Rastgele trafik oluşması (poisson dağılışı)
- Arama bloke edilir edilmez tekrar arama oluşmaması

Erlang B tablosu, bir hücrenin trafik kanalı sayısına göre kabul edilen GoS değeri ile kaç Erlang trafik alabileceğini gösterir. Örneğin 2 adet TRU (Alıcı verici birim)'ya sahip bir hücrede 16 Time Slot (2x8) vardır. Bunlardan 1 Time Slot BCCH, en az 1 Time Slot ta SDCCH için kullanılır. Geriye kalan 14 Time Slot ise trafik kanalı (TCH) olarak kullanılır. Erlang B tablosuna bakıldığında 14 trafik kanalına sahip bu hücrenin kabul edilebilir %2 GoS oranıyla alabileceği trafik 8.2003 Erlang'tır. %2 GoS değeri 100 aramadan 98'inin başarılı olacağını gösterir.

Abone başına düşen ortalama trafik değeri tahmin edilebiliyorsa hücrenin servis verebileceği abone sayısı bulunabilir. Bugüne kadar yapılan çalışmalar sonucunda, günün en yoğun saatinde abone başına düşen ortalama trafik değeri 15–20 mE olarak bulunmuştur. Bu değer, bir saat içinde bir aramanın ortalama 54–72 saniye içinde sonlandığı anlamını taşımaktadır. Her abonenin 20 mE (0.025 E) trafik yarattığını düşünürsek, 8.2 Erlang trafik alabilen bir hücrenin 328 (8.2 / 0.025) aboneye servis verebileceğini görürüz. Buradaki 328 sayısı 14 trafik kanalına sahip bir hücrede 1 saat içerisinde rastgele trafik oluşması sonucunda mümkündür. 14 full-rate trafik kanalına sahip bir hücreden aynı anda sadece 14 kişi konuşabilir.

Şebeke boyutlandırılması yapılırken kullanılacak hücreler belirlendikten sonra bu hücrelerde kaç adet trafik kanalının kullanılacağının belirlenmesi gerekir. Gerekli olan trafik kanalı sayısı ile her hücrede kullanılacak taşıyıcı frekans sayısı belirlenir.

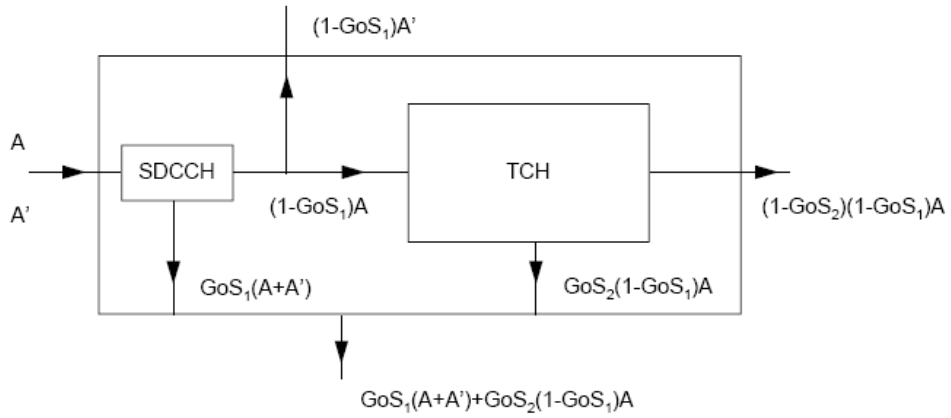
Aboneler mobil olduğu için gün içerisinde bir alandan başka bir alana yer değiştirebilirler. Belirli bir alandaki (örneğin bir havalimanı) belli sayıda abonenin yaratacağı trafiği karşılayabilmek için bir hücrede kaç tane taşıyıcı frekans kullanılması gerektiğine karar verilirken oluşacak trafiğin günden güne farklılık gösterebileceğine dikkat edilmelidir. Hatta aynı gün içerisinde farklı saatlerde farklı trafik değerleri oluşabilir.

Trafik kanalları yanında sinyalleşme kanallarının (SDCCH) sayısının belirlenmesinde önemlidir. Örneğin farklı LA (Location Area) sınırlarındaki hücrelerde "Location Update" işleminin fazla yapılmasından dolayı daha çok sinyalleşme kanalına ihtiyaç olabilir. Ayrıca bir otoyola servis veren hücreler arasında çok sayıda "Handover" işlemi yapılabileceğinden bu tip hücrelerde de sinyalleşme kanalı sayısına dikkat etmek gerekir. Bir hücredeki sinyalleşme kanalı ihtiyacını hesaplamak için hücrenin SDCCH kullanım prosedürlerine dikkat etmek gerekir. Bu prosedürler şunlardır:

- Mevkiinin uygun hale getirilmesi (Location updating)
- Periyodik kayıtlanma (Periodic registration)
- IMSI bağlanması/koparılması (IMSI attach/detach)
- Arama kurulumu (Call setup)
- Kısa mesaj (SMS)
- Faks (Facsimile)
- Ek servisler (Supplementary services)

Bunlar dışında yanlış erişim sayısı da tahmin edilmelidir.

Trafik kapasitesi hesaplanmasında kullanılan servis derecesinin (Grade of Service) belirlenmesinde çağrının iki farklı kanaldan geçtiği unutulmamalıdır. Arama önce SDCCH üzerinde başlar ve TCH üzerinde devam eder. Şekil 5.5 'te bu durum görülmektedir.



Şekil 5.5 - Bir çağrının iki farklı kanaldan geçmesi (Ericsson, 1998b)

Burada A, normal bir çağrının SDCCH üzerinde oluşturduğu trafiktir. A' ise SDCCH üzerinde gerçekleştirilen prosedürler için oluşan trafiktir. SDCCH ve TCH üzerinden geçen çağrılar için elde edilen genel servis derecesi GoS_T, şu şekilde gösterilir:

$$GoS_T = GoS_1 + (1\text{-}GoS_1)GoS_2$$

Burada GoS_1, SDCCH üzerindeki servis derecesidir. GoS_2 ise TCH üzerindeki servis derecesidir. Sinyalleşme kanallarına verilen servis derecesinin trafik kanallarına verilen servis derecesinden daha iyi olması gerekir. Çünkü bu değer genel servis derecesini (GoS_T) etkilediğinden tüm sistemin performansını belirler.

En doğru SDCCH boyutlandırması her hücreye özel olarak bakılarak sağlanır. Her hücrenin SDCCH ve TCH yoğunluğuna göre boyutlandırma (dimensioning) yapılmalıdır. Optimum çözüm olarak mümkün olabildiğince trafik kanalı kullanımı ile GoS_1 değerinin GoS_2 değerinin dörtte birini geçmemesi sağlanmalıdır (Ericsson, 1998b).

Sinyalleşme kanalları (SDCCH), GSM deki tanımlamalardan ötürü sadece dördün katları şeklinde tahsis edilebilir. Bir hücre en fazla 128 sinyalleşme kanalı içerebilir.

5.4 Kanal Kullanımı

Kanal kullanımı aşağıdaki örnekle açıklanabilir:

En yoğun saatte, 33 E trafik yaratabilen abonelerin bulunduğu bir bölgede, bu abonelere servis vermesi gereken bir hücre için gerekli olan trafik kanalı (TCH) sayısı hesaplanacaktır. Yoğun saatteki kayıp değeri %2 yi geçmemelidir. Bu gereksinimlerle Erlang B tablosu incelenirse gerekli olan TCH sayısının 43 olduğu görülür (Şekil 5.6).

n	.007	.008	.009	.01	.02	.03	.05	.1	.2	.4	n
43	30.734	31.069	31.374	31.656	33.758	35.253	37.565	42.011	49.851	69.342	43

Şekil 5.6 - Erlang B Tablosunun bir bölümü (GoS : %2, Trafik: 33 E) (Ericsson, 1998d)

Eğer bu bölgeye bir hücre yerine beş hücre ile servis verilseydi, bu beş hücrenin toplam trafiği yine 33 E olacaktı. Burada kabul edilebilir GoS değeri yine %2 ’dir. 33 E ’lık trafiği beş hücrenin paylaşması sonucu oluşan trafik dağılımı Şekil 5.7 ’de görülmektedir. Şekilde 33 E ’lık trafik için tek bir hücrede 43 trafik kanalı yeterli iken, aynı trafiğin beş hücreye dağılması sonucunda toplamda daha fazla trafik kanalına (62) ihtiyaç duyulduğu görülmektedir.

Bu örnek daha çok trafik kanalı içeren büyük bir hücrenin daha verimli olduğunu göstermektedir. Kanal kullanımını hesaplamak için önce Erlang B tablosunda 43 kanalın, GoS %2 değeri ile alabileceği 33.758 E trafik değeri %98 (100 - 2) ile çarpılarak verimli trafik (33.083 E) bulunur. Bu değer tek hücreli durumdaki kanal sayısına bölünür. Böylece ilk durumdaki kanal kullanımı 33.083 / 43 = %77 olarak bulunur. Ancak bu hücrenin daha küçük hücrelere bölünmesi sonucu kanal kullanımı azaldığı için daha fazla trafik kanalına ihtiyaç duyulmaktadır.

Cell	Traffic (%)	Traffic (E)	No. of channels	Channel utilization (%)
A	40	13.20	21	62
B	25	8.25	15	54
C	15	4.95	10	49
D	10	3.30	8	40
E	10	3.30	8	40
Σ	100	33.00	62	

Şekil 5.7 - Bir hücrenin küçük hücrelere bölünmesi sonucu oluşan trafik dağılımı (Ericsson, 1998b)

Kapasite ve girişim problemlerinden dolayı her zaman en verimli kanal kullanımı gerçekleştirilemez. Bu yüzden şebekede uygulanan çözümlerin, verim ve kalite arasında denge sağlaması gereklidir (Ericsson, 1998b).

5.5 Frekans Planlama Yöntemleri

Bölgede kullanılacak frekans planlama tekniği hücre planlamanın kalitesine göre değişkenlik gösterir. Frekans planı, şebekedeki kaliteyi ve kapasiteyi etkileyen bir faktördür.

Şebekelerde kullanılabilecek çok sayıda frekans planlama teknikleri mevcuttur. Bunlardan bazıları; Open Planning, Strict MRP (Multiple Reuse Pattern), Modified MRP, MRP Base-band Hopping, Synthesizer Hopping (FLP - Fractional Load Planning) ’dir. Şebekenin ve bölgenin yapısına göre frekans planlarında değişiklikler yapılarak özel frekans planları ve uygulama teknikleri oluşabilir.

Open planning belli kriterler içerisinde spektrum verimliliği yüksek olan bir planlama şeklidir. Ancak hücre planlamanın uygun yapılmadığı

durumlarda hata olasılığı yüksektir. Planlamada Modified veya Strict MRP yapısının seçilmesi için bazı gereksinimler oluşmalıdır. Ortalamada TRU/Hücre ’nin yüksek olduğu bir bölgede, Modified MRP metoduna geçilebilir. Bu şekilde benzer bir spektrum verimliliğine ulaşılabilir. Aynı zamanda daha planlı bir frekans planı elde edilmiş olur ve frekanslar daha etkili kullanıldığından TRU eklemek kolaylaşır. Modified MRP metodunu uygulamak, Strict MRP metoduna nazaran daha kolaydır. Bunun sebebi Strict MRP gibi her katmanında belli sayıda frekansın bulunmamasıdır.

Frekans planları tüm şebekedeki kapasitenin kullanıldığı düşünülerek hazırlanır. Bunun anlamı, TRU’lara atanan frekansların tümünün havada bulunduğu düşünülür. Yeniden kullanılma sıklığı (reuse) şebekenin kapasitesini artırması yanında kaliteyi kötü yönde etkilemektedir. Kaliteden ödün vererek kapasite arttırılabilir.

Base-band hopping metodunu kullanan şebekelerde TRU eklendikçe hücrelerde kullanılan frekans sayısı da artmaktadır. Dolayısıyla kapasiteyi arttırdıkça bölgede kullanılan frekans sayısı artmakta ve sınırlı sayıda olan frekansları daha sık kullanma zorunluluğu doğmaktadır. Sonuçta tekrar kullanımın sıklığından doğan kalite düşüklüğüne karşın kapasite artmaktadır. Kapasite artırılması istenmediği durumlarda, kalitede iyileşme sağlanması amacıyla yeni frekans planı hazırlanabilir. Frekans planlama teknikleri kullanılarak şebekenin kalitesi iyileştirilebilir, baz istasyonları arası uzaklık azaltılarak kapasite artırılabilir veya hücrelere frekans problemi yaşatmadan TRU eklenebilir.

Frekans planlama metotları şebekede kullanılan özelliklerle yakından ilgilidir. Hücrelere ne kadar frekans atanacağı veya planlama yaparken parametrelerin nasıl olacağı bu özellikler tarafından belirlenir. Burada en etkileyici özellik frekans hoplatma (Frequency Hopping) özelliğidir ve kullanılan frekans sayısını belirler:

Base-band hopping metodunda;

- Hücredeki TRU sayısı = Hücrede kullanılan frekans sayısı
- Her TRU, sabit bir frekansta yayın yapar
- Konuşmalar, hücredeki TRU ’lar üzerinde yer değiştirir

Synthesizer hopping metodunda;

- Hücredeki TRU sayısı < Hücrede kullanılan frekans sayısı
- Her TRU, verilen frekans grubundaki tüm frekanslarda sırayla yayın yapar
- Konuşmalar, hücredeki TRU lar üzerinde yer değiştirmez

Base-band hopping metodunda BCCH frekansları da hoplamaktadır. Böylece BCCH katmanında kullanılan frekansların kullanım sıklığı, hoplamanın kalite üzerindeki olumlu etkisi nedeniyle artmaktadır. Synthesizer hopping metodunda BCCH frekansları hoplamamaktadır. Bu sebepten dolayı Base-band 'deki BCCH katmanı kalitesine ulaşmak için daha fazla BCCH frekansı kullanmak gerekir. TCH frekanslarında kullanılan tekrar kullanım değerleri BCCH frekanslarına nazaran daha düşük tutulabilir. Bunun sebebi, BCCH frekanslarının içerisindeki kanalların, sinyalleşme ve şebekeye ulaşma amaçlı kullanıldığı için servis kalitesi (QoS) değerlerinin daha yüksek tutulmaya çalışılmasıdır.

Synthesizer hopping 'in uygulama yöntemlerinden olan FLP 1/1 ve FLP 1/3 metotları, şebekede iyi kalite sağlamak için kolay ve güçlü bir yöntemdir. Bu yöntemle frekans planı daha kolay hazırlanır. Kapasite ihtiyacı sebebiyle hücrelere eklenen yeni TRU'lar için temiz frekans aramaya gerek yoktur. Synthesizer hopping, hybrid combiner kullanılan sistemlerde çalışır.

Synthesizer hopping FLP 1/1 metodunda her hücreye aynı frekans grubu atanır. Hücrelerin birbirlerine girişimde bulunmasını önlemek amacı ile HSN (Hopping Sequence Number – Hoplama Sıra Numarası) planı yapılır. HSN hücrede tanımlı frekansların hangi sıra ile hoplayacağını belirler. 63 adet birbiri ile ortogonal HSN mevcuttur. Hücre içerisindeki TRU 'ların birbiri ile girişimde bulunmasını önlemek amacı ile her TRU 'ya belli bir sapma (offset) verilir. Bu parametrenin adı MAIO 'dur (Mobile Allocation Index Offset). Her hücre TRU sayısı kadar MAIO değeri içermelidir. Örnek olarak 20 frekanslık bir frekans

grubu kullanılırken 3 TRU içeren bir hücreye 0, 2 ve 4 MAIO değerleri verilirse aynı anda ilk TRU 1. frekans ile yayın yaparken, ikinci TRU 3. frekans ve üçüncü TRU da 5. frekans ile yayın yapar. İlk TRU 2. frekans ile yayın yapmaya başladığında ise ikinci TRU 4. frekans ve üçüncü TRU 6. frekans ile yayın yapmaya başlar.

FLP 1/3 metodunda ise 3 farklı frekans grubu mevcuttur, aynı baz istasyonundaki hücrelere farklı frekans grupları atanarak hücrelerin birbirleri ile girişimde bulunması önlenmiş olur.

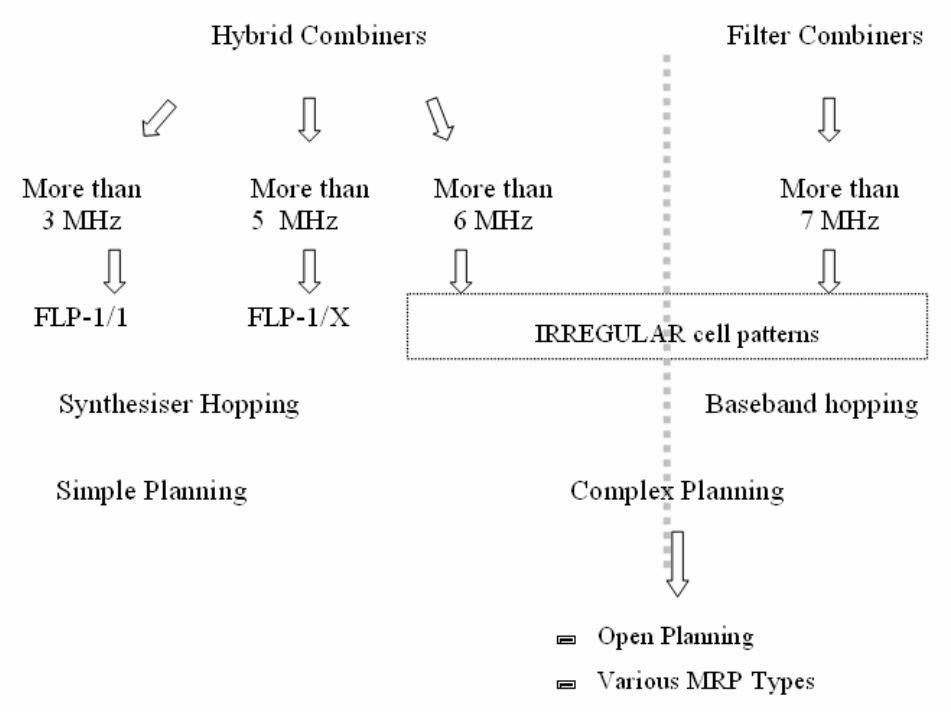
Frekans hoplama yöntemi, verimi yüksek spektrum kullanımı için temel gereksinimdir. Frekans hoplama yöntemi her TDMA frame 'inin iletiminden sonra iletişimin başka frekansta devam etmesi anlamına gelmektedir. Bu yöntemin kazancı bir hücredeki tüm konuşmaların ortalama bir girişime maruz kalmalarıdır. Hoplamasız yöntemlerde, her konuşma sabit bir frekansta gerçekleştiği için temiz frekanstaki bir konuşma iyi kalitede sağlanırken kirli frekanstaki diğer bir konuşma kötü kaliteye maruz kalabilir. Frekans hoplama yöntemleri, şebekedeki girişim (interference) değerini ortalama bir değerde tutarak konuşmaları belli bir kalitede gerçekleştirir.

FLP yöntemi ile hoplamayan sistemlere göre çok daha iyi konuşma kalitesi elde edilmesi beklenemez. Yoğunluğu yüksek şehirlerde –90 dB sinyal seviyesinin altında gerçekleşen konuşmalarda kalite bozuklukları yaşanabilmektedir. Hücre sınırlarındaki kalite bozuklukları synthesizer hopping yöntemi ile hoplayan yüksek sayıdaki frekans sayesinde ortalama bir değer kazanır. Bu sayede hücre sınırlarında oluşan ani kalite bozulmaları gerçekleşmez, dolayısıyla hücrenin kaliteli konuşma alanı genişler.

5.5.1 Frekans Planlama Metodu Seçimi

Frekans planlama metodu birçok değişkene bağlıdır. Hangi frekans metodunun kullanılacağı ağırlıklı olarak frekans sayısı ve şebeke yapısıyla ilgilidir. Genel anlamda frekans planlama metodu önerilerinin bulunduğu grafik Şekil 5.8'de görülmektedir. Spektrumu dar (5 MHz ve altı) operatörlerin yüksek kalite değerli ve kapasiteli şebeke kurmaları çok zordur. Kalite ve kapasitelerini artırmak amacı ile özel frekans

planlama teknikleri kullanmalıdırlar. Şekil 5.8'de synthesizer hopping 'in bu tür şebekeler için ideal olduğu görülmektedir.

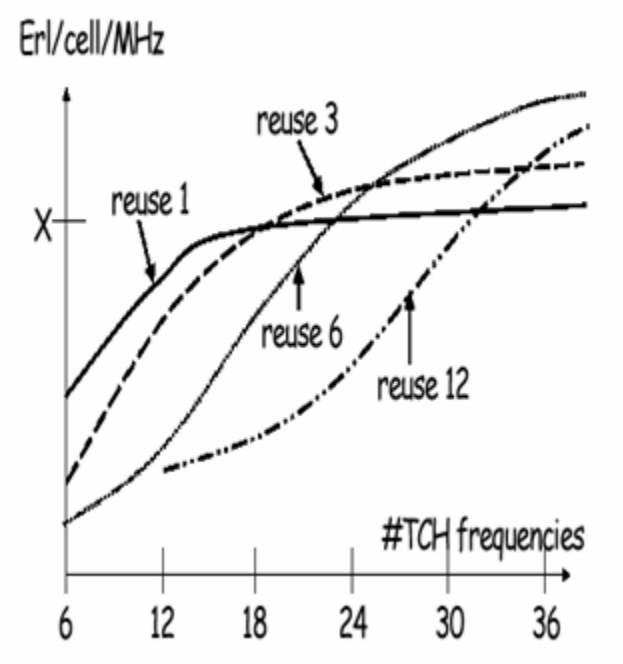


Şekil 5.8 - Frekans planlama metodu önerileri (Ericsson, 1998e)

Spektrum genişliğinin yüksek olmasına rağmen önemli alanlarda trafiğin yoğun olması nedeni ile sıkışmalar, kalite problemleri ve frekans bulamama gibi durumlar oluşabilmektedir. MRP yapısı ile BCCH ve TCH frekansları ayrılmakta, bu şekilde şebekede daha düzenli frekans planı elde edilmektedir.

Frekans planlama yöntemi hücre planının kalitesine yüksek ölçüde bağlıdır. Genelde FLP 1/1 ve 1/3 metotları düzensiz hücre planına sahip şebekelerde frekans bandı dar olmadığı müddetçe kullanılmazlar. Çok dar spektruma sahip şebekelerde de FLP 1/1 en iyi çözümdür. Şekil 5.9, spektrum kullanımını, seçilen frekans tekrar kullanımı (reuse) ile TCH

frekans sayısının fonksiyonu cinsinden ifade etmektedir. Tekrar kullanım (reuse), Toplam Frekans sayısı / Ortalama TRU işlemi ile hesaplanır.



Şekil 5.9 - TCH frekans sayısı ve tekrar kullanıma göre spektrum kullanımı (Ericsson, 1998e)

Y ekseninin değeri, şebekenin yapısına ve hücre planına bağlı olduğundan değişkendir. X sayısı 2 ile 5 Erl/cell/MHz arasında değişebilir.

TCH frekansı sayısının artışıyla frekans kullanımının, farklı tekrar kullanım planlarına göre değişken bir şekilde arttığı Şekil 5.9'da görülmektedir. Reuse 1, 3, 6, 12 arasında uygulama basitliği ve kapasite açısından ters bir ilişki mevcuttur. 36 frekansın TCH için ayrıldığı bir şebekede, reuse 6 olan yapı spektrumun en verimli kullanıldığı yapıdır.

5.5.2 Ayrık Frekans Dizilimi

BCCH frekans planlaması, TCH frekans planlamasına göre daha önemlidir, çünkü Broadcast ve Common Control kanalları BCCH üzerinden yayınlanmaktadır. Bu kanal üzerinden hücrelere bağlanılır. Ayrıca paging ve komşu hücrelere geçiş (Locating) gibi önemli bilgiler, bu kanalın bulunduğu frekans üzerinden taşınmaktadır. Özetle konuşmanın başlaması ve düzgün sürmesi için BCCH frekansının temiz olması gerekmektedir. Bu durumda BCCH frekanslarında aynı kanal (co-channel) ve ardışık kanal (adjacent-channel) kavramları ön plana çıkmaktadır.

Bir BCCH frekansına, bir TCH frekansından çok başka bir BCCH frekansı tarafından girişim yapılması daha yüksek ihtimaldir. Çünkü bir hücrede hiçbir konuşma yapılmadığı sırada bile, hücre BCCH frekansından tam güçle yayın yapmaya devam eder. Bunun sebebi ise, bölgedeki mobil cihazların hangi hücreden servis alacağının, hücre sinyal seviyesine göre belirlenmesidir. Böylece mobil cihazlar, bölgedeki hücrelerin BCCH frekanslarının sinyal seviyesini ölçerek servis alacağı hücreye karar verirler.

TCH frekanslarında ise, hücrede konuşma yapılmıyorken aynı bölgedeki hücrelerde girişim yaratmamak için bir yayın yapılmaz. Konuşma yapılırken ise yine bölgedeki hücrelerde girişim yaratmamak için, TCH frekansında konuşma kalitesini bozmayacak şekilde minimum güç ile yayın yapılır. Bu yüzden bir TCH frekansına bir BCCH frekansı tarafından girişim yapılması daha kolaydır.

Ayrık frekans diziliminde BCCH frekanslarının birbirleriyle ardışık olma durumları yoktur. Ancak BCCH frekansı sayısına ve spektrumun darlığına göre BCCH frekansları ile TCH frekansları birbirleriyle ardışık olabilirler. BCCH için tahsis edilen frekanslar birbirleriyle ardışık olmadıkları için ve TCH frekanslarının ardışık etkileri daha az olduğu için ardışık kanal girişimi (Adjacent Channel Interference) görülme olasılığı azdır. Bu yöntemle BCCH performansı artmakta ve bu frekanslarda oluşan kalite problemlerinin tespit edilmesi kolaylaşmaktadır. Ancak TCH frekans planına dikkat edilmelidir. TCH

frekanslarında ardışık kanal girişimi görülmemesi için aynı bölgede ardışık bir BCCH frekansı kullanımından kaçınılmalıdır.

5.5.3 Sürekli Frekans Dizilimi

Sürekli frekans dizilimi yönteminde TCH ve BCCH frekansları arasında ardışık kanal girişiminin oluşma olasılığı yoktur. BCCH frekansları kendi aralarında ardışık olabilirler. Bu yüzden BCCH frekans planına dikkat edilmelidir. BCCH frekanslarında ardışık kanal girişimi görülmemesi için aynı bölgede ardışık BCCH frekansı kullanımından kaçınılmalıdır.

TCH frekansları da kendi aralarında ardışık olabilirler. Ancak konuşma yapılmadığı anlarda yayının kesilmesi ve konuşma anında düşük güçte yayın yapılabilmesi özellikleri ile TCH frekanslarında ardışık kanal girişimi oluşması, BCCH frekansından oluşan girişime göre daha etkisizdir. Yinede en kaliteli iletişim için aynı bölgede ardışık TCH frekansı kullanımına dikkat edilmelidir.

Sürekli frekans diziliminde BCCH ve TCH frekansları birbirlerinden bağımsız oldukları için bu yöntemin uygulanması ve tuning işlemi daha kolay gerçekleştirilir.

5.5.4 MRP (Multiple Reuse Pattern)

MRP, yüksek kapasiteli şebekelerde kullanılan frekans planlama tekniklerinden biridir (Şekil 5.10). İki şekilde şebekede uygulanması mümkündür:

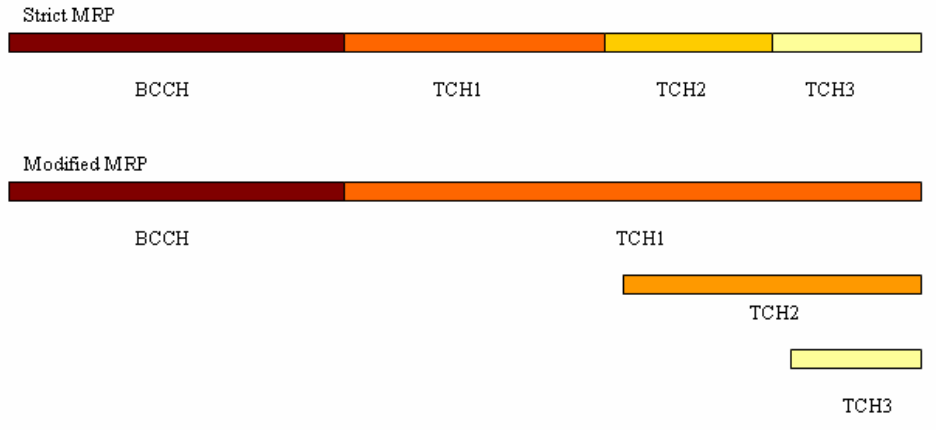
- Strict MRP
- Modified MRP

MRP'nin uygulanmasındaki ana sebepler şunlardır:

- Her TRU için ayrılmış frekans bandı sağlar
- Daha düzenli frekans atanması sonucunda, kapasiteyi arttırmak için yeni hücreler oluşturmak yerine mevcut hücrelerde 8 TRU 'ya kadar kullanım imkanı sağlar

TRU sayılarının 2 - 6 arasında değiştiği bir şebekede, Open Planning yöntemi frekans spektrumunu etkin kullanmada daha verimlidir. Ancak çoğunluğu 5 TRU 'ya yakın çalışan hücrelerin bulunduğu bölgelerde Strict MRP kullanılması uygundur. Bu bölgedeki hücrelerin çoğunluğunun TRU sayısı 5 ise Strict MRP 'den elde edilecek spektrum verimliliği Open Planning 'e göre aynıdır. Birbirinden farklı frekanslar içeren Bcch, Tch1, Tch2 ve Tch3 gruplarından oluşur. Az sayıda TRU içeren yerlerde Bcch ve Tch3, orta sayıdaki yerlerde Bcch ve Tch2, çok sayıdaki yerlerde Bcch ve Tch1 grupları kullanılır.

Modified MRP ise planlaması kolay ve kısıtlaması strict MRP 'ye göre daha az olan planlama şeklidir. Ancak bu yöntemde hücre planlamacının bölge bilgisini kullanması gerekir. Modified MRP spektrum verimliliğini daha iyi sağlamaktadır. Burada Tch1 grubu, Tch2 ve Tch3 grubunu kapsar. Tch2 grubu da Tch3 grubunu kapsar.



Şekil 5.10 - MRP yapıları

Strict MRP ’nin Open Planning ’e göre avantajları ve dezavantajları vardır.

Avantajlar:

- Bir katmanı planlarken kısıtlamalar eklenebilir
- Frekans planını geliştirmek daha kolaydır
- TRU eklemek daha kolaydır
- Kontrollü bir yapı vardır

Dezavantajlar:

- Katmanlara atanacak frekans sayısını tahmin etmek zordur
- Frekans spektrumu verimli olarak kullanılamaz

Strict MRP ’de BCCH ve TCH katmanları kendilerine özel frekanslardan oluşmaktadır. TCH kanal gruplarındaki frekans sayıları, katmanların içindeki frekansların yeniden kullanım (reuse) değerleri birbirlerine eşit olacak şekilde dağıtılmalıdır.

TRU sayısı arttıkça hoplayan frekans sayısı arttığından girişimin etkisi azalmaktadır. Bu sebepten üst katmanlara çıkıldıkça frekans sayısı azalır.

Modified MRP ’yi planlamak ve uygulamak Strict MRP ’ye göre daha kolaydır. Uygulanmasında hücre planlamacının saha bilgisine daha çok ihtiyaç duyulan Modified MRP, spektrum verimliliğinin yüksek olduğu bir yöntemdir. Yapısı Şekil 5.10’da belirtilen Modified MRP metodunda, 1. katmanda tüm TCH frekansları bulunmaktadır. Üst katmanlara çıkıldıkça frekans sayıları azalır.

Modified MRP ’de frekans planına en üst katmandan başlanmalıdır (Şekil 5.10’da TCH3 katmanı). Öncelikle en yüksek sayıda TRU ’ya sahip hücrelere plan yapılmalıdır. En üst katmanda kullanılmayan frekans

mevcut ise, bu frekanslar bir alt katmana eklenebilir. Aynı kural, diğer katmanlar için de uygulanırsa spektrum daha verimli kullanılır.

Frekans planı şebekeye uygulandıktan sonra;

- Şebekenin performansı incelenmelidir. Kötü kaliteyle çalışan hücreler belirlenmeli ve frekans değişikliklerine gidilmelidir
- Hücrelerdeki parametre uyumsuzlukları kontrol edilip raporlanmalıdır
- MRP'nin uygulandığı alanda bazı hücresel istatistiklere de bakılmalıdır:
  - Trafik ve sıkışma zamanları
  - Drop/ERL (Konuşma kesilme sayısı / Trafik)
  - Bad Quality Drop % (Kötü kalite kaynaklı konuşma kesilme yüzdesi)
  - Bad Quality Urgency Handover % (Kötü kalite kaynaklı hücre değiştirme yüzdesi)
- Bölgede Drive Test yapılmalıdır. Drive Test, özel bir telefon ve bilgisayardaki yazılım ile bölgedeki yollarda saha ölçümleri yapılmasıdır.

Genel bir sonuç olarak, normal trafik yoğunluğu olan alanlarda open planning, yüksek ve çok yüksek trafik yoğunluğu olan alanlarda Modified MRP metodu planlama için uygun bir yöntemdir. Çok yüksek trafik yoğunluğu olan alanlarda, frekans tahsisi çok zor ise Base-band hopping 'den Synthesizer hopping 'e geçilebilir. Böylece frekans problemi olmaksızın TRU eklenebilir ya da yeni baz istasyonları açılabilir.

6 HÜCRELERE FREKANS ATAMA UYGULAMASI

Çok sayıda abonenin bulunduğu GSM şebekelerinde, mobil telefon servislerinin kullanımının artması ve bunun yanında ses ve görüntü içeren yeni servislerin de gelmesiyle birlikte operatörler bu trafiği karşılayabilmek için ağ kapasitesini arttırmak zorunda kalırlar. Kapasiteyi arttırmak için yeni hücreler eklenebilir ya da mevcut makro hücreler bölünebilir. Bunun yanında operatörler gelişmiş ağ özelliklerini (advanced network feature) kullanarak frekans tekrar kullanım faktörünü azaltırlar. Spektrumun limitli olmasından ve donanımsal kısıtlardan dolayı mevcut makro hücrelerde kapasite artırımının konuşma kalitesinde kötüleşmeye neden olması kaçınılmazdır (Halonen et al., 2003). Kaliteyi korumak için makro hücrelerin mikro hücrelere bölünmesiyle aynı alana daha çok hücre ile servis verilecektir. Bu durumda frekans planlama önem kazanacaktır.

Bu bölümde, şebekeden servis alan abonelerin mobil cihazları tarafından yapılan ölçümleri kullanarak frekans planlaması yapan bir uygulama anlatılacaktır.

6.1 Giriş

Operatörler, ağ'daki artan talebi karşılayabilmek için, mevcut hücrelerin kapasitelerini donanımsal eklemeler ile genişletebilir. Bunun için, hücredeki TRU adı verilen alıcı-verici birimlerin sayısı arttırılır. Böylece eklenen her TRU ile hücreye ek bir frekans tanımlanarak kapasite arttırılır. Ancak GSM radyo şebekesinde spektrumun limitli olmasından dolayı, frekans sayıları sınırlıdır. Ayrıca donanımsal kısıtlar nedeniyle hücrelerde ardışık frekanslar kullanılamamaktadır. Eklenen frekansın mevcut ve komşu hücrelerde girişim yaratıp konuşma kalitesini düşürmemesi için çok iyi seçilmesi gerekir (Eisenblaetter et al., 1999).

TRU eklenmesinin yeterli olmadığı durumlarda mevcut makro hücrelerin mikro hücrelere bölünmesi ve bunun için de, mevcut baz istasyonunun servis sahası küçültülerek bu istasyonun altına yeni bir istasyon açılması gerekir. Şehir merkezlerinde, bir GSM operatörünün

baz istasyonları arasındaki mesafe 300 metrenin altındadır. Ancak çevresel tepkilerden dolayı baz istasyonları çoğu zaman uygun yerlere konumlandırılamamaktadır. GSM operatörleri yeni baz istasyonları için uygun yer bulmada birbirleriyle de çekişme halindedir.

Hücre kapsama alanları homojen dağılmadığı için hücrelerin birbirleri ile kesiştiği çok sayıda alanlar mevcuttur. Eğer bu hücrelerde aynı ya da ardışık frekanslar kullanılmakta ise, kesişim bölgelerinde girişim oluşacaktır ve bu bölgede bulunan mobil cihazlar kötü kaliteyle servis alacaklar yada yüksek hata oranı nedeniyle servis alamayacaklardır

Bir hücreye frekansları atanırken diğer hücrelerle olan komşuluğu dikkate alınarak en uygun frekanslar verilmelidir. Ancak GSM operatörünün kullanabileceği frekans sayısı, atanacak frekans sayısının çok altında olduğu için aynı frekansı şebeke içinde birçok kez kullanmak gerekir. Örneğin 50 taşıyıcı frekansa sahip ve 20000 TRU içeren bir şebekede, her frekans ortalama 400 kez tekrar kullanılacaktır (Halonen et al., 2003).

Hücrelere en uygun frekansların verilebilmesi için bir girişim matrisine (Interference Matrix) ihtiyaç vardır. Aboneler tarafından yapılan her arama, girişim matrisi oluşturulmasına katkıda bulunur. Girişim matrisi, tamamen abonelerin ne zaman ve nerede konuştuklarını gösterir (Lundqvist et al., 1999).

Bu tez çalışmasında, şebekeden servis alan mobil cihazlar ile alınan ölçümler kullanılarak hazırlanmış bir girişim matrisini, girdi olarak kullanan bir program yazılarak hücrelere atanacak frekansların saptanması sağlanmıştır.

6.2 Mobil Ölçümlere Dayalı Frekans Planlama

Bir mobil cihaz, sadece ona en yakın olan baz istasyonundan sinyal almaz. Daha uzaktaki, aynı frekansta yayın yapan başka baz istasyonlarından da sinyaller alır. Buna aynı kanal girişimi (co-channel interference) adı verilir. Ardışık kanal ise ardışık bir frekans kullanan kanaldır. Aynı kanal girişimine göre ardışık kanal girişimi (adjacent

channel interference) daha az etkilidir. Frekans sayıları limitli olduğundan dolayı frekanslar tekrar kullanılmak zorundadır. Bu sebeple girişim problemi oluşmaktadır. Girişimi azaltmak için frekans planlamaya ihtiyaç vardır.

Mobil ölçümlere dayalı frekans planlama için bir hücreler arası bağımlılık matrisine (ICDM – Inter Cell Dependency Matrix) ihtiyaç vardır. ICDM, BCCH Allocation List (BA List) adı verilen ölçüm raporları kullanılarak oluşturulur.

6.2.1 BA List (BCCH Allocation List)

Bir BA List (BCCH Allocation List), baz istasyonundan mobil cihazlara gönderilen bir listedir. Bu liste, mobil cihazın hangi BCCH frekanslarında ölçüm yapması gerektiğini bilmesini sağlar. Mobil cihaz, bu BCCH frekanslarının sinyal seviyesini ve ölçülen bu frekanslarla ilişkili BSIC (Base Station Identity Code – Baz İstasyonu Kimlik Numarası) bilgisini baz istasyonuna gönderir. BSIC değeri, hangi BCCH frekansının hangi baz istasyonundan geldiğinin belirlenmesinde kullanılır (Lundqvist et al., 1999).

Eğer bir mobil cihaz, bir baz istasyonundan uzaklaşmaya başlarsa sinyal seviyesi azalmaya başlar ve başka bir baz istasyonuna yeterince yaklaşmışsa bu baz istasyonundaki hücreye aktarılır (handover). BA raporları, handover yapılması gereken durumlarda aktarma yapılabilecek hücrelerin belirlenmesinde de kullanılır.

6.2.2 ICDM (Inter Cell Dependency Matrix)

ICDM, bir hücrenin başka bir hücreye ne kadar girişimde bulunabileceğini trafik yüzdesi olarak gösteren bir matristir. ICDM, çok boyutlu bir array olarak da düşünülebilir. Bir hücrenin başka bir hücre ile aynı veya ardışık frekans kullanması durumunda, trafiğinin girişime uğrayacak yüzdesini, iki durum içinde ayrı ayrı gösterir.

BA listeleri değiştirilerek tüm BCCH frekanslarının ölçülmesi sağlanır. Mobil cihazlar, bir hücreden servis alırken BA listesindeki tüm BCCH frekansları için sinyal seviyesini ölçer ve BSIC değerlerini

çözmeye (decode) çalışırlar. Eğer çözme işlemi başarılı olursa mobil cihaz, ölçtüğü sinyal seviyeleri ve BSIC değerleri ile servis aldığı hücrenin sinyal seviyesini baz istasyonuna yollar. Ölçülen hücrelerin sinyal seviyelerinden servis alınan hücrenin sinyal seviyesi çıkarıldığında elde edilen fark, limitlerin üstündeyse bu hücreler potansiyel girişimcilerdir. Eğer potansiyel hücreler, servis alınan hücre ile aynı frekansı kullanırlarsa gerçek girişimciye dönüşeceklerdir.

Aynı kanal girişimi oluşmaması için servis alınan hücre ile aynı frekansa sahip başka bir hücrenin sinyal seviyesi farkının en az 9 dB olması gerekir. Ardışık kanal girişimi oluşmaması için ise servis alınan hücre ile ardışık frekansa sahip başka bir hücrenin sinyal seviyesi farkının en az -3 dB olması gerekir (Lundqvist et al., 1999).

6.3 Frekans Atama Yazılımı

Bu bölümde, mobil ölçümlere dayalı bir frekans planı oluşturmak için hücrelere uygun frekansları atayan bir yazılım anlatılacaktır. Geliştirmiş olduğumuz bu yazılım ile mobil ölçümler kullanılarak şebeke için uygun bir frekans planı elde edilebilir.

Program girdi olarak iki tablo kullanmaktadır. Bu tablolar şunlardır:

- Frekans planı yapılacak hücrelerin bilgilerini içeren hücre tablosu (Çizelge 6.1)
- ICDM (Inter Cell Dependency Matrix) tablosu (Çizelge 6.2)

Hücre tablosu “Hücre Adı”, “Taşıyıcı Sayısı” ve “Yoğun Saatteki Trafik (E)” kolonlarını içerir (Çizelge 6.1). “Hücre Adı” kolonu, 6 karakter içeren ve eşi olmayan hücre adı bilgisini belirtir. “Taşıyıcı Sayısı” kolonu, hücreye kaç adet taşıyıcı kanal atanacağını belirtir. “Yoğun Saatteki Trafik (E)” kolonu ise hücrenin gün içerisinde en yoğun saatteki trafik değerini Erlang cinsinden belirtir.

Çizelge 6.1 - Hücre tablosu

Hücre Adı	Taşıyıcı Sayısı	Yoğun Saatteki Trafik (E)
AHMTL1	2	3.8654
AHMTL2	5	26.8889
TURDA2	4	19.5540
TURDA1	5	23.3690
AKBAL1	3	9.8557

ICDM tablosu “Kaynak Hücre”, “Girişimci Hücre”, “Aynı Kanal Girişiminden Etkilenen Trafik (%)” ve “Ardışık Kanal Girişiminden Etkilenen Trafik (%)” kolonlarını içerir (Çizelge 6.2). “Kaynak Hücre” kolonu, ölçümleri alan mobil cihazın servis aldığı hücre adını belirtir. “Girişimci Hücre”, potansiyel girişimci hücrelerin adını belirtir. “Aynı Kanal Girişiminden Etkilenen Trafik (%)”, kaynak ve girişimci hücreye aynı taşıyıcı kanal atanması durumunda kaynak hücrede girişime uğrayacak trafik yüzdesini belirtir. “Ardışık Kanal Girişiminden Etkilenen Trafik (%)” ise kaynak ve girişimci hücreye ardışık taşıyıcı kanal atanması durumunda kaynak hücrede girişime uğrayacak trafik yüzdesini belirtir.

Geliştirilen yazılımın algoritması, maliyet hesabına ve hücrelere hangi sırayla frekans atama işlemi yapılacağının belirlenmesine dayanır. Frekans atama sırasının belirlenmesi için önceliklendirilmiş hücre tablosu oluşturulur. Çok sayıda potansiyel girişimci hücre içeren bir kaynak hücreye, temiz bir frekans bulmak çok daha zordur. Bu yüzden ICDM tablosunda en çok potansiyel girişimci hücre içeren kaynak hücrenin önceliği en yüksek olarak belirlenir. Potansiyel girişimci hücre sayısı

aynı olan kaynak hücreler önceliklendirilirken yoğun saatteki trafik değeri en yüksek olan kaynak hücrenin önceliği daha yüksek olarak belirlenir (Çizelge 6.3). Geliştirilen yazılım, frekans atama işlemine önceliklendirilmiş hücre tablosundaki en yüksek öncelikli hücreden başlar.

Çizelge 6.2 - ICDM tablosu

Kaynak Hücre	**Girişimci Hücre**	**Aynı Kanal Girişiminden Etkilenen Trafik (%)**	**Ardışık Kanal Girişiminden Etkilenen Trafik (%)**
AHMTL2	AKBAL1	0.38	0
AHMTL2	TURDA1	11.46	3.09
AHMTL2	TURDA2	7.24	0.7
AHMTL2	AHMTL1	80.62	41.44
AHMTL1	TURDA1	2.58	0
AHMTL1	TURDA2	0.54	0
AHMTL1	AHMTL2	50.15	13.89
AKBAL1	AHMTL1	3.45	0.23
AKBAL1	TURDA1	30.24	9.48
AKBAL1	TURDA2	10.03	2.77
TURDA1	AHMTL2	5.37	1.29
TURDA1	TURDA2	45.21	15.61
TURDA2	AHMTL2	12.82	4.63
TURDA2	TURDA1	70.33	24.68

Çizelge 6.3 - Önceliklendirilmiş hücre tablosu

Öncelik	Hücre Adı	Yoğun Saatteki Trafik (E)	Girişimci Sayısı
1	AHMTL2	26.8889	4
2	AKBAL1	9.8557	3
3	AHMTL1	3.8654	3
4	TURDA1	23.3690	2
5	TURDA2	19.5540	2

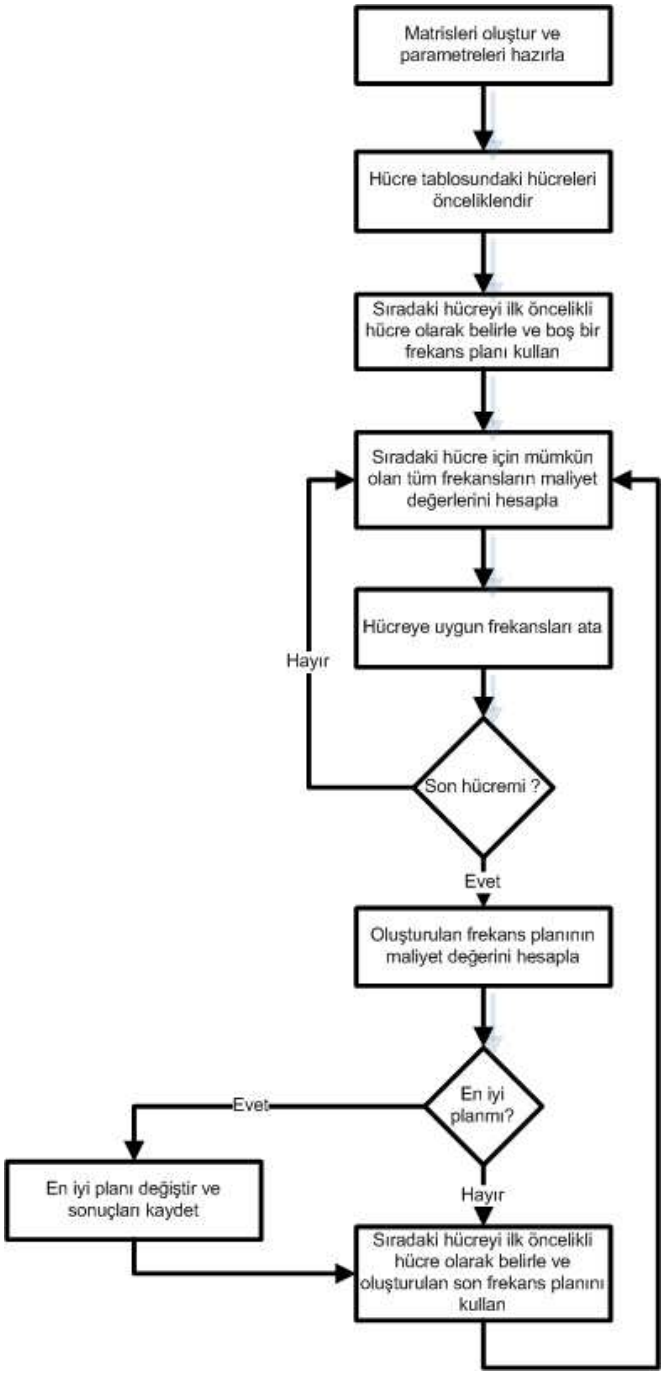
Geliştirilen yazılım, ICDM tablosundaki veriler ve yoğun saatteki trafik değerlerini kullanarak önceliklendirilmiş hücre tablosundaki her hücre için kullanım izni olan tüm frekansların maliyet değerlerini hesaplar. Bir kaynak hücre için bir frekansın maliyet değeri, aynı frekansı ve ardışık frekansları kullanan girişimci hücrelerin ICDM tablosundaki girişimden etkilenen trafik yüzdesi değerleri ile kaynak hücrenin trafik değeri çarpımlarının toplanması ile elde edilir. Ayrıca donanımsal kısıtlardan ötürü aynı baz istasyonundaki hücrelerde aynı ve ardışık frekanslar kullanılması kaliteyi kötü yönde etkilediğinden aynı baz istasyonunda aynı frekans kullanımında 10000, ardışık frekans kullanımında 1000 maliyet puanı maliyet değerine eklenir. Yazılım, her hücreye tek tek en düşük maliyet değerine sahip olan frekansları atar. Bir hücrenin maliyet değeri, ona atanan frekansların maliyet değerleri toplamına eşittir. Oluşturulan frekans planının maliyet değeri ise hücrelerin maliyet değerleri toplamına eşittir. Yazılım, en uygun frekans planını elde edebilmek için frekans planının maliyet değerini iyileştirmeyi amaçlar. Bunu için hücrelere birçok kez farklı frekans ataması yapabilir.

Geliştirilen yazılımın işleyiş biçimi Şekil 6.1'de görülmektedir. Bu yazılım sonsuz bir döngüde ya da kullanıcının belirleyeceği bir döngü sayısında çalışabilir. Kullanıcılar istediklerinde o ana kadar düşürülebilen en düşük maliyet değerine sahip kaydedilmiş frekans planına ulaşabilirler. Ayrıca kullanıcılar, BCCH ve TCH frekans planında kullanılacak taşıyıcı frekansları da planlama işlemine başlamadan önce belirtmelidir.

Örnek bir TCH planı oluşturmak için taşıyıcı kanallar 1 ile 15 arasında seçilerek Çizelge 6.1'deki hücre bilgileri ve Çizelge 6.2'deki ICDM bilgileri ile programı çalıştırdığımızda oluşan sonuç plan Çizelge 6.4'te görülmektedir.

Çizelge 6.4 - Oluşturulan sonuç plan

Hücre Adı	**Taşıyıcı 1**	**Taşıyıcı 2**	**Taşıyıcı 3**	**Taşıyıcı 4**	**Taşıyıcı 5**
AHMTL1	1	3			
AHMTL2	5	7	10	12	14
AKBAL1	5	7	15		
TURDA1	1	3	9	11	13
TURDA2	4	6	8	15	



Şekil 6.1 - Geliştirilen yazılımın işleyiş biçimi

Frekans planında kullanılmak için seçilen 1 ile 15 arasındaki taşıyıcı kanalların oluşturulan sonuç planındaki hücrelere göre maliyet değerleri Çizelge 6.5'te görülmektedir.

Çizelge 6.5 - Hücrelere göre taşıyıcı kanalların maliyet değerleri

Taşıyıcı Kanal	AHMTL1	AHMTL2	AKBAL1	TURDA1	TURDA2
1	**0,10**	10024,76	3,32	**0,00**	10013,8
2	0,00	2023,95	1,91	0,00	2009,65
3	**0,10**	10024,95	3,59	**1003,65**	10013,8
4	1000,56	1013,92	1,95	10010,87	**1005,73**
5	10001,9	**0,48**	**0,55**	2008,55	2,51
6	2001,09	1,95	0,99	10011,17	**1,81**
7	10001,9	**0,48**	**0,55**	2008,55	2,51
8	1000,56	2,78	1,92	10010,87	**1005,73**
9	1000,64	3,27	3,25	**1003,95**	10014,7
10	10001,9	**1,66**	1,87	1,25	2012,16
11	2001,17	3,08	2,98	**0,60**	10015,6
12	10001,9	**1,66**	1,87	1,25	2012,16
13	2001,17	3,08	2,98	**0,60**	10015,6
14	10001,9	**1,02**	1,21	1004,90	1007,33
15	1000,56	2,05	**0,99**	10010,87	**0,91**

Örnek olarak “TURDA1” hücresinin ikinci taşıyıcı kanalı olan 3 için maliyet değeri 1003,65 olarak hesaplanmıştır. Bu hücrenin, ICDM tablosunda iki tane potansiyel girişimci hücresi vardır. Bunlardan biri “AHMTL2” hücresi diğeri ise “TURDA2” hücresidir. 1003,65 değeri şu işlemler ile hesaplanmıştır:

- “AHMTL2” ve “TURDA2” potansiyel girişimcileri, 3 numaralı taşıyıcı kanalı içermedikleri için bu hücrelerden kaynaklı aynı kanal girişimi bulunmamaktadır.

- “AHMTL2” potansiyel girişimcisi, 2 ve 4 numaralı taşıyıcı kanalları içermediği için bu hücre kaynaklı ardışık kanal girişimi bulunmamaktadır.

- “TURDA2” potansiyel girişimcisi, 4 numaralı taşıyıcı kanalı içermektedir. “TURDA1” kaynak hücresi için, “TURDA2” girişimcisinin ICDM tablosundaki “Ardışık Kanal Girişiminden Etkilenen Trafik (%)” değeri %15,61 ’dir. Bu değer ile “TURDA1” kaynak hücresinin “Yoğun Saatteki Trafik (E)” değeri çarpılarak (0,1561 x 23,3690) 3,65 maliyet değeri elde edilir.

- “TURDA1” ve “TURDA2” hücreleri aynı baz istasyonundaki hücrelerdir. Bu sebeple 3,65 değerine, aynı baz istasyonunda ardışık kanal kullanımından dolayı 1000 maliyet değeri eklenerek 1003,65 frekans maliyet değeri elde edilir.

Çizelge 6.5 ’deki frekans maliyet değerleri, döngü içerisinde her frekans ataması ya da frekans değişimi sonrası tekrar hesaplanırlar. Her frekans ataması ya da değişimi öncesi bu tablo kullanılarak o hücre için en düşük maliyete sahip frekans belirlenir ve tek bir atama yapılır. Eğer hücrenin birden fazla taşıyıcı kanal ihtiyacı var ise atanan ilk frekansın ve bu frekansa ardışık olan frekansların aynı hücre içinde tekrar kullanımını engellemek için bu frekanslara geçici olarak 100000 maliyet değeri eklenir. Bu değer eklendikten sonra frekans maliyetleri bir sonraki atama için tekrar hesaplanır ve en düşük maliyete sahip ikinci frekans belirlenir. Hücreye gerekli olan tüm taşıyıcı kanallar atandıktan sonra bu frekanslara ve bu frekansların ardışık frekanslarına eklenen 100000 maliyet değeri

kaldırılır ve frekans maliyet değerleri toplanarak hücrenin maliyet değeri elde edilir. Örnek olarak “AKBAL1” hücresine 5, 7 ve 15 taşıyıcı kanalları atanmıştır. Bu hücrenin maliyet değeri 2,09 (0,55 + 0,55 + 0,99) ’dur. Bir hücrenin tüm frekansları atandıktan sonra önceliklendirilmiş hücre tablosunda sıradaki hücreye geçilir ve sıradaki hücre için de Çizelge 6.5 ’deki frekans maliyet değerleri hesaplanarak aynı işlemler gerçekleştirilir.

Oluşturulan sonuç planının toplam maliyeti Çizelge 6.6 ’da görülmektedir. Bu değer tüm hücre maliyetlerinin toplamına eşittir ve en iyi planın elde edilip edilmediğinin değerlendirilmesinde kullanılır. Yazılımın amacı, sonuç planının maliyetini en aza indirmektir.

Çizelge 6.6 - Oluşturulan sonuç planının toplam maliyeti

Hücre Adı	**Maliyet Değeri**
AHMTL1	0,20
AHMTL2	5,30
AKBAL1	2,09
TURDA1	2008,80
TURDA2	2014,18
TOTAL	**4030,57**

6.3.1 Frekans Atama Yazılımının Çalışma Sonuçları

Spektrumun darlığından ötürü limitli frekans sayısına sahip olunduğu için aynı frekanslar sıkça tekrar kullanılmak zorundadır. Sinyaller arası girişim problemini azaltmak için frekans planı yapılması gereklidir.

Girişim seviyesi, doğrudan konuşma kalitesini, kesintisiz konuşma süresini ve kesinti yaşanan konuşma sayısını etkiler. Limitli frekans spektrumunu verimli şekilde kullanmak için radyo şebekesindeki girişim seviyesi en aza indirilmelidir.

Geliştirilen yazılım, girişim seviyesini en aza indirmek için mobil ölçümleri kullanarak uygun bir frekans planı oluşturmaktadır. Oluşturulan plandaki frekanslar, abonelerin mobil cihazlarına servis veren ilgili hücrelere atanır. Dolayısıyla aboneler atanan frekanslardan, yani radyo şebekesinin kalitesinden doğrudan etkilenmektedirler. Bu yüzden şebekeden servis alan abonelerin mobil cihazları tarafından yapılan ölçümlere dayalı bir plan oluşturmak önemlidir.

Bu yazılım, Türkiye'deki GSM operatörlerinden birinde test edilmiştir. Test edilen iller ve frekans planları şunlardır:

- Ankara ili TCH planı
- Antalya ili BCCH planı
- Antalya ili TCH planı
- Gaziantep, Hatay, Kilis, Mardin, Şanlıurfa, Şırnak illeri BCCH planı
- Gaziantep, Hatay, Kilis, Mardin, Şanlıurfa, Şırnak illeri TCH planı
- İzmir ve Manisa illeri TCH planı
- Ağrı, Ardahan, Artvin, Bayburt, Erzincan, Erzurum, Giresun, Gümüşhane, Iğdır, Kars, Ordu, Rize, Trabzon illeri BCCH planı
- Ağrı, Ardahan, Artvin, Bayburt, Erzincan, Erzurum, Giresun, Gümüşhane, Iğdır, Kars, Ordu, Rize, Trabzon illeri TCH planı

Geliştirilen yazılımın test edilebilmesi için sonuçlarının kıyaslanabileceği bir ortama ihtiyaç vardır. Bu yüzden sonuçlar, tüm dünyada kullanılan AIRCOM International firmasının ASSET programı altındaki otomatik frekans planlama modülü olan ILSA ile kıyaslanmıştır. ILSA programı da maliyet hesabına dayalı bir programdır ve oluşturduğu frekans planının maliyet değerini en aza indirmeye çalışır. ILSA programının diğer bir özelliği ise, kendisine mevcut bir frekans planı girildiğinde bu planın maliyet değerini kullanıcıya bildirmektedir. Bu özellik ile geliştirilen yazılımın oluşturduğu frekans planları, ILSA programına girilip ILSA programındaki maliyet değerleri görülmüştür.

Aynı şartları sağlayabilmek için geliştirilen yazılım ile ILSA programına aynı veriler girilmiştir ve her iki program aynı sürelerde çalıştırılıp frekans planları elde edilmiştir. ILSA programının dezavantajı, her çalıştırıldığında aynı sürede farklı maliyet değerine sahip bir plan oluşturmasıdır. Bu sebeple ILSA programı, çalışma süresinin uzun olmasından dolayı her plan için 5'er kez çalıştırılabilmiştir. Oluşturulan planların maliyet değeri kıyaslaması sonuçları şunlardır;

- Ankara ili TCH Planı:

Kullanılan Program	**ILSA Programında Hesaplanan Maliyet Değeri**
Geliştirilen Yazılım	237.312.734
ILSA	237.872.490
ILSA	237.803.615
ILSA	238.129.314
ILSA	237.798.221
ILSA	237.885.382

- Antalya ili BCCH Planı:

Kullanılan Program	ILSA Programında Hesaplanan Maliyet Değeri
Geliştirilen Yazılım	905.329
ILSA	1.128.981
ILSA	1.342.256
ILSA	1.107.134
ILSA	1.148.332
ILSA	1.221.719

- Antalya ili TCH Planı:

Kullanılan Program	ILSA Programında Hesaplanan Maliyet Değeri
Geliştirilen Yazılım	190.297.223
ILSA	192.887.902
ILSA	193.239.155
ILSA	192.388.687
ILSA	192.582.430
ILSA	192.129.201

- Gaziantep, Hatay, Kilis, Mardin, Şanlıurfa, Şırnak illeri BCCH planı:

Kullanılan Program	ILSA Programında Hesaplanan Maliyet Değeri
Geliştirilen Yazılım	445.812.333
ILSA	506.302.769
ILSA	506.121.664
ILSA	507.243.982
ILSA	506.476.894
ILSA	506.843.756

- Gaziantep, Hatay, Kilis, Mardin, Şanlıurfa, Şırnak illeri TCH planı:

Kullanılan Program	ILSA Programında Hesaplanan Maliyet Değeri
Geliştirilen Yazılım	47.832.778.721
ILSA	48.256.667.483
ILSA	48.554.561.487
ILSA	48.387.688.322
ILSA	48.336.975.815
ILSA	48.410.056.729

- İzmir ve Manisa illeri TCH planı:

Kullanılan Program	ILSA Programında Hesaplanan Maliyet Değeri
Geliştirilen Yazılım	12.190.253.131
ILSA	12.328.471.111
ILSA	12.305.832.592
ILSA	12.289.335.241
ILSA	12.173.257.211
ILSA	12.131.861.230

- Ağrı, Ardahan, Artvin, Bayburt, Erzincan, Erzurum, Giresun, Gümüşhane, Iğdır, Kars, Ordu, Rize, Trabzon illeri BCCH planı:

Kullanılan Program	ILSA Programında Hesaplanan Maliyet Değeri
Geliştirilen Yazılım	2.350.421
ILSA	2.681.239
ILSA	2.720.334
ILSA	2.695.934
ILSA	2.879.238
ILSA	2.751.383

- Ağrı, Ardahan, Artvin, Bayburt, Erzincan, Erzurum, Giresun, Gümüşhane, Iğdır, Kars, Ordu, Rize, Trabzon illeri TCH planı:

Kullanılan Program	**ILSA Programında Hesaplanan Maliyet Değeri**
Geliştirilen Yazılım	307.990.286
ILSA	314.421.454
ILSA	315.300.291
ILSA	314.567.912
ILSA	314.455.510
ILSA	314.741.353

İzmir ve Manisa illeri TCH planı dışında diğer tüm planlarda geliştirilen yazılım ILSA programına göre daha iyi sonuç vermiştir. İzmir ve Manisa illeri TCH planında ise beş çalıştırmanın üçüne göre daha iyi sonuç vermiştir.

Geliştirilen yazılım, özellikle BCCH planlarında ILSA programına göre çok daha iyi sonuçlar vermiştir. TCH planlarında ise sonuçlar birbirine daha yakındır.

6.3.2 Frekans Atama Sonuçlarının Değerlendirilmesi

Geliştirilen yazılımın sonuçlarının ILSA ile kıyaslanmasından sonra ILSA programına göre daha iyi bir frekans planı elde edildiği görülmüştür. Elde edilen bu frekansların, plan yapılan bölgedeki hücrelere atanmasından sonra hücrelerin tüm kalite değerlerinde iyileşme sağlanmıştır. Bu kalite değerleri Şekil 6.2'de görülmektedir. Burada Integrity hücrelerin TCH frekanslarının ses kalitesini göstermektedir. Bad 6&7 Samples değerlerini içerir. Kalite değeri 6 ve 7 olduğu durumlarda konuşma gerçekleştirilemez. Call Setup Success Rate hücrelerin BCCH frekansı kalitesini gösterir. BCCH frekans kalitesi iyiyse hücreye erişim sağlanabilir. Retainability hücrelerin TCH frekanslarının kalitesini gösterir. Bad Quality Disconnection ve Drop değerlerini içerir. Bu değerler konuşma sırasında yaşanan kesintileri gösterirler. Route Rxqual değeri ise belirli yollardan geçilerek alınan sinyal kalitesi değerleridir.

	İYİLEŞME
INTEGRITY (TCH Kalitesi)	8,48%
CALL SETUP SUCCESS RATE (BCCH Kalitesi)	10,65%
RETAINABILITY (TCH Kalitesi)	4,14%
BAD QUALITY DISCONNECTION	4,71%
DROP	3,28%
BAD 6&7 SAMPLES (UPLINK RATE)	4,69%
BAD 6&7 SAMPLES (DOWNLINK RATE)	13,87%
ROUTE RXQUAL 0-7 (Yol ölçümleri)	0,14

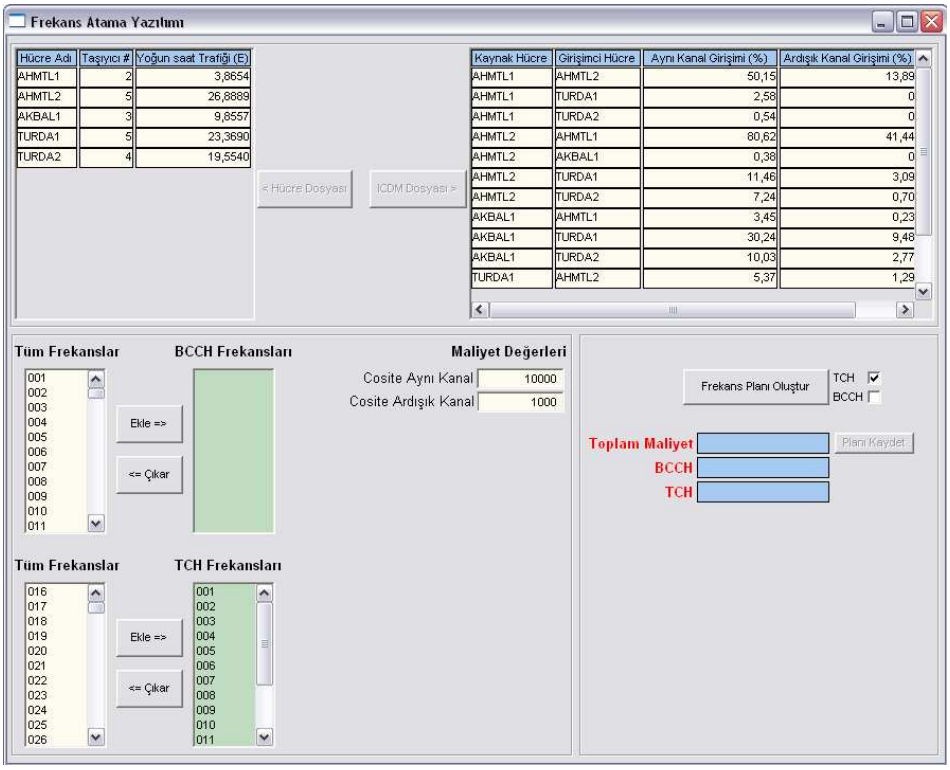
Şekil 6.2 - Hücrelerin kalite değerlerindeki iyileşme

6.3.3 Frekans Atama Yazılımının Kullanımı

Geliştirilen yazılım ile frekans atama işlemine başlamadan önce yapılması gerekenler şunlardır:

- Hücre tablosu (Çizelge 6.1) programa yerleştirilmelidir. Bunun için bu tablodaki kolon sırasına uygun olarak hazırlanmış tab ayıraçlı bir text dosya kullanılmalıdır. "< Hücre Dosyası" butonuna basılarak ekrana gelen pencereden bu dosyanın seçilmesi yeterlidir. Dosya seçildikten sonra içerdiği değerler ekrana gelecektir.
- ICDM tablosu (Çizelge 6.2) programa yerleştirilmelidir. Bunun için bu tablodaki kolon sırasına uygun olarak hazırlanmış tab ayıraçlı bir text dosya kullanılmalıdır.
- "ICDM Dosyası >" butonuna basılarak ekrana gelen pencereden bu dosyanın seçilmesi yeterlidir. Dosya seçildikten sonra içerdiği değerler ekrana gelecektir.
- Frekans atama işleminde kullanılacak BCCH frekansları "Ekle =>" butonu kullanılarak seçilmelidir.
- Frekans atama işleminde kullanılacak TCH frekansları "Ekle =>" butonu kullanılarak seçilmelidir.
- "Cosite Aynı Kanal" maliyet değeri 10000 olarak belirlenmiştir. İstenirse bu değer ekrandan değiştirilebilir.
- "Cosite Ardışık Kanal" maliyet değeri 1000 olarak belirlenmiştir. İstenirse bu değer ekrandan değiştirilebilir.
- TCH frekans planı yapılmak isteniyorsa TCH kutucuğu işaretlenmelidir.
- BCCH frekans planı yapılmak isteniyorsa BCCH kutucuğu işaretlenmelidir.
- Hem TCH hem BCCH frekans planı yapılmak isteniyorsa TCH ve BCCH kutucuğu işaretlenmelidir.

Bu işlemler yapıldıktan sonra frekans atama işlemine başlamak için “Frekans Planı Oluştur” butonuna basılır. Atama işlemi yapılırken o ana kadarki elde edilen en iyi maliyet değeri ekrandaki “Toplam Maliyet” alanında görülmektedir. Atama işlemi bittikten sonra “Planı Kaydet” butonu ile oluşturulan son frekans planı istenen dosya adıyla kaydedilir. Yazılımın ekran görüntüsü Şekil 6.3’te görülmektedir.



Şekil 6.3 - Frekans atama yazılımı ekran görüntüsü

Ek 1 Frekans Atama Yazılımının Kaynak Kodu

Frekans atama yazılımı, PowerBuilder (Powersoft PowerBuilder Enterprise by Sybase 6.5.1) ortamında geliştirilmiştir. Yazılımın PowerBuilder ortamında geliştirilmesinin sebebi, yazılımın test edildiği GSM operatöründe bu ortamın kullanılıyor olmasıdır.

Kaynak kodu:

```
$PBExportHeader$w_afp.srw

forward
global type w_afp from window_common
end type
type d_afp_cells from datawindow_common within w_afp
end type
type cb_import_cells from commandbutton within w_afp
end type
type d_afp_icdm from datawindow_common within w_afp
end type
type cb_import_icdm from commandbutton within w_afp
end type
type cb_plan from commandbutton within w_afp
end type
type lb_all_frequencies from listbox within w_afp
end type
type st_all_freq from statictext within w_afp
end type
type lb_bcch_frequencies from listbox within w_afp
end type
type cb_add_bcch from commandbutton within w_afp
end type
type st_bcch_freq from statictext within w_afp
end type
type cb_remove_bcch from commandbutton within w_afp
end type
type st_all_freq_3 from statictext within w_afp
```

end type
type lb_all_frequencies_3 from listbox within w_afp
end type
type st_tch_freq from statictext within w_afp
end type
type lb_tch_frequencies from listbox within w_afp
end type
type cb_add_tch from commandbutton within w_afp
end type
type cb_remove_tch from commandbutton within w_afp
end type
type d_afp_cell_frequencies from datawindow_common within w_afp
end type
type st_initial_cost from statictext within w_afp
end type
type st_initial_cost_label from statictext within w_afp
end type
type cb_optimize from commandbutton within w_afp
end type
type cb_save_initial from commandbutton within w_afp
end type
type st_best_cost_label from statictext within w_afp
end type
type st_best_cost from statictext within w_afp
end type
type d_afp_cell_frequencies_best from datawindow_common within
w_afp
end type
type sle_optimize_times from singlelineedit within w_afp
end type
type st_times from statictext within w_afp
end type
type st_status from statictext within w_afp
end type
type st_status_label from statictext within w_afp
end type
type d_afp_initial_plan from datawindow_common within w_afp
end type
type cb_import_initial_plan from commandbutton within w_afp

end type
type cb_clear_initial_plan from commandbutton within w_afp
end type
type d_afp_icdm_interferer from datawindow_common within w_afp
end type
type cb_check_plan from commandbutton within w_afp
end type
type dr_frequency from datawindow_main within w_afp
end type
type lb_cell_name from listbox within w_afp
end type
type lb_external_name from listbox within w_afp
end type
type cbx_bcch_plan from checkbox within w_afp
end type
type cbx_tch_plan from checkbox within w_afp
end type
type st_best_bcch_cost from statictext within w_afp
end type
type st_best_tch_cost from statictext within w_afp
end type
type st_best_bcch_cost_label from statictext within w_afp
end type
type st_best_tch_cost_label from statictext within w_afp
end type
type gb_3 from groupbox within w_afp
end type
type gb_1 from groupbox within w_afp
end type
type cb_save_best from commandbutton within w_afp
end type
type gb_2 from groupbox within w_afp
end type
type st_initial_bcch_cost_label from statictext within w_afp
end type
type st_initial_tch_cost_label from statictext within w_afp
end type
type st_initial_bcch_cost from statictext within w_afp
end type

```
type st_initial_tch_cost from statictext within w_afp
end type
type cb_import_icdm_msmt from commandbutton within w_afp
end type
type sle_same_ch_cost from singlelineedit within w_afp
end type
type sle_adj_ch_cost from singlelineedit within w_afp
end type
type sle_forbid_ch_cost from singlelineedit within w_afp
end type
type st_1 from statictext within w_afp
end type
type st_2 from statictext within w_afp
end type
type st_3 from statictext within w_afp
end type
type st_4 from statictext within w_afp
end type
end forward

type struct_temp_cell_freq_bcch from structure
        long            temp_cell_freq_bcch[]
        long            temp_best_cell_freq_bcch[]
        long            temp_freq_no[124]
        decimal {0}     temp_freq_cost[124]
end type

global type w_afp from window_common
int X=5
int Y=4
int Width=4681
int Height=3052
boolean TitleBar=true
string Title="AFP"
long BackColor=82042848
d_afp_cells d_afp_cells
cb_import_cells cb_import_cells
d_afp_icdm d_afp_icdm
cb_import_icdm cb_import_icdm
```

cb_plan cb_plan
lb_all_frequencies lb_all_frequencies
st_all_freq st_all_freq
lb_bcch_frequencies lb_bcch_frequencies
cb_add_bcch cb_add_bcch
st_bcch_freq st_bcch_freq
cb_remove_bcch cb_remove_bcch
st_all_freq_3 st_all_freq_3
lb_all_frequencies_3 lb_all_frequencies_3
st_tch_freq st_tch_freq
lb_tch_frequencies lb_tch_frequencies
cb_add_tch cb_add_tch
cb_remove_tch cb_remove_tch
d_afp_cell_frequencies d_afp_cell_frequencies
st_initial_cost st_initial_cost
st_initial_cost_label st_initial_cost_label
cb_optimize cb_optimize
cb_save_initial cb_save_initial
st_best_cost_label st_best_cost_label
st_best_cost st_best_cost
d_afp_cell_frequencies_best d_afp_cell_frequencies_best
sle_optimize_times sle_optimize_times
st_times st_times
st_status st_status
st_status_label st_status_label
d_afp_initial_plan d_afp_initial_plan
cb_import_initial_plan cb_import_initial_plan
cb_clear_initial_plan cb_clear_initial_plan
d_afp_icdm_interferer d_afp_icdm_interferer
cb_check_plan cb_check_plan
dr_frequency dr_frequency
lb_cell_name lb_cell_name
lb_external_name lb_external_name
cbx_bcch_plan cbx_bcch_plan
cbx_tch_plan cbx_tch_plan
st_best_bcch_cost st_best_bcch_cost
st_best_tch_cost st_best_tch_cost
st_best_bcch_cost_label st_best_bcch_cost_label
st_best_tch_cost_label st_best_tch_cost_label

gb_3 gb_3
gb_1 gb_1
cb_save_best cb_save_best
gb_2 gb_2
st_initial_bcch_cost_label st_initial_bcch_cost_label
st_initial_tch_cost_label st_initial_tch_cost_label
st_initial_bcch_cost st_initial_bcch_cost
st_initial_tch_cost st_initial_tch_cost
cb_import_icdm_msmt cb_import_icdm_msmt
sle_same_ch_cost sle_same_ch_cost
sle_adj_ch_cost sle_adj_ch_cost
sle_forbid_ch_cost sle_forbid_ch_cost
st_1 st_1
st_2 st_2
st_3 st_3
st_4 st_4
end type
global w_afp w_afp

type variables
Boolean bcch_frequency_list[124]
Boolean tch_frequency_list[124]
long cell_count
long cell_tch_number[]
decimal cell_traffic[]
decimal icdm_co[4000,500]
decimal icdm_co_int[4000,500]
decimal icdm_adj[4000,500]
decimal icdm_adj_int[4000,500]
long icdm_interferer_ind[4000,500]
long icdm_interfered_ind[4000,500]
long cell_interferer_count[]
long cell_interfered_count[]
long cell_freq_bcch[]
long cell_freq_tch[4000,16]
long best_cell_freq_bcch[]
long last_best_cell_freq_bcch[]
long best_cell_freq_tch[4000,16]
long last_best_cell_freq_tch[4000,16]

```
long freq_no[124]
decimal freq_cost[124]
long random_bcch_list[]
long random_tch_list[]
long random_bcch_count
long random_tch_count
decimal freq_cost_table[4000,124]
decimal temp_freq_cost_table[4000,124]
decimal last_freq_cost_table[4000,124]
long forbidden_bcch_list[4000,124]
long forbidden_tch_list[4000,124]
long forbidden_bcch_count[4000]
long forbidden_tch_count[4000]
long fixed_bcch_list[4000]
long fixed_tch_list[4000,124]
long fixed_tch_count[4000]
boolean flag_cosite_cost_added[4000,4]
boolean flag_check_frequency[4000]
long log_file
long same_ch_cost,adj_ch_cost,forbid_ch_cost
end variables

forward prototypes
public subroutine f_clear_frequency_cost ()
public function decimal f_calculate_cost ()
public subroutine f_copy_cell_freq_to_best ()
public subroutine f_copy_initial_to_cell_freq ()
public subroutine f_calculate_nbr_count ()
public subroutine f_copy_best_to_cell_freq ()
public subroutine f_set_random_bcch ()
public function long f_get_best_bcch (long cell_id)
public subroutine f_clear_freq_cost_table ()
public subroutine f_set_random_tch ()
public function long f_get_best_tch (long cell_id)
public subroutine f_set_forbidden_and_fixed_freq ()
public subroutine f_clear_forbidden_and_fixed_list ()
public subroutine f_clear_cell_freqs ()
public subroutine f_set_arrays_from_dw ()
public subroutine f_add_cost_to_forbidden_fixed ()
```

```
public subroutine f_clean_cosite_costs ()
public subroutine f_check_new_plan ()
public subroutine f_check_initial_plan ()
public subroutine f_add_cost_in_cosite ()
public subroutine f_make_best_plan_dw ()
public subroutine f_change_bcch_frequency (long cell_order, long
new_frequency)
public subroutine f_change_tch_frequency (long cell_order, long
tch_order, integer new_frequency)
public function decimal f_calculate_cost_new ()
end prototypes

public subroutine f_clear_frequency_cost ();long i
For i = 1 to 124
        freq_no[i] = i
        if bcch_frequency_list[i] = true then
          freq_cost[i] = 0
        elseif tch_frequency_list[i] = true then
          freq_cost[i] = 0
        else
          freq_cost[i] = 4000000000
        end if
Next
end subroutine

public function decimal f_calculate_cost ();long i,rc,o,k,j,m
decimal co_affect,adj_affect,total_cost,result_cost
long tch_count,bcch,tch,tru_number
long int_cll_ind

total_cost = 0

For i = 1 to cell_count
        f_clear_frequency_cost()

        //Frekans Cost Tablosu hazirlama
        For o = 1 to cell_interferer_count[i]

                int_cll_ind = icdm_interferer_ind[i,o]
```

```
                co_affect = icdm_co[i,o]
                adj_affect = icdm_adj[i,o]
                tch_count = cell_tch_number[int_cll_ind]

                bcch = cell_freq_bcch[int_cll_ind]
                if bcch > 0 then
                        freq_cost[bcch] += (co_affect * 10)
                        if bcch > 1 then freq_cost[bcch - 1] += (adj_affect
* 10)
                        if bcch < 124 then freq_cost[bcch + 1] +=
(adj_affect * 10)
                end if

                For k = 1 to tch_count
                        tch = cell_freq_tch[int_cll_ind,k]
                        if tch > 0 then
                                freq_cost[tch] += co_affect
                                if tch > 1 then
                                        freq_cost[tch - 1] += adj_affect
                                end if
                                if tch < 124 then
                                        freq_cost[tch + 1] += adj_affect
                                end if
                        end if
                Next

        Next


        if cbx_bcch_plan.checked then
          bcch = cell_freq_bcch[i]
          For m = 1 to forbidden_bcch_count[i]
                if bcch = forbidden_bcch_list[i,m] then
                        freq_cost[bcch] += forbid_ch_cost
                        exit
                end if
          Next
        end if
```

```
if cbx_tch_plan.checked then
  For j = 1 to cell_tch_number[i]
        tch = cell_freq_tch[i,j]
        if tch > 0 then

              if cbx_bcch_plan.checked then
                    if Abs(bcch - tch) <= 1 then
                          freq_cost[tch] += 100000
                          freq_cost[bcch] += 100000
                    end if
              end if

              For k = (j+1) to cell_tch_number[i]
                    if Abs(cell_freq_tch[i,k] - tch) <= 1 then
                          freq_cost[tch] += 100000
                          freq_cost[cell_freq_tch[i,k]] +=
100000
                    end if
              Next
              For m = 1 to forbidden_tch_count[i]
                    if tch = forbidden_tch_list[i,m] then
                          freq_cost[tch] += forbid_ch_cost
                          exit
                    end if
              Next

        end if
  Next
end if

if cbx_bcch_plan.checked then
  if cell_freq_bcch[i] > 0 then
        result_cost = freq_cost[cell_freq_bcch[i]]
  else
        result_cost = 0
  end if
else
  result_cost = 0
end if
```

```
        if cbx_tch_plan.checked then
          For j = 1 to cell_tch_number[i]
                  if cell_freq_tch[i,j] > 0 then result_cost +=
freq_cost[cell_freq_tch[i,j]]
          Next
        end if

        total_cost += result_cost
Next

return total_cost
end function

public subroutine f_copy_cell_freq_to_best ();long i,j,rc

d_afp_cell_frequencies_best.reset()
rc = d_afp_cell_frequencies.rowcount()

For i = 1 to rc
        d_afp_cell_frequencies_best.insertrow(0)
        d_afp_cell_frequencies_best.setitem(i,"cell_name",d_afp_cell_freq
uencies.getitemstring(i,"cell_name"))
        d_afp_cell_frequencies_best.setitem(i,"bcch",d_afp_cell_frequenci
es.getitemnumber(i,"bcch"))
        d_afp_cell_frequencies_best.setitem(i,"tru_number",d_afp_cell_fre
quencies.getitemnumber(i,"tru_number"))
        For j = 1 to 16

        d_afp_cell_frequencies_best.setitem(i,("tch"+string(j)),d_afp_cell_
frequencies.getitemnumber(i,("tch"+string(j))))
        Next
        d_afp_cell_frequencies_best.setitem(i,"bcch_cost",d_afp_cell_freq
uencies.getitemdecimal(i,"bcch_cost"))
        d_afp_cell_frequencies_best.setitem(i,"tch_cost",d_afp_cell_freque
ncies.getitemdecimal(i,"tch_cost"))
        d_afp_cell_frequencies_best.setitem(i,"total_cost",d_afp_cell_freq
uencies.getitemdecimal(i,"total_cost"))
Next
```

```
end subroutine

public subroutine f_copy_initial_to_cell_freq ();long i,j,o,last_row,rc,m
decimal co_affect,adj_affect
long bcch,tch,tru_number,frequency_no
string src_cll, int_cll, tch_string
long src_cll_ind, int_cll_ind, ext_cll_ind, cell_no
decimal src_trf
string forbidden_bcch_string,forbidden_tch_string,fixed_tch_string

lb_cell_name.reset()
lb_external_name.reset()

cell_count = d_afp_cells.rowcount()
last_row = d_afp_icdm.rowcount()

d_afp_icdm.SetRedraw(false)
d_afp_cells.SetRedraw(false)

d_afp_cells.SetSort("#9 D, #4 D")
d_afp_cells.Sort()

For i = 1 to cell_count
        lb_cell_name.InsertItem(d_afp_cells.getitemstring(i,"cell_name"),i)
        cell_tch_number[i] =
d_afp_cells.getitemnumber(i,"total_tch_count")
        cell_traffic[i] = d_afp_cells.getitemdecimal(i,"tot_traf")
        cell_interferer_count[i] = 0
        cell_interfered_count[i] = 0
Next

f_clear_freq_cost_table()
f_clear_forbidden_and_fixed_list()

For i = 1 to 4000
        cell_freq_bcch[i] = 0
        For o = 1 to 16
          cell_freq_tch[i,o] = 0
        Next
```

```
Next

For i = 1 to 4000
        For cell_no = 1 to 4
          flag_cosite_cost_added[i,cell_no] = false
        Next
Next

f_set_forbidden_and_fixed_freq()

ext_cll_ind = 3000

For i = 1 to last_row

        src_cll = d_afp_icdm.getitemstring(i,"source_cell")
        src_cll_ind = lb_cell_name.FindItem(src_cll,0)
        if src_cll_ind < 0 then
          continue
        end if

        int_cll = d_afp_icdm.getitemstring(i,"interferer_cell")

        src_trf = cell_traffic[src_cll_ind]

        if left(int_cll,5) = left(src_cll,5) then // cosite ta costu yukselt
          co_affect = same_ch_cost
          adj_affect = adj_ch_cost
          cell_no = long(mid(int_cll,6,1))
          flag_cosite_cost_added[src_cll_ind,cell_no] = true
        else
          co_affect = 0
          adj_affect = 0
        end if


        int_cll_ind = lb_cell_name.FindItem(int_cll,0)
        if int_cll_ind < 0 then //external cell

          int_cll_ind = lb_external_name.FindItem(int_cll,0)
```

```
        if int_cll_ind < 0 then // new external

                bcch = f_get_bcch_from_cell(int_cll)

                if (bcch > 0) then

                        ext_cll_ind++
                        lb_external_name.InsertItem(int_cll,ext_cll_ind -
3000)
                        int_cll_ind = ext_cll_ind

                        cell_freq_bcch[int_cll_ind] = bcch
                        freq_cost_table[src_cll_ind,bcch] +=
(((d_afp_icdm.getitemdecimal(i,"traf_affected_co_percent") * src_trf /
100) + co_affect) * 10)
                        if bcch > 1 then freq_cost_table[src_cll_ind,bcch -
1] += (((d_afp_icdm.getitemdecimal(i,"traf_affected_adj_percent") *
src_trf / 100) + adj_affect) * 10)
                        if bcch < 124 then freq_cost_table[src_cll_ind,bcch
+ 1] += (((d_afp_icdm.getitemdecimal(i,"traf_affected_adj_percent") *
src_trf / 100) + adj_affect) * 10)



                        tru_number = 0
                        tch_string = f_get_all_tch_from_cell(int_cll)
                        if len(tch_string) > 0 then
                                tru_number = f_get_from_count(tch_string)
                                For o = 1 to tru_number
                                        tch = long(f_get_from(tch_string,o))
                                        cell_freq_tch[int_cll_ind,o] = tch

                                        freq_cost_table[src_cll_ind,tch] +=
(d_afp_icdm.getitemdecimal(i,"traf_affected_co_percent") * src_trf /
100) + co_affect
                                        if tch > 1 then
freq_cost_table[src_cll_ind,tch - 1] +=
(d_afp_icdm.getitemdecimal(i,"traf_affected_adj_percent") * src_trf /
100) + adj_affect
```

```
                                        if tch < 124 then
freq_cost_table[src_cll_ind,tch + 1] +=
(d_afp_icdm.getitemdecimal(i,"traf_affected_adj_percent") * src_trf /
100) + adj_affect

                                    Next
                            end if

                            cell_tch_number[int_cll_ind] = tru_number

                    else

                            continue
                    end if
            else // current external

                    int_cll_ind += 3000

                    bcch = cell_freq_bcch[int_cll_ind]
                    freq_cost_table[src_cll_ind,bcch] +=
(((d_afp_icdm.getitemdecimal(i,"traf_affected_co_percent") * src_trf /
100) + co_affect) * 10)
                    if bcch > 1 then freq_cost_table[src_cll_ind,bcch - 1] +=
(((d_afp_icdm.getitemdecimal(i,"traf_affected_adj_percent") * src_trf /
100) + adj_affect) * 10)
                    if bcch < 124 then freq_cost_table[src_cll_ind,bcch + 1]
+= (((d_afp_icdm.getitemdecimal(i,"traf_affected_adj_percent") * src_trf
/ 100) + adj_affect) * 10)


                    tru_number = cell_tch_number[int_cll_ind]

                    For o = 1 to tru_number
                            tch = cell_freq_tch[int_cll_ind,o]

                            freq_cost_table[src_cll_ind,tch] +=
(d_afp_icdm.getitemdecimal(i,"traf_affected_co_percent") * src_trf /
100) + co_affect
```

```
                        if tch > 1 then freq_cost_table[src_cll_ind,tch - 1]
+= (d_afp_icdm.getitemdecimal(i,"traf_affected_adj_percent") * src_trf /
100) + adj_affect
                        if tch < 124 then freq_cost_table[src_cll_ind,tch +
1] += (d_afp_icdm.getitemdecimal(i,"traf_affected_adj_percent") *
src_trf / 100) + adj_affect

                    Next

          end if
        end if

        cell_interfered_count[int_cll_ind]++
        icdm_interfered_ind[int_cll_ind,cell_interfered_count[int_cll_ind]]
= src_cll_ind
        icdm_co_int[int_cll_ind,cell_interfered_count[int_cll_ind]] =
(d_afp_icdm.getitemdecimal(i,"traf_affected_co_percent") * src_trf /
100) + co_affect
        icdm_adj_int[int_cll_ind,cell_interfered_count[int_cll_ind]] =
(d_afp_icdm.getitemdecimal(i,"traf_affected_adj_percent") * src_trf /
100) + adj_affect

        cell_interferer_count[src_cll_ind]++
        icdm_interferer_ind[src_cll_ind,cell_interferer_count[src_cll_ind]]
= int_cll_ind
        icdm_co[src_cll_ind,cell_interferer_count[src_cll_ind]] =
(d_afp_icdm.getitemdecimal(i,"traf_affected_co_percent") * src_trf /
100) + co_affect
        icdm_adj[src_cll_ind,cell_interferer_count[src_cll_ind]] =
(d_afp_icdm.getitemdecimal(i,"traf_affected_adj_percent") * src_trf /
100) + adj_affect
Next

//Cosite icinde icdm olcum yapmadiysa eksik costlari yukseltme
f_add_cost_in_cosite()

//Forbidden frekanslara cost eklenmesi
For i = 1 to cell_count
```

```
        //Forbidden

        For o = 1 to forbidden_bcch_count[i]
          freq_cost_table[i,forbidden_bcch_list[i,o]] += forbid_ch_cost
        Next

        For o = 1 to forbidden_tch_count[i]
          freq_cost_table[i,forbidden_tch_list[i,o]] += forbid_ch_cost
        Next

Next


rc = d_afp_initial_plan.rowcount()

For i = 1 to rc

        src_cll = d_afp_initial_plan.getitemstring(i,"cell_name")
        src_cll_ind = lb_cell_name.FindItem(src_cll,0)
        if src_cll_ind < 0 then
          continue
        end if

        cell_freq_bcch[src_cll_ind] =
d_afp_initial_plan.getitemnumber(i,"bcch")
        if isnull(cell_freq_bcch[src_cll_ind]) then
cell_freq_bcch[src_cll_ind] = 0
        For j = 1 to cell_tch_number[src_cll_ind]
          cell_freq_tch[src_cll_ind,j] =
d_afp_initial_plan.getitemnumber(i,("tch"+string(j)))
        Next
Next



if cbx_bcch_plan.checked then
        For i = 1 to cell_count

          frequency_no = cell_freq_bcch[i]
```

```
        For o = 1 to cell_interfered_count[i]

                src_cll_ind = icdm_interfered_ind[i,o]

                co_affect = icdm_co_int[i,o]
                adj_affect = icdm_adj_int[i,o]

                freq_cost_table[src_cll_ind,frequency_no] += (co_affect *
10)
                if frequency_no > 1 then
freq_cost_table[src_cll_ind,frequency_no - 1] += (adj_affect * 10)
                if frequency_no < 124 then
freq_cost_table[src_cll_ind,frequency_no + 1] += (adj_affect * 10)

        Next

        freq_cost_table[i,frequency_no] += 100000
        if frequency_no > 1 then freq_cost_table[i,frequency_no - 1] +=
100000
        if frequency_no < 124 then freq_cost_table[i,frequency_no + 1]
+= 100000

      Next // end i
end if


if cbx_tch_plan.checked then
      For i = 1 to cell_count

        For m = 1 to cell_tch_number[i]

                frequency_no = cell_freq_tch[i,m]

                For o = 1 to cell_interfered_count[i]

                        src_cll_ind = icdm_interfered_ind[i,o]

                        co_affect = icdm_co_int[i,o]
```

```
                        adj_affect = icdm_adj_int[i,o]

                        freq_cost_table[src_cll_ind,frequency_no] +=
(co_affect)
                        if frequency_no > 1 then
freq_cost_table[src_cll_ind,frequency_no - 1] += (adj_affect)
                        if frequency_no < 124 then
freq_cost_table[src_cll_ind,frequency_no + 1] += (adj_affect)

                Next

                freq_cost_table[i,frequency_no] += 100000
                if frequency_no > 1 then freq_cost_table[i,frequency_no -
1] += 100000
                if frequency_no < 124 then freq_cost_table[i,frequency_no
+ 1] += 100000


          Next // end m
        Next // end i
end if

d_afp_icdm.SetRedraw(true)
d_afp_cells.SetRedraw(true)
end subroutine

public subroutine f_calculate_nbr_count ();long
i,j,rc,found_row,last_row,found_row_2,nbr_tru_count
string find_string,source_cell,interferer_cell

rc = d_afp_cells.rowcount()
last_row = d_afp_icdm.rowcount()

//Nbr Count hesaplama
For i = 1 to rc
        find_string = "source_cell =
'"+d_afp_cells.getitemstring(i,"cell_name")+"'"
        found_row = d_afp_icdm.Find(find_string,1,last_row)
```

```
        if found_row > 0 then
          source_cell = d_afp_icdm.getitemstring(found_row,"source_cell")


        //Total Nbr TRU count hesaplama
          interferer_cell =
d_afp_icdm.getitemstring(found_row,"interferer_cell")
          find_string = "cell_name = '"+interferer_cell+"'"
          found_row_2 = d_afp_cells.Find(find_string,1,rc)
          if found_row_2 > 0 then nbr_tru_count =
d_afp_cells.getitemnumber(found_row_2,"total_tch_count") + 1
        //--------

          j = found_row + 1
          if found_row <> last_row then
                  Do while source_cell =
d_afp_icdm.getitemstring(j,"source_cell")

                          //Total Nbr TRU count hesaplama
                          interferer_cell =
d_afp_icdm.getitemstring(j,"interferer_cell")
                          find_string = "cell_name = '"+interferer_cell+"'"
                          found_row_2 = d_afp_cells.Find(find_string,1,rc)
                          if found_row_2 > 0 then nbr_tru_count =
nbr_tru_count +
d_afp_cells.getitemnumber(found_row_2,"total_tch_count") + 1
                          //----------

                          j = j + 1
                          if j > last_row then exit
                  Loop
          end if
          d_afp_cells.setitem(i,"nbr_count",(j - found_row))
          d_afp_cells.setitem(i,"nbr_tch_tru_count",(nbr_tru_count - (j -
found_row)))
        else
          d_afp_cells.setitem(i,"nbr_count",0)
          d_afp_cells.setitem(i,"nbr_tch_tru_count",0)
        end if
```

```
Next
//-------
end subroutine

public subroutine f_copy_best_to_cell_freq ();long i,j,rc

d_afp_cell_frequencies.reset()
rc = d_afp_cell_frequencies_best.rowcount()

For i = 1 to rc
	d_afp_cell_frequencies.insertrow(0)
	d_afp_cell_frequencies.setitem(i,"cell_name",d_afp_cell_frequenci
es_best.getitemstring(i,"cell_name"))
	d_afp_cell_frequencies.setitem(i,"bcch",d_afp_cell_frequencies_be
st.getitemnumber(i,"bcch"))
	d_afp_cell_frequencies.setitem(i,"tru_number",d_afp_cell_frequen
cies_best.getitemnumber(i,"tru_number"))
	For j = 1 to 16

	d_afp_cell_frequencies.setitem(i,("tch"+string(j)),d_afp_cell_frequ
encies_best.getitemnumber(i,("tch"+string(j))))
	Next
	d_afp_cell_frequencies.setitem(i,"bcch_cost",d_afp_cell_frequenci
es_best.getitemdecimal(i,"bcch_cost"))
	d_afp_cell_frequencies.setitem(i,"tch_cost",d_afp_cell_frequencies
_best.getitemdecimal(i,"tch_cost"))
	d_afp_cell_frequencies.setitem(i,"total_cost",d_afp_cell_frequenci
es_best.getitemdecimal(i,"total_cost"))
Next
end subroutine

public subroutine f_set_random_bcch ();long i

random_bcch_count = 0

For i = 1 to 124
	if bcch_frequency_list[i] = true then
	  random_bcch_count++
```

```
        random_bcch_list[random_bcch_count] = i
      end if
Next
end subroutine

public function long f_get_best_bcch (long cell_id);long frequency_no,i
decimal temp_cost

frequency_no = 0
temp_cost = 4000000000

For i = 1 to random_bcch_count
      if freq_cost_table[cell_id,random_bcch_list[i]] = 0 then
        frequency_no = random_bcch_list[i]
        exit
      end if
      if freq_cost_table[cell_id,random_bcch_list[i]] < temp_cost then
        temp_cost = freq_cost_table[cell_id,random_bcch_list[i]]
        frequency_no = random_bcch_list[i]
      end if
Next

return frequency_no
end function

public subroutine f_clear_freq_cost_table ();long i,j

For i = 1 to cell_count
      For j = 1 to 124
        freq_cost_table[i,j] = 0
        temp_freq_cost_table[i,j] = 0
      Next
Next
end subroutine

public subroutine f_set_random_tch ();long i

random_tch_count = 0
For i = 1 to 124
```

```
        if tch_frequency_list[i] = true then
          random_tch_count++
          random_tch_list[random_tch_count] = i
        end if
Next
end subroutine
public function long f_get_best_tch (long cell_id);long frequency_no,i
decimal temp_cost

frequency_no = 0
temp_cost = 4000000000

For i = 1 to random_tch_count
        if freq_cost_table[cell_id,random_tch_list[i]] = 0 then
          frequency_no = random_tch_list[i]
          exit
        end if
        if freq_cost_table[cell_id,random_tch_list[i]] < temp_cost then
          temp_cost = freq_cost_table[cell_id,random_tch_list[i]]
          frequency_no = random_tch_list[i]
        end if
Next

return frequency_no
end function

public subroutine f_set_forbidden_and_fixed_freq ();string
forbidden_bcch_string,forbidden_tch_string,fixed_tch_string
long fixed_bcch
long i,o

For i = 1 to cell_count

        // Forbidden BCCH lerin eklenmesi
        forbidden_bcch_string =
d_afp_cells.getitemstring(i,"forbidden_bcch_list")
        if len(forbidden_bcch_string) > 0 then
          forbidden_bcch_count[i] =
f_get_from_count(forbidden_bcch_string)
```

```
        For o = 1 to forbidden_bcch_count[i]
                forbidden_bcch_list[i,o] =
long(f_get_from(forbidden_bcch_string,o))
        Next
      end if

      // Forbidden TCH lerin eklenmesi
      forbidden_tch_string =
d_afp_cells.getitemstring(i,"forbidden_tch_list")
      if len(forbidden_tch_string) > 0 then
        forbidden_tch_count[i] = f_get_from_count(forbidden_tch_string)
        For o = 1 to forbidden_tch_count[i]
                forbidden_tch_list[i,o] =
long(f_get_from(forbidden_tch_string,o))
        Next
      end if

      // Fixed BCCH in eklenmesi
      fixed_bcch = d_afp_cells.getitemnumber(i,"fixed_bcch")
      if isnull(fixed_bcch) then fixed_bcch = 0
      if fixed_bcch > 0 then
        fixed_bcch_list[i] = fixed_bcch
      end if

      // Fixed TCH lerin eklenmesi
      fixed_tch_string = d_afp_cells.getitemstring(i,"fixed_tch_list")
      if len(fixed_tch_string) > 0 then
        fixed_tch_count[i] = f_get_from_count(fixed_tch_string)
        For o = 1 to fixed_tch_count[i]
                fixed_tch_list[i,o] = long(f_get_from(fixed_tch_string,o))
        Next
      end if

Next
end subroutine

public subroutine f_clear_forbidden_and_fixed_list ();long i,j

For i = 1 to 4000
```

```
        forbidden_bcch_count[i] = 0
        forbidden_tch_count[i] = 0
        fixed_bcch_list[i] = 0
        fixed_tch_count[i] = 0

        For j = 1 to 124
          forbidden_bcch_list[i,j] = 0
          forbidden_tch_list[i,j] = 0
          fixed_tch_list[i,j] = 0
        Next

Next
end subroutine

public subroutine f_clear_cell_freqs ();long i,o

For i = 1 to 4000
        cell_freq_bcch[i] = 0
        For o = 1 to 16
          cell_freq_tch[i,o] = 0
        Next
Next
end subroutine

public subroutine f_set_arrays_from_dw ();long i

For i = 1 to cell_count
        lb_cell_name.InsertItem(d_afp_cells.getitemstring(i,"cell_name"),i)
        cell_tch_number[i] =
d_afp_cells.getitemnumber(i,"total_tch_count")
        cell_traffic[i] = d_afp_cells.getitemdecimal(i,"tot_traf")
        cell_interferer_count[i] = 0
        cell_interfered_count[i] = 0
Next
end subroutine

public subroutine f_add_cost_to_forbidden_fixed ();long i,o,m
long frequency_no,src_cll_ind
```

```
decimal co_affect,adj_affect

For i = 1 to cell_count

        //Forbidden
        For o = 1 to forbidden_bcch_count[i]
          freq_cost_table[i,forbidden_bcch_list[i,o]] += forbid_ch_cost
        Next

        For o = 1 to forbidden_tch_count[i]
          freq_cost_table[i,forbidden_tch_list[i,o]] += forbid_ch_cost
        Next


        //Fixed BCCH
        if fixed_bcch_list[i] > 0 then

          frequency_no = fixed_bcch_list[i]

          For o = 1 to cell_interfered_count[i]

                    src_cll_ind = icdm_interfered_ind[i,o]

                    co_affect = icdm_co_int[i,o]
                    adj_affect = icdm_adj_int[i,o]

                    freq_cost_table[src_cll_ind,frequency_no] += (co_affect *
10)
                    if frequency_no > 1 then
freq_cost_table[src_cll_ind,frequency_no - 1] += (adj_affect * 10)
                    if frequency_no < 124 then
freq_cost_table[src_cll_ind,frequency_no + 1] += (adj_affect * 10)

          Next

          cell_freq_bcch[i] = frequency_no

          freq_cost_table[i,frequency_no] += 100000
```

```
        if frequency_no > 1 then freq_cost_table[i,frequency_no - 1] +=
100000
        if frequency_no < 124 then freq_cost_table[i,frequency_no + 1]
+= 100000

      end if

      //Fixed TCH
      For m = 1 to fixed_tch_count[i]

        frequency_no = fixed_tch_list[i,m]

        For o = 1 to cell_interfered_count[i]

                src_cll_ind = icdm_interfered_ind[i,o]

                co_affect = icdm_co_int[i,o]
                adj_affect = icdm_adj_int[i,o]

                freq_cost_table[src_cll_ind,frequency_no] += (co_affect)
                if frequency_no > 1 then
freq_cost_table[src_cll_ind,frequency_no - 1] += (adj_affect)
                if frequency_no < 124 then
freq_cost_table[src_cll_ind,frequency_no + 1] += (adj_affect)

        Next

        cell_freq_tch[i,m] = frequency_no

        freq_cost_table[i,frequency_no] += 100000
        if frequency_no > 1 then freq_cost_table[i,frequency_no - 1] +=
100000
        if frequency_no < 124 then freq_cost_table[i,frequency_no + 1]
+= 100000

      Next

Next
end subroutine
```

```
public subroutine f_clean_cosite_costs ();long i,k

For i = 1 to cell_count
        if cbx_bcch_plan.checked then
          freq_cost_table[i,cell_freq_bcch[i]] -= 100000
          if cell_freq_bcch[i] > 1 then freq_cost_table[i,cell_freq_bcch[i] -
1] -= 100000
          if cell_freq_bcch[i] < 124 then freq_cost_table[i,cell_freq_bcch[i]
+ 1] -= 100000
        else
          if fixed_bcch_list[i] > 0 then
                    freq_cost_table[i,cell_freq_bcch[i]] -= 100000
                    if cell_freq_bcch[i] > 1 then
freq_cost_table[i,cell_freq_bcch[i] - 1] -= 100000
                    if cell_freq_bcch[i] < 124 then
freq_cost_table[i,cell_freq_bcch[i] + 1] -= 100000
          end if
        end if

        if cbx_tch_plan.checked then
          For k = 1 to cell_tch_number[i]
                    freq_cost_table[i,cell_freq_tch[i,k]] -= 100000
                    if cell_freq_tch[i,k] > 1 then
freq_cost_table[i,cell_freq_tch[i,k] - 1] -= 100000
                    if cell_freq_tch[i,k] < 124 then
freq_cost_table[i,cell_freq_tch[i,k] + 1] -= 100000
          Next
        else
          For k = 1 to cell_tch_number[i]
                    if fixed_tch_list[i,k] > 0 then
                              freq_cost_table[i,cell_freq_tch[i,k]] -= 100000
                              if cell_freq_tch[i,k] > 1 then
freq_cost_table[i,cell_freq_tch[i,k] - 1] -= 100000
                              if cell_freq_tch[i,k] < 124 then
freq_cost_table[i,cell_freq_tch[i,k] + 1] -= 100000
                    end if
          Next
        end if
```

```
Next
end subroutine

public subroutine f_check_new_plan ();long
i,o,k,cell_no,src_cll_ind,same_cell_no
string forbidden_use_list, fixed_not_use_list, co_list, adj_list
boolean flag_fixed_tch_use
string site_code,cell_name

For i = 1 to cell_count

        //Forbidden kullanilanlar
        forbidden_use_list = ""
        For o = 1 to forbidden_bcch_count[i]
          if cell_freq_bcch[i] = forbidden_bcch_list[i,o] then
                  forbidden_use_list = string(cell_freq_bcch[i])

        d_afp_cell_frequencies_best.setitem(i,"forbidden_bcch_use",forbid
den_use_list)
                  exit
          end if
        Next

        forbidden_use_list = ""
        For o = 1 to forbidden_tch_count[i]
          For k = 1 to cell_tch_number[i]
                  if cell_freq_tch[i,k] = forbidden_tch_list[i,o] then
                          forbidden_use_list = forbidden_use_list +
string(cell_freq_tch[i,k]) + ","
                          exit
                  end if
          Next
        Next
        if len(forbidden_use_list) > 0 then
          forbidden_use_list = left(
forbidden_use_list,(len(forbidden_use_list) - 1) )

        d_afp_cell_frequencies_best.setitem(i,"forbidden_tch_use",forbidd
en_use_list)
```

```
        end if

        //Fixed kullanilmayanlar
        if cbx_bcch_plan.checked then
          fixed_not_use_list = ""
          if fixed_bcch_list[i] > 0 then
                  if cell_freq_bcch[i] <> fixed_bcch_list[i] then
                          fixed_not_use_list = string(fixed_bcch_list[i])

        d_afp_cell_frequencies_best.setitem(i,"fixed_bcch_not_use",fixed_
not_use_list)
                  end if
          end if
        end if

        if cbx_tch_plan.checked then
          fixed_not_use_list = ""
          For o = 1 to fixed_tch_count[i]
                  flag_fixed_tch_use = false
                  For k = 1 to cell_tch_number[i]
                          if cell_freq_tch[i,k] = fixed_tch_list[i,o] then
                                  flag_fixed_tch_use = true
                          end if
                  Next
                  if flag_fixed_tch_use = false then
                          fixed_not_use_list = fixed_not_use_list +
string(fixed_tch_list[i,o]) + ","
                  end if
          Next
          if len(fixed_not_use_list) > 0 then
                  fixed_not_use_list = left(
fixed_not_use_list,(len(fixed_not_use_list) - 1) )

        d_afp_cell_frequencies_best.setitem(i,"fixed_tch_not_use",fixed_n
ot_use_list)
          end if
        end if
```

```
        //Same Cell de Co ve Adjacent Frekans kontrolu
        co_list = ""
        adj_list = ""

        if cbx_bcch_plan.checked then
          For k = 1 to cell_tch_number[i]
                if cell_freq_bcch[i] = cell_freq_tch[i,k] then
                        co_list = co_list + string(cell_freq_bcch[i]) + ","
                elseif Abs(cell_freq_bcch[i] - cell_freq_tch[i,k]) = 1 then
                        adj_list = adj_list + string(cell_freq_bcch[i]) + "-"
+ string(cell_freq_tch[i,k]) + ","
                end if
          Next
        end if

        if cbx_tch_plan.checked then
          For k = 1 to (cell_tch_number[i] - 1)
                For o = (k + 1) to cell_tch_number[i]
                        if cell_freq_tch[i,k] = cell_freq_tch[i,o] then
                                co_list = co_list + string(cell_freq_tch[i,k])
+ ","
                        elseif Abs(cell_freq_tch[i,k] - cell_freq_tch[i,o]) =
1 then
                                adj_list = adj_list +
string(cell_freq_tch[i,k]) + "-" + string(cell_freq_tch[i,o]) + ","
                        end if
                Next
          Next
        end if

        if len(co_list) > 0 then
          co_list = left( co_list,(len(co_list) - 1) )

        d_afp_cell_frequencies_best.setitem(i,"same_cell_co_use",co_list)
        end if
        if len(adj_list) > 0 then
          adj_list = left( adj_list,(len(adj_list) - 1) )
```

```
        d_afp_cell_frequencies_best.setitem(i,"same_cell_adj_use",adj_list
)
        end if


        //Cosite da Co ve Adjacent Frekans kontrolu
        cell_name = lb_cell_name.text(i)
        same_cell_no = long( mid(cell_name,6,1) )
        site_code = left(cell_name,5)

        co_list = ""
        adj_list = ""

        For cell_no = 1 to 4

          if cell_no <> same_cell_no then
                  cell_name = site_code + string(cell_no)
                  src_cll_ind = lb_cell_name.FindItem(cell_name,0)

                  if src_cll_ind > 0 then

                          if cbx_bcch_plan.checked then
                                  if cell_freq_bcch[i] =
cell_freq_bcch[src_cll_ind] then
                                          co_list = co_list +
string(cell_freq_bcch[i]) + ","
                                  elseif Abs(cell_freq_bcch[i] -
cell_freq_bcch[src_cll_ind]) = 1 then
                                          adj_list = adj_list +
string(cell_freq_bcch[i]) + "-" + string(cell_freq_bcch[src_cll_ind]) + ","

                                  end if
                          end if

                          if cbx_bcch_plan.checked and
cbx_tch_plan.checked then
                                  For k = 1 to cell_tch_number[src_cll_ind]
```

```
                                        if cell_freq_bcch[i] =
cell_freq_tch[src_cll_ind,k] then
                                            co_list = co_list +
string(cell_freq_bcch[i]) + ","
                                        elseif Abs(cell_freq_bcch[i] -
cell_freq_tch[src_cll_ind,k]) = 1 then
                                            adj_list = adj_list +
string(cell_freq_bcch[i]) + "-" + string(cell_freq_tch[src_cll_ind,k]) + ","
                                        end if
                                    Next
                                end if

                                if cbx_tch_plan.checked then
                                    For k = 1 to cell_tch_number[i]
                                        For o = 1 to
cell_tch_number[src_cll_ind]
                                            if cell_freq_tch[i,k] =
cell_freq_tch[src_cll_ind,o] then
                                                co_list = co_list +
string(cell_freq_tch[i,k]) + ","
                                            elseif Abs(cell_freq_tch[i,k]
- cell_freq_tch[src_cll_ind,o]) = 1 then
                                                adj_list = adj_list +
string(cell_freq_tch[i,k]) + "-" + string(cell_freq_tch[src_cll_ind,o]) + ","
                                            end if
                                        Next
                                    Next
                                end if

                            end if
                        end if

                    Next

                    if len(co_list) > 0 then
                      co_list = left( co_list,(len(co_list) - 1) )
                      d_afp_cell_frequencies_best.setitem(i,"cosite_co_use",co_list)
                    end if
                    if len(adj_list) > 0 then
```

```
        adj_list = left( adj_list,(len(adj_list) - 1) )
        d_afp_cell_frequencies_best.setitem(i,"cosite_adj_use",adj_list)
      end if

Next
end subroutine

public subroutine f_check_initial_plan ();long
i,o,k,cell_no,src_cll_ind,same_cell_no
string forbidden_use_list, fixed_not_use_list, co_list, adj_list
boolean flag_fixed_tch_use
string site_code,cell_name

For i = 1 to cell_count

      //Forbidden kullanilanlar
      forbidden_use_list = ""
      For o = 1 to forbidden_bcch_count[i]
        if cell_freq_bcch[i] = forbidden_bcch_list[i,o] then
              forbidden_use_list = string(cell_freq_bcch[i])

      d_afp_cell_frequencies.setitem(i,"forbidden_bcch_use",forbidden_
use_list)
              exit
        end if
      Next

      forbidden_use_list = ""
      For o = 1 to forbidden_tch_count[i]
        For k = 1 to cell_tch_number[i]
              if cell_freq_tch[i,k] = forbidden_tch_list[i,o] then
                    forbidden_use_list = forbidden_use_list +
string(cell_freq_tch[i,k]) + ","
                    exit
              end if
        Next
      Next
      if len(forbidden_use_list) > 0 then
```

```
          forbidden_use_list = left(
forbidden_use_list,(len(forbidden_use_list) - 1) )

        d_afp_cell_frequencies.setitem(i,"forbidden_tch_use",forbidden_us
e_list)
        end if

        //Fixed kullanilmayanlar
        if cbx_bcch_plan.checked then
          fixed_not_use_list = ""
          if fixed_bcch_list[i] > 0 then
                  if cell_freq_bcch[i] <> fixed_bcch_list[i] then
                          fixed_not_use_list = string(fixed_bcch_list[i])

        d_afp_cell_frequencies.setitem(i,"fixed_bcch_not_use",fixed_not_
use_list)
                  end if
          end if
        end if

        if cbx_tch_plan.checked then
          fixed_not_use_list = ""
          For o = 1 to fixed_tch_count[i]
                  flag_fixed_tch_use = false
                  For k = 1 to cell_tch_number[i]
                          if cell_freq_tch[i,k] = fixed_tch_list[i,o] then
                                  flag_fixed_tch_use = true
                          end if
                  Next
                  if flag_fixed_tch_use = false then
                          fixed_not_use_list = fixed_not_use_list +
string(fixed_tch_list[i,o]) + ","
                  end if
          Next
          if len(fixed_not_use_list) > 0 then
                  fixed_not_use_list = left(
fixed_not_use_list,(len(fixed_not_use_list) - 1) )
```

```
        d_afp_cell_frequencies.setitem(i,"fixed_tch_not_use",fixed_not_us
e_list)
          end if
        end if

        //Same Cell de Co ve Adjacent Frekans kontrolu
        co_list = ""
        adj_list = ""

        if cbx_bcch_plan.checked then
          For k = 1 to cell_tch_number[i]
                if cell_freq_bcch[i] = cell_freq_tch[i,k] then
                        co_list = co_list + string(cell_freq_bcch[i]) + ","
                elseif Abs(cell_freq_bcch[i] - cell_freq_tch[i,k]) = 1 then
                        adj_list = adj_list + string(cell_freq_bcch[i]) + "-"
+ string(cell_freq_tch[i,k]) + ","
                end if
          Next
        end if

        if cbx_tch_plan.checked then
          For k = 1 to (cell_tch_number[i] - 1)
                For o = (k + 1) to cell_tch_number[i]
                        if cell_freq_tch[i,k] = cell_freq_tch[i,o] then
                                co_list = co_list + string(cell_freq_tch[i,k])
+ ","
                        elseif Abs(cell_freq_tch[i,k] - cell_freq_tch[i,o]) =
1 then
                                adj_list = adj_list +
string(cell_freq_tch[i,k]) + "-" + string(cell_freq_tch[i,o]) + ","
                        end if
                Next
          Next
        end if

        if len(co_list) > 0 then
          co_list = left( co_list,(len(co_list) - 1) )
          d_afp_cell_frequencies.setitem(i,"same_cell_co_use",co_list)
```

```
        end if
        if len(adj_list) > 0 then
          adj_list = left( adj_list,(len(adj_list) - 1) )
          d_afp_cell_frequencies.setitem(i,"same_cell_adj_use",adj_list)
        end if


        //Cosite da Co ve Adjacent Frekans kontrolu
        cell_name = lb_cell_name.text(i)
        same_cell_no = long( mid(cell_name,6,1) )
        site_code = left(cell_name,5)

        co_list = ""
        adj_list = ""

        For cell_no = 1 to 4

          if cell_no <> same_cell_no then
                  cell_name = site_code + string(cell_no)
                  src_cll_ind = lb_cell_name.FindItem(cell_name,0)

                  if src_cll_ind > 0 then

                          if cbx_bcch_plan.checked then

                                  if cell_freq_bcch[i] =
cell_freq_bcch[src_cll_ind] then
                                          co_list = co_list +
string(cell_freq_bcch[i]) + ","
                                  elseif Abs(cell_freq_bcch[i] -
cell_freq_bcch[src_cll_ind]) = 1 then
                                          adj_list = adj_list +
string(cell_freq_bcch[i]) + "-" + string(cell_freq_bcch[src_cll_ind]) + ","

                                  end if
                          end if

                          if cbx_bcch_plan.checked and
cbx_tch_plan.checked then
```

```
                                For k = 1 to cell_tch_number[src_cll_ind]
                                    if cell_freq_bcch[i] =
cell_freq_tch[src_cll_ind,k] then
                                        co_list = co_list +
string(cell_freq_bcch[i]) + ","
                                    elseif Abs(cell_freq_bcch[i] -
cell_freq_tch[src_cll_ind,k]) = 1 then
                                        adj_list = adj_list +
string(cell_freq_bcch[i]) + "-" + string(cell_freq_tch[src_cll_ind,k]) + ","
                                    end if
                                Next
                            end if

                            if cbx_tch_plan.checked then
                                For k = 1 to cell_tch_number[i]
                                    For o = 1 to
cell_tch_number[src_cll_ind]
                                        if cell_freq_tch[i,k] =
cell_freq_tch[src_cll_ind,o] then
                                            co_list = co_list +
string(cell_freq_tch[i,k]) + ","
                                        elseif Abs(cell_freq_tch[i,k]
- cell_freq_tch[src_cll_ind,o]) = 1 then
                                            adj_list = adj_list +
string(cell_freq_tch[i,k]) + "-" + string(cell_freq_tch[src_cll_ind,o]) + ","
                                        end if
                                    Next
                                Next
                            end if

                        end if
                    end if

                Next

                if len(co_list) > 0 then
                  co_list = left( co_list,(len(co_list) - 1) )
                  d_afp_cell_frequencies.setitem(i,"cosite_co_use",co_list)
                end if
```

```
        if len(adj_list) > 0 then
          adj_list = left( adj_list,(len(adj_list) - 1) )
          d_afp_cell_frequencies.setitem(i,"cosite_adj_use",adj_list)
        end if

Next
end subroutine

public subroutine f_add_cost_in_cosite ();long i
decimal co_affect, adj_affect
long cell_no,same_cell_no,src_cll_ind,int_cll_ind
string cell_name,site_code

co_affect = same_ch_cost
adj_affect = adj_ch_cost

For i = 1 to cell_count

        src_cll_ind = i
        cell_name = lb_cell_name.text(i)
        site_code = left(cell_name,5)
        same_cell_no = long(mid(cell_name,6,1))

        For cell_no = 1 to 4
          if cell_no <> same_cell_no then
                  if flag_cosite_cost_added[i,cell_no] = false then

                          flag_cosite_cost_added[i,cell_no] = true
                          cell_name = site_code + string(cell_no)
                          int_cll_ind = lb_cell_name.FindItem(cell_name,0)

                          if int_cll_ind > 0 then
                                  cell_interfered_count[int_cll_ind]++

        icdm_interfered_ind[int_cll_ind,cell_interfered_count[int_cll_ind]]
= src_cll_ind

        icdm_co_int[int_cll_ind,cell_interfered_count[int_cll_ind]] =
co_affect
```

```
        icdm_adj_int[int_cll_ind,cell_interfered_count[int_cll_ind]] =
adj_affect

                                    cell_interferer_count[src_cll_ind]++

        icdm_interferer_ind[src_cll_ind,cell_interferer_count[src_cll_ind]]
= int_cll_ind

        icdm_co[src_cll_ind,cell_interferer_count[src_cll_ind]] = co_affect

        icdm_adj[src_cll_ind,cell_interferer_count[src_cll_ind]] =
adj_affect
                            end if

                    end if
          end if
        Next

Next
end subroutine

public subroutine f_make_best_plan_dw ();decimal
result_cost,total_cost,bcch_cost,tch_cost
long i,j,tru_number

        // Best plani dw ye atma
        For i = 1 to cell_count
          d_afp_cell_frequencies_best.insertrow(i)

        d_afp_cell_frequencies_best.setitem(i,"cell_name",lb_cell_name.te
xt(i))
          tru_number = cell_tch_number[i] + 1
          d_afp_cell_frequencies_best.setitem(i,"tru_number",tru_number)

          bcch_cost = 0
          if cbx_bcch_plan.checked then
```

```
        d_afp_cell_frequencies_best.setitem(i,"bcch_cost",(freq_cost_table
[i,cell_freq_bcch[i]] - 100000))

        d_afp_cell_frequencies_best.setitem(i,"bcch",cell_freq_bcch[i])
                bcch_cost = (freq_cost_table[i,cell_freq_bcch[i]] -
100000)
          end if

          tch_cost = 0
          result_cost = 0
          if cbx_tch_plan.checked then
                For j = 1 to cell_tch_number[i]
                        result_cost = result_cost +
(freq_cost_table[i,cell_freq_tch[i,j]] - 100000)

        d_afp_cell_frequencies_best.setitem(i,("tch"+string(j)),cell_freq_tc
h[i,j])
                Next

        d_afp_cell_frequencies_best.setitem(i,"tch_cost",result_cost)
                tch_cost = result_cost
          end if

          total_cost = bcch_cost + tch_cost
          d_afp_cell_frequencies_best.setitem(i,"total_cost",total_cost)

        Next
end subroutine

public subroutine f_change_bcch_frequency (long cell_order, long
new_frequency);long o,freq_level,src_cll_ind
decimal co_affect,adj_affect
long frequency_no

freq_level = cell_order
frequency_no = new_frequency

For o = 1 to cell_interfered_count[freq_level]
```

```
        src_cll_ind = icdm_interfered_ind[freq_level,o]

        co_affect = icdm_co_int[freq_level,o]
        adj_affect = icdm_adj_int[freq_level,o]

        freq_cost_table[src_cll_ind,frequency_no] += (co_affect * 10)
        if frequency_no > 1 then freq_cost_table[src_cll_ind,frequency_no
- 1] += (adj_affect * 10)
        if frequency_no < 124 then
freq_cost_table[src_cll_ind,frequency_no + 1] += (adj_affect * 10)

        freq_cost_table[src_cll_ind,cell_freq_bcch[freq_level]] -=
(co_affect * 10)
        if cell_freq_bcch[freq_level] > 1 then
freq_cost_table[src_cll_ind,cell_freq_bcch[freq_level] - 1] -= (adj_affect
* 10)
        if cell_freq_bcch[freq_level] < 124 then
freq_cost_table[src_cll_ind,cell_freq_bcch[freq_level] + 1] -= (adj_affect
* 10)

Next

freq_cost_table[freq_level,cell_freq_bcch[freq_level]] -= 100000
if cell_freq_bcch[freq_level] > 1 then
freq_cost_table[freq_level,cell_freq_bcch[freq_level] - 1] -= 100000
if cell_freq_bcch[freq_level] < 124 then
freq_cost_table[freq_level,cell_freq_bcch[freq_level] + 1] -= 100000

cell_freq_bcch[freq_level] = frequency_no

freq_cost_table[freq_level,frequency_no] += 100000
if frequency_no > 1 then freq_cost_table[freq_level,frequency_no - 1] +=
100000
if frequency_no < 124 then freq_cost_table[freq_level,frequency_no + 1]
+= 100000
end subroutine
```

```
public subroutine f_change_tch_frequency (long cell_order, long
tch_order, integer new_frequency);long o,freq_level,src_cll_ind
decimal co_affect,adj_affect
long frequency_no,k

freq_level = cell_order
k = tch_order
frequency_no = new_frequency

For o = 1 to cell_interfered_count[freq_level]

        src_cll_ind = icdm_interfered_ind[freq_level,o]

        co_affect = icdm_co_int[freq_level,o]
        adj_affect = icdm_adj_int[freq_level,o]

        freq_cost_table[src_cll_ind,frequency_no] += (co_affect)
        if frequency_no > 1 then freq_cost_table[src_cll_ind,frequency_no
- 1] += (adj_affect)
        if frequency_no < 124 then
freq_cost_table[src_cll_ind,frequency_no + 1] += (adj_affect)

        freq_cost_table[src_cll_ind,cell_freq_tch[freq_level,k]] -=
(co_affect)
        if cell_freq_tch[freq_level,k] > 1 then
freq_cost_table[src_cll_ind,cell_freq_tch[freq_level,k] - 1] -=
(adj_affect)
        if cell_freq_tch[freq_level,k] < 124 then
freq_cost_table[src_cll_ind,cell_freq_tch[freq_level,k] + 1] -=
(adj_affect)

Next

freq_cost_table[freq_level,cell_freq_tch[freq_level,k]] -= 100000
if cell_freq_tch[freq_level,k] > 1 then
freq_cost_table[freq_level,cell_freq_tch[freq_level,k] - 1] -= 100000
if cell_freq_tch[freq_level,k] < 124 then
freq_cost_table[freq_level,cell_freq_tch[freq_level,k] + 1] -= 100000
```

```
cell_freq_tch[freq_level,k] = frequency_no

freq_cost_table[freq_level,frequency_no] += 100000
if frequency_no > 1 then freq_cost_table[freq_level,frequency_no - 1] +=
100000
if frequency_no < 124 then freq_cost_table[freq_level,frequency_no + 1]
+= 100000
end subroutine

public function decimal f_calculate_cost_new ();decimal
result_cost,total_cost
long p,j

total_cost = 0

if cbx_bcch_plan.checked = true then
        result_cost = 0
        For p = 1 to cell_count
          result_cost = result_cost + freq_cost_table[p,cell_freq_bcch[p]] -
100000
        Next

        total_cost = result_cost
end if

if cbx_tch_plan.checked = true then
        result_cost = 0
        For p = 1 to cell_count
          For j = 1 to cell_tch_number[p]
                  result_cost = result_cost +
freq_cost_table[p,cell_freq_tch[p,j]] - 100000
          Next
        Next

        total_cost += result_cost
end if

return total_cost
end function
```

```
on w_afp.create
int iCurrent
call super::create
this.d_afp_cells=create d_afp_cells
this.cb_import_cells=create cb_import_cells
this.d_afp_icdm=create d_afp_icdm
this.cb_import_icdm=create cb_import_icdm
this.cb_plan=create cb_plan
this.lb_all_frequencies=create lb_all_frequencies
this.st_all_freq=create st_all_freq
this.lb_bcch_frequencies=create lb_bcch_frequencies
this.cb_add_bcch=create cb_add_bcch
this.st_bcch_freq=create st_bcch_freq
this.cb_remove_bcch=create cb_remove_bcch
this.st_all_freq_3=create st_all_freq_3
this.lb_all_frequencies_3=create lb_all_frequencies_3
this.st_tch_freq=create st_tch_freq
this.lb_tch_frequencies=create lb_tch_frequencies
this.cb_add_tch=create cb_add_tch
this.cb_remove_tch=create cb_remove_tch
this.d_afp_cell_frequencies=create d_afp_cell_frequencies
this.st_initial_cost=create st_initial_cost
this.st_initial_cost_label=create st_initial_cost_label
this.cb_optimize=create cb_optimize
this.cb_save_initial=create cb_save_initial
this.st_best_cost_label=create st_best_cost_label
this.st_best_cost=create st_best_cost
this.d_afp_cell_frequencies_best=create d_afp_cell_frequencies_best
this.sle_optimize_times=create sle_optimize_times
this.st_times=create st_times
this.st_status=create st_status
this.st_status_label=create st_status_label
this.d_afp_initial_plan=create d_afp_initial_plan
this.cb_import_initial_plan=create cb_import_initial_plan
this.cb_clear_initial_plan=create cb_clear_initial_plan
this.d_afp_icdm_interferer=create d_afp_icdm_interferer
this.cb_check_plan=create cb_check_plan
this.dr_frequency=create dr_frequency
```

```
this.lb_cell_name=create lb_cell_name
this.lb_external_name=create lb_external_name
this.cbx_bcch_plan=create cbx_bcch_plan
this.cbx_tch_plan=create cbx_tch_plan
this.st_best_bcch_cost=create st_best_bcch_cost
this.st_best_tch_cost=create st_best_tch_cost
this.st_best_bcch_cost_label=create st_best_bcch_cost_label
this.st_best_tch_cost_label=create st_best_tch_cost_label
this.gb_3=create gb_3
this.gb_1=create gb_1
this.cb_save_best=create cb_save_best
this.gb_2=create gb_2
this.st_initial_bcch_cost_label=create st_initial_bcch_cost_label
this.st_initial_tch_cost_label=create st_initial_tch_cost_label
this.st_initial_bcch_cost=create st_initial_bcch_cost
this.st_initial_tch_cost=create st_initial_tch_cost
this.cb_import_icdm_msmt=create cb_import_icdm_msmt
this.sle_same_ch_cost=create sle_same_ch_cost
this.sle_adj_ch_cost=create sle_adj_ch_cost
this.sle_forbid_ch_cost=create sle_forbid_ch_cost
this.st_1=create st_1
this.st_2=create st_2
this.st_3=create st_3
this.st_4=create st_4
iCurrent=UpperBound(this.Control)
this.Control[iCurrent+1]=this.d_afp_cells
this.Control[iCurrent+2]=this.cb_import_cells
this.Control[iCurrent+3]=this.d_afp_icdm
this.Control[iCurrent+4]=this.cb_import_icdm
this.Control[iCurrent+5]=this.cb_plan
this.Control[iCurrent+6]=this.lb_all_frequencies
this.Control[iCurrent+7]=this.st_all_freq
this.Control[iCurrent+8]=this.lb_bcch_frequencies
this.Control[iCurrent+9]=this.cb_add_bcch
this.Control[iCurrent+10]=this.st_bcch_freq
this.Control[iCurrent+11]=this.cb_remove_bcch
this.Control[iCurrent+12]=this.st_all_freq_3
this.Control[iCurrent+13]=this.lb_all_frequencies_3
this.Control[iCurrent+14]=this.st_tch_freq
```

```
this.Control[iCurrent+15]=this.lb_tch_frequencies
this.Control[iCurrent+16]=this.cb_add_tch
this.Control[iCurrent+17]=this.cb_remove_tch
this.Control[iCurrent+18]=this.d_afp_cell_frequencies
this.Control[iCurrent+19]=this.st_initial_cost
this.Control[iCurrent+20]=this.st_initial_cost_label
this.Control[iCurrent+21]=this.cb_optimize
this.Control[iCurrent+22]=this.cb_save_initial
this.Control[iCurrent+23]=this.st_best_cost_label
this.Control[iCurrent+24]=this.st_best_cost
this.Control[iCurrent+25]=this.d_afp_cell_frequencies_best
this.Control[iCurrent+26]=this.sle_optimize_times
this.Control[iCurrent+27]=this.st_times
this.Control[iCurrent+28]=this.st_status
this.Control[iCurrent+29]=this.st_status_label
this.Control[iCurrent+30]=this.d_afp_initial_plan
this.Control[iCurrent+31]=this.cb_import_initial_plan
this.Control[iCurrent+32]=this.cb_clear_initial_plan
this.Control[iCurrent+33]=this.d_afp_icdm_interferer
this.Control[iCurrent+34]=this.cb_check_plan
this.Control[iCurrent+35]=this.dr_frequency
this.Control[iCurrent+36]=this.lb_cell_name
this.Control[iCurrent+37]=this.lb_external_name
this.Control[iCurrent+38]=this.cbx_bcch_plan
this.Control[iCurrent+39]=this.cbx_tch_plan
this.Control[iCurrent+40]=this.st_best_bcch_cost
this.Control[iCurrent+41]=this.st_best_tch_cost
this.Control[iCurrent+42]=this.st_best_bcch_cost_label
this.Control[iCurrent+43]=this.st_best_tch_cost_label
this.Control[iCurrent+44]=this.gb_3
this.Control[iCurrent+45]=this.gb_1
this.Control[iCurrent+46]=this.cb_save_best
this.Control[iCurrent+47]=this.gb_2
this.Control[iCurrent+48]=this.st_initial_bcch_cost_label
this.Control[iCurrent+49]=this.st_initial_tch_cost_label
this.Control[iCurrent+50]=this.st_initial_bcch_cost
this.Control[iCurrent+51]=this.st_initial_tch_cost
this.Control[iCurrent+52]=this.cb_import_icdm_msmt
this.Control[iCurrent+55]=this.sle_same_ch_cost
```

```
this.Control[iCurrent+56]=this.sle_adj_ch_cost
this.Control[iCurrent+57]=this.sle_forbid_ch_cost
this.Control[iCurrent+58]=this.st_1
this.Control[iCurrent+59]=this.st_2
this.Control[iCurrent+60]=this.st_3
this.Control[iCurrent+61]=this.st_4
end on

on w_afp.destroy
call super::destroy
destroy(this.d_afp_cells)
destroy(this.cb_import_cells)
destroy(this.d_afp_icdm)
destroy(this.cb_import_icdm)
destroy(this.cb_plan)
destroy(this.lb_all_frequencies)
destroy(this.st_all_freq)
destroy(this.lb_bcch_frequencies)
destroy(this.cb_add_bcch)
destroy(this.st_bcch_freq)
destroy(this.cb_remove_bcch)
destroy(this.st_all_freq_3)
destroy(this.lb_all_frequencies_3)
destroy(this.st_tch_freq)
destroy(this.lb_tch_frequencies)
destroy(this.cb_add_tch)
destroy(this.cb_remove_tch)
destroy(this.d_afp_cell_frequencies)
destroy(this.st_initial_cost)
destroy(this.st_initial_cost_label)
destroy(this.cb_optimize)
destroy(this.cb_save_initial)
destroy(this.st_best_cost_label)
destroy(this.st_best_cost)
destroy(this.d_afp_cell_frequencies_best)
destroy(this.sle_optimize_times)
destroy(this.st_times)
destroy(this.st_status)
destroy(this.st_status_label)
```

```
destroy(this.d_afp_initial_plan)
destroy(this.cb_import_initial_plan)
destroy(this.cb_clear_initial_plan)
destroy(this.d_afp_icdm_interferer)
destroy(this.cb_check_plan)
destroy(this.dr_frequency)
destroy(this.lb_cell_name)
destroy(this.lb_external_name)
destroy(this.cbx_bcch_plan)
destroy(this.cbx_tch_plan)
destroy(this.st_best_bcch_cost)
destroy(this.st_best_tch_cost)
destroy(this.st_best_bcch_cost_label)
destroy(this.st_best_tch_cost_label)
destroy(this.gb_3)
destroy(this.gb_1)
destroy(this.cb_save_best)
destroy(this.gb_2)
destroy(this.st_initial_bcch_cost_label)
destroy(this.st_initial_tch_cost_label)
destroy(this.st_initial_bcch_cost)
destroy(this.st_initial_tch_cost)
destroy(this.cb_import_icdm_msmt)
destroy(this.sle_same_ch_cost)
destroy(this.sle_adj_ch_cost)
destroy(this.sle_forbid_ch_cost)
destroy(this.st_1)
destroy(this.st_2)
destroy(this.st_3)
destroy(this.st_4)
end on

event open;call super::open;long i

For i = 1 to 124

        if (i >= 10 and i <= 19) or (i >=81 and i <= 89) then
          lb_bcch_frequencies.additem(string(i,"000"))
          bcch_frequency_list[i] = true
```

```
        else
          lb_all_frequencies.additem(string(i,"000"))
          bcch_frequency_list[i] = false
        end if

        if (i >=91 and i <= 120) then
          lb_tch_frequencies.additem(string(i,"000"))
          tch_frequency_list[i] = true
        else
          lb_all_frequencies_3.additem(string(i,"000"))
          tch_frequency_list[i] = false
        end if

Next
end event

type d_afp_cells from datawindow_common within w_afp
int X=27
int Y=44
int Width=2359
int Height=1032
int TabOrder=10
boolean BringToTop=true
string DataObject="dwe_afp_cells"
boolean TitleBar=false
BorderStyle BorderStyle=StyleLowered!
boolean ControlMenu=false
boolean MinBox=false
boolean MaxBox=false
boolean HScrollBar=false
boolean Resizable=false
end type

type cb_import_cells from commandbutton within w_afp
int X=50
int Y=1100
int Width=343
int Height=144
int TabOrder=120
```

```
boolean BringToTop=true
string Text="Import Cells"
int TextSize=-8
int Weight=400
string FaceName="Arial TUR"
FontCharSet FontCharSet=TurkishCharSet!
FontFamily FontFamily=Swiss!
FontPitch FontPitch=Variable!
end type

event clicked;string file_name
long i,def_ch,tru_number, rc

setNull(file_name)

d_afp_cells.reset()
d_afp_cells.ImportFile ( file_name )

rc = d_afp_cells.rowcount()
if rc > 0 then

	d_afp_cells.SetRedraw(false)
	d_afp_cells.SetSort("#1 A")
	d_afp_cells.Sort()
	d_afp_cells.SetRedraw(true)

	cb_import_cells.enabled = false

	st_status.text = "Cells imported!"

end if
end event

type d_afp_icdm from datawindow_common within w_afp
int X=2446
int Y=44
int Width=1385
int Height=1032
int TabOrder=110
```

```
boolean BringToTop=true
string DataObject="dwe_afp_icdm"
boolean TitleBar=false
BorderStyle BorderStyle=StyleLowered!
boolean ControlMenu=false
boolean MinBox=false
boolean MaxBox=false
boolean Resizable=false
end type

type cb_import_icdm from commandbutton within w_afp
int X=2464
int Y=1100
int Width=421
int Height=144
int TabOrder=130
boolean BringToTop=true
string Text="Import ICDM (.txt)"
int TextSize=-8
int Weight=400
string FaceName="Arial TUR"
FontCharSet FontCharSet=TurkishCharSet!
FontFamily FontFamily=Swiss!
FontPitch FontPitch=Variable!
end type

event clicked;string docname, named
integer value

value = GetFileOpenName("Select ICDM File",docname, named,
"TXT","Text Files (*.TXT),*.TXT")

IF value = 1 THEN

        d_afp_icdm.reset()
        d_afp_icdm.ImportFile ( docname )

        d_afp_icdm.SetRedraw(false)
        d_afp_icdm.SetSort("#1 A, #2 A")
```

```
        d_afp_icdm.Sort()
        d_afp_icdm.SetRedraw(true)

        d_afp_icdm_interferer.reset()
        d_afp_icdm_interferer.ImportFile ( docname )
        d_afp_icdm_interferer.SetSort("#2 A, #1 A")
        d_afp_icdm_interferer.Sort()

        cb_import_icdm.enabled = false

        st_status.text = "ICDM imported!"

END IF
end event

type cb_plan from commandbutton within w_afp
int X=2917
int Y=1480
int Width=631
int Height=136
int TabOrder=160
boolean BringToTop=true
string Text="New Frequency Planing"
int TextSize=-8
int Weight=400
string FaceName="Arial TUR"
FontCharSet FontCharSet=TurkishCharSet!
FontFamily FontFamily=Swiss!
FontPitch FontPitch=Variable!
end type

event clicked;long i,j,rc,last_row,o,p,k,m,n
string nul
decimal co_affect,adj_affect,result_cost, best_cost, total_cost,
last_best_cost
long tch_count,bcch,tch,tru_number,frequency_no,cell_no
string src_cll, int_cll, tch_string, forbidden_bcch_string,
forbidden_tch_string, fixed_tch_string
long src_cll_ind, int_cll_ind, ext_cll_ind
```

```
decimal src_trf
long freq_level,random_bcch,random_tch
long plan_file,optimize_times
string log_text,plan_text

//---------------------Initial Islemler--------------------------
log_file = FileOpen("c:\neptune\neptune_afp_log.txt", LineMode!,
Write!, LockWrite!, Replace!)
log_text = string(today()) + " - " + string(now()) + " :  Planning started!"
FileWrite(log_file , log_text)

if len(sle_optimize_times.text) > 0 then
        optimize_times = long(sle_optimize_times.text)
else
        optimize_times = 1
end if

same_ch_cost = long(sle_same_ch_cost.text)
adj_ch_cost = long(sle_adj_ch_cost.text)
forbid_ch_cost = long(sle_forbid_ch_cost.text)

lb_cell_name.reset()
lb_external_name.reset()

setnull(nul)
st_status.text = nul
st_best_cost.text = nul
st_best_bcch_cost.text = nul
st_best_tch_cost.text = nul

rc = d_afp_cells.rowcount()
if rc < 1 then
        Messagebox("Warning","Cells table must be imported")
        return
end if
cell_count = rc

last_row = d_afp_icdm.rowcount()
if last_row < 1 then
```

```
        Messagebox("Warning","ICDM table must be imported")
        return
end if

if not(cbx_tch_plan.checked) and not(cbx_bcch_plan.checked) then
        messagebox("Warning","TCH, BCCH or both checkboxes must be
selected")
        return
end if

enabled = false
SetPointer(HourGlass!)

d_afp_icdm.SetRedraw(false)
d_afp_cells.SetRedraw(false)

st_status.text = "Wait for planing!"

d_afp_cell_frequencies_best.reset()

//Kullanilan Frekanslarin belirlenmesi
if cbx_bcch_plan.checked = true then f_set_random_bcch()
if cbx_tch_plan.checked = true then f_set_random_tch()

//Nbr Count hesaplama
f_calculate_nbr_count()
d_afp_cells.SetSort("#9 D, #4 D") //Nbr Count , Tot Traf
d_afp_cells.Sort()


//DW lerin Arraylara Aktarimi
f_set_arrays_from_dw()

//Temizleme islemleri
f_clear_freq_cost_table()
f_clear_forbidden_and_fixed_list()
f_clear_cell_freqs()

For i = 1 to 4000
```

```
        For cell_no = 1 to 4
          flag_cosite_cost_added[i,cell_no] = false
        Next
        flag_check_frequency[i] = true
Next

//Forbidden ve Fixed Frekanslarin DW den arraylere aktarimi
f_set_forbidden_and_fixed_freq()

ext_cll_ind = 3000

For i = 1 to last_row

        src_cll = d_afp_icdm.getitemstring(i,"source_cell")
        src_cll_ind = lb_cell_name.FindItem(src_cll,0)
        if src_cll_ind < 0 then
          continue
        end if

        int_cll = d_afp_icdm.getitemstring(i,"interferer_cell")

        src_trf = cell_traffic[src_cll_ind]

        if left(int_cll,5) = left(src_cll,5) then // cosite ta costu yukselt
          co_affect = same_ch_cost
          adj_affect = adj_ch_cost
          cell_no = long(mid(int_cll,6,1))
          flag_cosite_cost_added[src_cll_ind,cell_no] = true
        else
          co_affect = 0
          adj_affect = 0
        end if

        int_cll_ind = lb_cell_name.FindItem(int_cll,0)
        if int_cll_ind < 0 then //external cell

          int_cll_ind = lb_external_name.FindItem(int_cll,0)
          if int_cll_ind < 0 then // new external
```

```
                bcch = f_get_bcch_from_cell(int_cll)

                if (bcch > 0) then

                        ext_cll_ind++
                        lb_external_name.InsertItem(int_cll,ext_cll_ind -
3000)
                        int_cll_ind = ext_cll_ind

                        cell_freq_bcch[int_cll_ind] = bcch
                        freq_cost_table[src_cll_ind,bcch] +=
(((d_afp_icdm.getitemdecimal(i,"traf_affected_co_percent") * src_trf /
100) + co_affect) * 10)
                        if bcch > 1 then freq_cost_table[src_cll_ind,bcch -
1] += (((d_afp_icdm.getitemdecimal(i,"traf_affected_adj_percent") *
src_trf / 100) + adj_affect) * 10)
                        if bcch < 124 then freq_cost_table[src_cll_ind,bcch
+ 1] += (((d_afp_icdm.getitemdecimal(i,"traf_affected_adj_percent") *
src_trf / 100) + adj_affect) * 10)


                        tru_number = 0
                        tch_string = f_get_all_tch_from_cell(int_cll)
                        if len(tch_string) > 0 then
                                tru_number = f_get_from_count(tch_string)
                                For o = 1 to tru_number
                                        tch = long(f_get_from(tch_string,o))
                                        cell_freq_tch[int_cll_ind,o] = tch

                                        freq_cost_table[src_cll_ind,tch] +=
(d_afp_icdm.getitemdecimal(i,"traf_affected_co_percent") * src_trf /
100) + co_affect
                                        if tch > 1 then
freq_cost_table[src_cll_ind,tch - 1] +=
(d_afp_icdm.getitemdecimal(i,"traf_affected_adj_percent") * src_trf /
100) + adj_affect
                                        if tch < 124 then
freq_cost_table[src_cll_ind,tch + 1] +=
```

```
(d_afp_icdm.getitemdecimal(i,"traf_affected_adj_percent") * src_trf /
100) + adj_affect

                                Next
                        end if

                        cell_tch_number[int_cll_ind] = tru_number

                else

                        continue
                end if
        else // current external

                int_cll_ind += 3000

                bcch = cell_freq_bcch[int_cll_ind]
                freq_cost_table[src_cll_ind,bcch] +=
(((d_afp_icdm.getitemdecimal(i,"traf_affected_co_percent") * src_trf /
100) + co_affect) * 10)
                if bcch > 1 then freq_cost_table[src_cll_ind,bcch - 1] +=
(((d_afp_icdm.getitemdecimal(i,"traf_affected_adj_percent") * src_trf /
100) + adj_affect) * 10)
                if bcch < 124 then freq_cost_table[src_cll_ind,bcch + 1]
+= (((d_afp_icdm.getitemdecimal(i,"traf_affected_adj_percent") * src_trf
/ 100) + adj_affect) * 10)


                tru_number = cell_tch_number[int_cll_ind]

                For o = 1 to tru_number
                        tch = cell_freq_tch[int_cll_ind,o]

                        freq_cost_table[src_cll_ind,tch] +=
(d_afp_icdm.getitemdecimal(i,"traf_affected_co_percent") * src_trf /
100) + co_affect
                        if tch > 1 then freq_cost_table[src_cll_ind,tch - 1]
+= (d_afp_icdm.getitemdecimal(i,"traf_affected_adj_percent") * src_trf /
100) + adj_affect
```

```
                        if tch < 124 then freq_cost_table[src_cll_ind,tch +
1] += (d_afp_icdm.getitemdecimal(i,"traf_affected_adj_percent") *
src_trf / 100) + adj_affect

                Next

          end if
        end if

        cell_interfered_count[int_cll_ind]++
        icdm_interfered_ind[int_cll_ind,cell_interfered_count[int_cll_ind]]
= src_cll_ind
        icdm_co_int[int_cll_ind,cell_interfered_count[int_cll_ind]] =
(d_afp_icdm.getitemdecimal(i,"traf_affected_co_percent") * src_trf /
100) + co_affect
        icdm_adj_int[int_cll_ind,cell_interfered_count[int_cll_ind]] =
(d_afp_icdm.getitemdecimal(i,"traf_affected_adj_percent") * src_trf /
100) + adj_affect

        cell_interferer_count[src_cll_ind]++
        icdm_interferer_ind[src_cll_ind,cell_interferer_count[src_cll_ind]]
= int_cll_ind
        icdm_co[src_cll_ind,cell_interferer_count[src_cll_ind]] =
(d_afp_icdm.getitemdecimal(i,"traf_affected_co_percent") * src_trf /
100) + co_affect
        icdm_adj[src_cll_ind,cell_interferer_count[src_cll_ind]] =
(d_afp_icdm.getitemdecimal(i,"traf_affected_adj_percent") * src_trf /
100) + adj_affect
Next

//Cosite icinde icdm olcum yapmadiysa eksik costlari yukseltme
f_add_cost_in_cosite()

//Forbidden frekanslara cost eklenmesi - Fixed frekanslarin atanmasi ve
cost eklenmesi
f_add_cost_to_forbidden_fixed()
```

```
//---------------------Frekans Planlama------------------------------


//***************BCCH Bulma*****************

if cbx_bcch_plan.checked then

 last_best_cost = 1000000000
 For n = 1 to optimize_times

        log_text = string(today()) + " - " + string(now()) + " :  BCCH n =
"+ string(n)
        FileWrite(log_file , log_text)

        best_cost = 1000000000

        For freq_level = 1 to (cell_count - 1)

          log_text = string(today()) + " - " + string(now()) + " :  BCCH Freq
Level Left = "+ string(cell_count - freq_level)
          FileWrite(log_file , log_text)

          For random_bcch = 1 to random_bcch_count

                  if fixed_bcch_list[freq_level] <= 0 then

                          frequency_no = random_bcch_list[random_bcch]

                          if (cell_freq_bcch[freq_level] <> frequency_no)
then // Oncekinden Farkli Frekans ise

                                  if cell_freq_bcch[freq_level] > 0
then // Frekans degisimi

                                          For o = 1 to
cell_interfered_count[freq_level]

                                                  src_cll_ind =
icdm_interfered_ind[freq_level,o]
```

```
                                                co_affect =
icdm_co_int[freq_level,o]
                                                adj_affect =
icdm_adj_int[freq_level,o]


        freq_cost_table[src_cll_ind,frequency_no] += (co_affect * 10)
                                                if frequency_no > 1
then freq_cost_table[src_cll_ind,frequency_no - 1] += (adj_affect * 10)
                                                if frequency_no < 124
then freq_cost_table[src_cll_ind,frequency_no + 1] += (adj_affect * 10)



        freq_cost_table[src_cll_ind,cell_freq_bcch[freq_level]] -=
(co_affect * 10)
                                                if
cell_freq_bcch[freq_level] > 1 then
freq_cost_table[src_cll_ind,cell_freq_bcch[freq_level] - 1] -= (adj_affect
* 10)
                                                if
cell_freq_bcch[freq_level] < 124 then
freq_cost_table[src_cll_ind,cell_freq_bcch[freq_level] + 1] -= (adj_affect
* 10)


        flag_check_frequency[src_cll_ind] = true
                                                Next
                                        else // ilk frekans

                                                For o = 1 to
cell_interfered_count[freq_level]

                                                src_cll_ind =
icdm_interfered_ind[freq_level,o]

                                                co_affect =
icdm_co_int[freq_level,o]
```

```
                                                    adj_affect =
icdm_adj_int[freq_level,o]


        freq_cost_table[src_cll_ind,frequency_no] += (co_affect * 10)
                                                    if frequency_no > 1
then freq_cost_table[src_cll_ind,frequency_no - 1] += (adj_affect * 10)
                                                    if frequency_no < 124
then freq_cost_table[src_cll_ind,frequency_no + 1] += (adj_affect * 10)



        flag_check_frequency[src_cll_ind] = true

                                                Next

                                        end if

                                        cell_freq_bcch[freq_level] =
frequency_no
                        end if

                end if


                For i = (freq_level + 1) to cell_count
                if flag_check_frequency[i] = true then
                 flag_check_frequency[i] = false

                 if fixed_bcch_list[i] <= 0 then

                        frequency_no = f_get_best_bcch(i) //En iyi Frekans
Bul

                        if (cell_freq_bcch[i] <> frequency_no) then //
Oncekinden Farkli Frekans ise

                                        if cell_freq_bcch[i] > 0 then //
Frekans degisimi
```

```
                                        For o = 1 to
cell_interfered_count[i]
                                            src_cll_ind =
icdm_interfered_ind[i,o]
                                            co_affect =
icdm_co_int[i,o]
                                            adj_affect =
icdm_adj_int[i,o]

      freq_cost_table[src_cll_ind,frequency_no] += (co_affect * 10)
                                            if frequency_no > 1
then freq_cost_table[src_cll_ind,frequency_no - 1] += (adj_affect * 10)
                                            if frequency_no < 124
then freq_cost_table[src_cll_ind,frequency_no + 1] += (adj_affect * 10)

      freq_cost_table[src_cll_ind,cell_freq_bcch[i]] -= (co_affect * 10)
                                            if cell_freq_bcch[i] >
1 then freq_cost_table[src_cll_ind,cell_freq_bcch[i] - 1] -= (adj_affect *
10)
                                            if cell_freq_bcch[i] <
124 then freq_cost_table[src_cll_ind,cell_freq_bcch[i] + 1] -= (adj_affect
* 10)

                                        Next
                                    else // ilk Frekans

                                        For o = 1 to
cell_interfered_count[i]
                                            src_cll_ind =
icdm_interfered_ind[i,o]
                                            co_affect =
icdm_co_int[i,o]
```

```
                                                adj_affect =
icdm_adj_int[i,o]


        freq_cost_table[src_cll_ind,frequency_no] += (co_affect * 10)
                                                if frequency_no > 1
then freq_cost_table[src_cll_ind,frequency_no - 1] += (adj_affect * 10)
                                                if frequency_no < 124
then freq_cost_table[src_cll_ind,frequency_no + 1] += (adj_affect * 10)


                                                Next

                                        end if

                                        cell_freq_bcch[i] = frequency_no
                        end if

                 end if

                end if
                Next // end i



                result_cost = 0
                For i = 1 to cell_count //cost hesabi
                        result_cost += freq_cost_table[i,cell_freq_bcch[i]]
                Next

                if best_cost > result_cost then // Best Cost bulunmasi
                        best_cost = result_cost
                        best_cell_freq_bcch = cell_freq_bcch
                        For p = 1 to cell_count
                                For j = 1 to random_bcch_count

        temp_freq_cost_table[p,random_bcch_list[j]] =
freq_cost_table[p,random_bcch_list[j]]
                                Next
```

```
                        Next
                        log_text = string(today()) + " - " + string(now()) +
" :  BCCH Changing Best Cost = "+ string(Round(result_cost,3))
                        FileWrite(log_file , log_text)

                        result_cost = f_calculate_cost()
                        log_text = string(today()) + " - " + string(now()) +
" :  BCCH Real Best Cost = "+ string(Round(result_cost, 3))
                        FileWrite(log_file , log_text)

                end if

        Next // end random_bcch



        For p = 1 to cell_count
                For j = 1 to random_bcch_count
                        freq_cost_table[p,random_bcch_list[j]] =
temp_freq_cost_table[p,random_bcch_list[j]]
                Next
        Next

        cell_freq_bcch = best_cell_freq_bcch
        result_cost = best_cost

      Next // end freq_level

      if last_best_cost > best_cost then // Last Best Cost bulunmasi
        last_best_cost = best_cost
        last_best_cell_freq_bcch = best_cell_freq_bcch

        plan_file = FileOpen("c:\neptune\neptune_afp_last_plan.txt",
LineMode!, Write!, LockWrite!, Replace!)
        For p = 1 to cell_count
                For j = 1 to random_bcch_count
                        last_freq_cost_table[p,random_bcch_list[j]] =
temp_freq_cost_table[p,random_bcch_list[j]]
                Next
```

```
                plan_text = lb_cell_name.text(p) + Char(9) +
string(best_cell_freq_bcch[p])
                FileWrite(plan_file , plan_text)
        Next
        FileClose(plan_file)

        log_text = string(today()) + " - " + string(now()) + " :  BCCH
Changing Last Best Cost = "+ string(Round(best_cost,3))
        FileWrite(log_file , log_text)
        st_best_bcch_cost.text = string(Round(result_cost, 3))

        result_cost = f_calculate_cost()
        log_text = string(today()) + " - " + string(now()) + " :  BCCH Real
Last Best Cost = "+ string(Round(best_cost, 3))
        FileWrite(log_file , log_text)

      end if

      For p = 1 to cell_count
        flag_check_frequency[p] = true
      Next

 Next // end n

 For p = 1 to cell_count
      For j = 1 to random_bcch_count
        freq_cost_table[p,random_bcch_list[j]] =
last_freq_cost_table[p,random_bcch_list[j]]
      Next
 Next

 cell_freq_bcch = last_best_cell_freq_bcch
 result_cost = last_best_cost

 For i = 1 to cell_count
      if fixed_bcch_list[i] <= 0 then
        freq_cost_table[i,cell_freq_bcch[i]] += 100000
```

```
          if cell_freq_bcch[i] > 1 then freq_cost_table[i,cell_freq_bcch[i] -
1] += 100000
          if cell_freq_bcch[i] < 124 then freq_cost_table[i,cell_freq_bcch[i]
+ 1] += 100000
        end if
 Next

 For i = 1 to 4000
        flag_check_frequency[i] = true
 Next

end if


//***************TCH Bulma*********************

if cbx_tch_plan.checked then

 last_best_cost = 1000000000
 For n = 1 to optimize_times

        log_text = string(today()) + " - " + string(now()) + " :  TCH n = "+
string(n)
        FileWrite(log_file , log_text)

        For p = 1 to 4000
          For j = 1 to 16
                  best_cell_freq_tch[p,j] = cell_freq_tch[p,j]
          Next
        Next

        best_cost = 1000000000

        For freq_level = 1 to (cell_count - 1)

          log_text = string(today()) + " - " + string(now()) + " :  TCH Freq
Level Left = "+ string(cell_count - freq_level)
          FileWrite(log_file , log_text)
```

```
For k = 1 to cell_tch_number[freq_level]

For random_tch = 1 to random_tch_count

if fixed_tch_list[freq_level,k] <= 0 then

frequency_no =
random_tch_list[random_tch]

if (cell_freq_tch[freq_level,k] <>
frequency_no) then // Oncekinden Farkli Frekans ise

if
cell_freq_tch[freq_level,k] > 0 then // Frekans degisimi

For o = 1 to
cell_interfered_count[freq_level]


src_cll_ind = icdm_interfered_ind[freq_level,o]


co_affect = icdm_co_int[freq_level,o]

adj_affect = icdm_adj_int[freq_level,o]


freq_cost_table[src_cll_ind,frequency_no] += (co_affect)
if
frequency_no > 1 then freq_cost_table[src_cll_ind,frequency_no - 1] +=
(adj_affect)
if
frequency_no < 124 then freq_cost_table[src_cll_ind,frequency_no + 1]
+= (adj_affect)



freq_cost_table[src_cll_ind,cell_freq_tch[freq_level,k]] -=
(co_affect)
```

```
                                                                    if
cell_freq_tch[freq_level,k] > 1 then
freq_cost_table[src_cll_ind,cell_freq_tch[freq_level,k] - 1] -=
(adj_affect)
                                                                    if
cell_freq_tch[freq_level,k] < 124 then
freq_cost_table[src_cll_ind,cell_freq_tch[freq_level,k] + 1] -=
(adj_affect)


        flag_check_frequency[src_cll_ind] = true
                                                            Next


        freq_cost_table[freq_level,cell_freq_tch[freq_level,k]] -= 100000
                                                            if
cell_freq_tch[freq_level,k] > 1 then
freq_cost_table[freq_level,cell_freq_tch[freq_level,k] - 1] -= 100000
                                                            if
cell_freq_tch[freq_level,k] < 124 then
freq_cost_table[freq_level,cell_freq_tch[freq_level,k] + 1] -= 100000

                                                    else // ilk frekans

                                                            For o = 1 to
cell_interfered_count[freq_level]


        src_cll_ind = icdm_interfered_ind[freq_level,o]


        co_affect = icdm_co_int[freq_level,o]

        adj_affect = icdm_adj_int[freq_level,o]


        freq_cost_table[src_cll_ind,frequency_no] += (co_affect)
```

```
                                                                    if
frequency_no > 1 then freq_cost_table[src_cll_ind,frequency_no - 1] +=
(adj_affect)
                                                                    if
frequency_no < 124 then freq_cost_table[src_cll_ind,frequency_no + 1]
+= (adj_affect)


        flag_check_frequency[src_cll_ind] = true
                                                            Next

                                                        end if


        cell_freq_tch[freq_level,k] = frequency_no

        freq_cost_table[freq_level,frequency_no] += 100000
                                                        if frequency_no > 1
then freq_cost_table[freq_level,frequency_no - 1] += 100000
                                                        if frequency_no < 124
then freq_cost_table[freq_level,frequency_no + 1] += 100000

                                    end if

                                end if

                                For m = (k+1) to
cell_tch_number[freq_level]
                                    if fixed_tch_list[freq_level,m] <= 0
then

                                            frequency_no =
f_get_best_tch(freq_level) //En iyi Frekans Bul

                                            if
(cell_freq_tch[freq_level,m] <> frequency_no) then // Oncekinden Farkli
Frekans ise
```

if cell_freq_tch[freq_level,m] > 0 then // Frekans degisimi

For o = 1 to cell_interfered_count[freq_level]

src_cll_ind = icdm_interfered_ind[freq_level,o]

co_affect = icdm_co_int[freq_level,o]

adj_affect = icdm_adj_int[freq_level,o]

freq_cost_table[src_cll_ind,frequency_no] += (co_affect)

if frequency_no > 1 then freq_cost_table[src_cll_ind,frequency_no - 1] += (adj_affect)

if frequency_no < 124 then freq_cost_table[src_cll_ind,frequency_no + 1] += (adj_affect)

freq_cost_table[src_cll_ind,cell_freq_tch[freq_level,m]] -= (co_affect)

if cell_freq_tch[freq_level,m] > 1 then freq_cost_table[src_cll_ind,cell_freq_tch[freq_level,m] - 1] -= (adj_affect)

if cell_freq_tch[freq_level,m] < 124 then freq_cost_table[src_cll_ind,cell_freq_tch[freq_level,m] + 1] -= (adj_affect)

Next

freq_cost_table[freq_level,cell_freq_tch[freq_level,m]] -= 100000

```
                                                                    if
cell_freq_tch[freq_level,m] > 1 then
freq_cost_table[freq_level,cell_freq_tch[freq_level,m] - 1] -= 100000
                                                                    if
cell_freq_tch[freq_level,m] < 124 then
freq_cost_table[freq_level,cell_freq_tch[freq_level,m] + 1] -= 100000

                                                          else // ilk
Frekans

                                                                    For o =
1 to cell_interfered_count[freq_level]


        src_cll_ind = icdm_interfered_ind[freq_level,o]


        co_affect = icdm_co_int[freq_level,o]

        adj_affect = icdm_adj_int[freq_level,o]


        freq_cost_table[src_cll_ind,frequency_no] += (co_affect)

        if frequency_no > 1 then freq_cost_table[src_cll_ind,frequency_no
- 1] += (adj_affect)

        if frequency_no < 124 then
freq_cost_table[src_cll_ind,frequency_no + 1] += (adj_affect)


                                                                    Next

                                                          end if



        cell_freq_tch[freq_level,m] = frequency_no
```

```
        freq_cost_table[freq_level,frequency_no] += 100000
                                                        if
frequency_no > 1 then freq_cost_table[freq_level,frequency_no - 1] +=
100000
                                                        if
frequency_no < 124 then freq_cost_table[freq_level,frequency_no + 1]
+= 100000

                                            end if

                                        end if

                                    Next // end m


                                    For i = (freq_level + 1) to cell_count
                                     if flag_check_frequency[i] = true then
                                            flag_check_frequency[i] = false

                                            For m = 1 to cell_tch_number[i]
                                             if fixed_tch_list[i,m] <= 0 then

                                                    frequency_no =
f_get_best_tch(i) //En iyi Frekans Bul

                                                    if (cell_freq_tch[i,m] <>
frequency_no) then // Oncekinden Farkli Frekans ise

                                                                    if
cell_freq_tch[i,m] > 0 then // Frekans degisimi

                                                                            For o =
1 to cell_interfered_count[i]


        src_cll_ind = icdm_interfered_ind[i,o]
```

```
        co_affect = icdm_co_int[i,o]

        adj_affect = icdm_adj_int[i,o]


        freq_cost_table[src_cll_ind,frequency_no] += (co_affect)

        if frequency_no > 1 then freq_cost_table[src_cll_ind,frequency_no
- 1] += (adj_affect)

        if frequency_no < 124 then
freq_cost_table[src_cll_ind,frequency_no + 1] += (adj_affect)


        freq_cost_table[src_cll_ind,cell_freq_tch[i,m]] -= (co_affect)

        if cell_freq_tch[i,m] > 1 then
freq_cost_table[src_cll_ind,cell_freq_tch[i,m] - 1] -= (adj_affect)

        if cell_freq_tch[i,m] < 124 then
freq_cost_table[src_cll_ind,cell_freq_tch[i,m] + 1] -= (adj_affect)

                                                                Next


        freq_cost_table[i,cell_freq_tch[i,m]] -= 100000
                                                                if
cell_freq_tch[i,m] > 1 then freq_cost_table[i,cell_freq_tch[i,m] - 1] -=
100000
                                                                if
cell_freq_tch[i,m] < 124 then freq_cost_table[i,cell_freq_tch[i,m] + 1] -=
100000

                                                        else // ilk
Frekans

                                                                For o =
1 to cell_interfered_count[i]
```

```
        src_cll_ind = icdm_interfered_ind[i,o]


        co_affect = icdm_co_int[i,o]

        adj_affect = icdm_adj_int[i,o]


        freq_cost_table[src_cll_ind,frequency_no] += (co_affect)

        if frequency_no > 1 then freq_cost_table[src_cll_ind,frequency_no
- 1] += (adj_affect)

        if frequency_no < 124 then
freq_cost_table[src_cll_ind,frequency_no + 1] += (adj_affect)

                                                                Next

                                                            end if



        cell_freq_tch[i,m] = frequency_no

        freq_cost_table[i,frequency_no] += 100000
                                                            if
frequency_no > 1 then freq_cost_table[i,frequency_no - 1] += 100000
                                                            if
frequency_no < 124 then freq_cost_table[i,frequency_no + 1] += 100000

                                            end if

                                        end if
                                    Next // end m
                                end if
                            Next // end i
```

```
                                result_cost = 0
                                For i = 1 to cell_count //cost hesabi
                                        For j = 1 to cell_tch_number[i]
                                                result_cost = result_cost +
freq_cost_table[i,cell_freq_tch[i,j]] - 100000
                                        Next
                                Next

                                if best_cost > result_cost then // Best Cost
bulunmasi
                                        best_cost = result_cost
                                        For p = 1 to cell_count
                                                For j = 1 to
cell_tch_number[p]

        best_cell_freq_tch[p,j] = cell_freq_tch[p,j]
                                                Next
                                        Next
                                        For p = 1 to cell_count
                                                For j = 1 to
random_tch_count

        temp_freq_cost_table[p,random_tch_list[j]] =
freq_cost_table[p,random_tch_list[j]]
                                                Next
                                        Next

                                        log_text = string(today()) + " - " +
string(now()) + " :  TCH Changing Best Cost = "+
string(Round(result_cost,3))
                                        FileWrite(log_file , log_text)

                                end if

                        Next // end random_tch

                        For p = 1 to cell_count
                                For j = 1 to random_tch_count
```

```
        freq_cost_table[p,random_tch_list[j]] =
temp_freq_cost_table[p,random_tch_list[j]]
                                Next
                        Next

                        For p = 1 to cell_count
                                For j = 1 to cell_tch_number[p]
                                        cell_freq_tch[p,j] =
best_cell_freq_tch[p,j]
                                Next
                        Next
                        result_cost = best_cost

                Next // end k

        Next // end freq_level

        if last_best_cost > best_cost then // Last Best Cost bulunmasi
          last_best_cost = best_cost

          plan_file = FileOpen("c:\neptune\neptune_afp_last_plan.txt",
LineMode!, Write!, LockWrite!, Replace!)
          For p = 1 to cell_count
                  plan_text = lb_cell_name.text(p)
                  if cbx_bcch_plan.checked then plan_text = plan_text +
Char(9) + string(cell_freq_bcch[p])

                  For j = 1 to cell_tch_number[p]
                          last_best_cell_freq_tch[p,j] =
best_cell_freq_tch[p,j]
                          plan_text = plan_text + Char(9) +
string(best_cell_freq_tch[p,j])
                  Next

                  FileWrite(plan_file , plan_text)
          Next
          FileClose(plan_file)
```

```
        For p = 1 to cell_count
                For j = 1 to random_tch_count
                        last_freq_cost_table[p,random_tch_list[j]] =
temp_freq_cost_table[p,random_tch_list[j]]
                Next
        Next

        log_text = string(today()) + " - " + string(now()) + " :  TCH
Changing Last Best Cost = "+ string(Round(best_cost,3))
        FileWrite(log_file , log_text)
        st_best_tch_cost.text = string(Round(best_cost, 3))

       end if

       For p = 1 to cell_count
         flag_check_frequency[p] = true
       Next

 Next // end n

 For p = 1 to cell_count
        For j = 1 to random_tch_count
                freq_cost_table[p,random_tch_list[j]] =
last_freq_cost_table[p,random_tch_list[j]]
        Next
 Next
 For p = 1 to cell_count
        For j = 1 to cell_tch_number[p]
                cell_freq_tch[p,j] = last_best_cell_freq_tch[p,j]
        Next
 Next
 result_cost = last_best_cost

end if

//Cosite Cost larini temizleme
f_clean_cosite_costs()
```

```
//-------------- Sonuc Cost larin hesaplanmasi-----------------------------
total_cost = 0

if cbx_bcch_plan.checked = true then
        result_cost = 0
        For p = 1 to cell_count
          result_cost += freq_cost_table[p,cell_freq_bcch[p]]
        Next

        st_best_bcch_cost.text = string(Round(result_cost, 3))

        log_text = string(today()) + " - " + string(now()) + " :  BCCH
Result Cost = "+ string(Round(result_cost, 3))
        FileWrite(log_file , log_text)

        total_cost += result_cost
end if

if cbx_tch_plan.checked = true then
        result_cost = 0
        For p = 1 to cell_count
          For j = 1 to cell_tch_number[p]
                result_cost += freq_cost_table[p,cell_freq_tch[p,j]]
          Next
        Next

        st_best_tch_cost.text = string(Round(result_cost, 3))

        log_text = string(today()) + " - " + string(now()) + " :  TCH Result
Cost = "+ string(Round(result_cost, 3))
        FileWrite(log_file , log_text)

        total_cost += result_cost
end if

log_text = string(today()) + " - " + string(now()) + " :  Total Result Cost
= "+ string(Round(total_cost, 3))
FileWrite(log_file , log_text)
```

```
st_best_cost.text = string(Round(total_cost, 3))

result_cost = f_calculate_cost()
st_status.text = "Real Cost : " + string(Round(result_cost, 3))
log_text = string(today()) + " - " + string(now()) + " :  Real Total Cost =
"+ string(Round(result_cost, 3))
FileWrite(log_file , log_text)

log_text = string(today()) + " - " + string(now()) + " :  Planning finished!"
FileWrite(log_file , log_text)

For i = 1 to cell_count
        d_afp_cell_frequencies_best.insertrow(i)
        d_afp_cell_frequencies_best.setitem(i,"cell_name",lb_cell_name.te
xt(i))
        tru_number = cell_tch_number[i] + 1
        d_afp_cell_frequencies_best.setitem(i,"tru_number",tru_number)

        log_text = lb_cell_name.text(i)

        if cbx_bcch_plan.checked = true then

        d_afp_cell_frequencies_best.setitem(i,"bcch_cost",freq_cost_table[
i,cell_freq_bcch[i]])
          d_afp_cell_frequencies_best.setitem(i,"bcch",cell_freq_bcch[i])
          log_text = log_text + Char(9) + string(cell_freq_bcch[i])
        end if



        result_cost = 0
        For j = 1 to cell_tch_number[i]
          if cbx_tch_plan.checked = true then
                  result_cost += freq_cost_table[i,cell_freq_tch[i,j]]

        d_afp_cell_frequencies_best.setitem(i,("tch"+string(j)),cell_freq_tc
h[i,j])
          end if
          log_text = log_text + Char(9) + string(cell_freq_tch[i,j])
        Next
```

```
        if cbx_tch_plan.checked = true then
d_afp_cell_frequencies_best.setitem(i,"tch_cost",result_cost)
        total_cost = result_cost
        if cbx_bcch_plan.checked = true then total_cost +=
freq_cost_table[i,cell_freq_bcch[i]]
        d_afp_cell_frequencies_best.setitem(i,"total_cost",total_cost)

        FileWrite(log_file , log_text)
Next

//Best plan check etme
f_check_new_plan()

//---------------------------Bitirme Islemleri-------------------------
FileClose(log_file)
d_afp_icdm.SetRedraw(true)
d_afp_cells.SetRedraw(true)
cb_save_best.enabled = true
cb_check_plan.enabled = true
st_status.text += " - Planing finished!"
enabled = true
SetPointer(Arrow!)
end event

type lb_all_frequencies from listbox within w_afp
int X=73
int Y=1440
int Width=357
int Height=628
int TabOrder=170
boolean BringToTop=true
BorderStyle BorderStyle=StyleLowered!
boolean VScrollBar=true
boolean MultiSelect=true
long TextColor=33554432
long BackColor=15793151
int TextSize=-8
int Weight=400
string FaceName="Arial TUR"
```

FontCharSet FontCharSet=TurkishCharSet!
FontFamily FontFamily=Swiss!
FontPitch FontPitch=Variable!
end type

type st_all_freq from statictext within w_afp
int X=32
int Y=1344
int Width=503
int Height=80
boolean Enabled=false
boolean BringToTop=true
string Text="All Frequencies"
boolean FocusRectangle=false
long TextColor=33554432
long BackColor=82042848
int TextSize=-10
int Weight=700
string FaceName="Arial TUR"
FontCharSet FontCharSet=TurkishCharSet!
FontFamily FontFamily=Swiss!
FontPitch FontPitch=Variable!
end type
type lb_bcch_frequencies from listbox within w_afp
int X=800
int Y=1440
int Width=357
int Height=628
int TabOrder=190
boolean BringToTop=true
BorderStyle BorderStyle=StyleLowered!
boolean VScrollBar=true
boolean MultiSelect=true
long TextColor=33554432
long BackColor=12639424
int TextSize=-8
int Weight=400
string FaceName="Arial TUR"
FontCharSet FontCharSet=TurkishCharSet!

```
FontFamily FontFamily=Swiss!
FontPitch FontPitch=Variable!
end type

type cb_add_bcch from commandbutton within w_afp
int X=471
int Y=1580
int Width=288
int Height=144
int TabOrder=220
string Text="Add =>"
int TextSize=-8
int Weight=400
string FaceName="Arial TUR"
FontCharSet FontCharSet=TurkishCharSet!
FontFamily FontFamily=Swiss!
FontPitch FontPitch=Variable!
end type

event clicked;integer li_ItemTotal, li_ItemCount

li_ItemTotal = lb_all_frequencies.TotalItems( )

FOR li_ItemCount = 1 to li_ItemTotal


        IF lb_all_frequencies.State(li_ItemCount) = 1 THEN

        lb_bcch_frequencies.additem(lb_all_frequencies.text(li_ItemCount)
)
          bcch_frequency_list[long(lb_all_frequencies.text(li_ItemCount))]
= true
          lb_all_frequencies.deleteitem(li_ItemCount)
          li_ItemCount = li_ItemCount - 1
        END IF
NEXT
end event
type st_bcch_freq from statictext within w_afp
int X=722
```

```
int Y=1344
int Width=553
int Height=80
boolean Enabled=false
boolean BringToTop=true
string Text="BCCH Frequencies"
boolean FocusRectangle=false
long TextColor=33554432
long BackColor=82042848
int TextSize=-10
int Weight=700
string FaceName="Arial TUR"
FontCharSet FontCharSet=TurkishCharSet!
FontFamily FontFamily=Swiss!
FontPitch FontPitch=Variable!
end type

type cb_remove_bcch from commandbutton within w_afp
int X=471
int Y=1772
int Width=288
int Height=144
int TabOrder=230
string Text="<= Remove"
int TextSize=-8
int Weight=400
string FaceName="Arial TUR"
FontCharSet FontCharSet=TurkishCharSet!
FontFamily FontFamily=Swiss!
FontPitch FontPitch=Variable!
end type

event clicked;integer li_ItemTotal, li_ItemCount
long i,j

// Listedekilerin sayisini bul
li_ItemTotal = lb_bcch_frequencies.TotalItems( )

FOR li_ItemCount = 1 to li_ItemTotal
```

```
// Kaynak listedeki secildiyse Hedef listeye gonder
        IF lb_bcch_frequencies.State(li_ItemCount) = 1 THEN

        lb_all_frequencies.additem(lb_bcch_frequencies.text(li_ItemCount)
)

        bcch_frequency_list[long(lb_bcch_frequencies.text(li_ItemCount))
] = false
          lb_bcch_frequencies.deleteitem(li_ItemCount)
          li_ItemCount = li_ItemCount - 1
        END IF

NEXT
end event

type st_all_freq_3 from statictext within w_afp
int X=32
int Y=2152
int Width=503
int Height=80
boolean Enabled=false
boolean BringToTop=true
string Text="All Frequencies"
boolean FocusRectangle=false
long TextColor=33554432
long BackColor=82042848
int TextSize=-10
int Weight=700
string FaceName="Arial TUR"
FontCharSet FontCharSet=TurkishCharSet!
FontFamily FontFamily=Swiss!
FontPitch FontPitch=Variable!
end type

type lb_all_frequencies_3 from listbox within w_afp
int X=73
int Y=2248
int Width=357
```

int Height=628
int TabOrder=200
boolean BringToTop=true
BorderStyle BorderStyle=StyleLowered!
boolean VScrollBar=true
boolean MultiSelect=true
long TextColor=33554432
long BackColor=15793151
int TextSize=-8
int Weight=400
string FaceName="Arial TUR"
FontCharSet FontCharSet=TurkishCharSet!
FontFamily FontFamily=Swiss!
FontPitch FontPitch=Variable!
end type

type st_tch_freq from statictext within w_afp
int X=722
int Y=2152
int Width=553
int Height=80
boolean Enabled=false
boolean BringToTop=true
string Text="TCH Frequencies"
boolean FocusRectangle=false
long TextColor=33554432
long BackColor=82042848
int TextSize=-10
int Weight=700
string FaceName="Arial TUR"
FontCharSet FontCharSet=TurkishCharSet!
FontFamily FontFamily=Swiss!
FontPitch FontPitch=Variable!
end type

type lb_tch_frequencies from listbox within w_afp
int X=800
int Y=2248
int Width=357

```
int Height=628
int TabOrder=330
boolean BringToTop=true
BorderStyle BorderStyle=StyleLowered!
boolean VScrollBar=true
boolean MultiSelect=true
long TextColor=33554432
long BackColor=12639424
int TextSize=-8
int Weight=400
string FaceName="Arial TUR"
FontCharSet FontCharSet=TurkishCharSet!
FontFamily FontFamily=Swiss!
FontPitch FontPitch=Variable!
end type

type cb_add_tch from commandbutton within w_afp
int X=471
int Y=2388
int Width=288
int Height=144
int TabOrder=250
string Text="Add =>"
int TextSize=-8
int Weight=400
string FaceName="Arial TUR"
FontCharSet FontCharSet=TurkishCharSet!
FontFamily FontFamily=Swiss!
FontPitch FontPitch=Variable!
end type

event clicked;integer li_ItemTotal, li_ItemCount


li_ItemTotal = lb_all_frequencies_3.TotalItems( )

FOR li_ItemCount = 1 to li_ItemTotal
```

```
        IF lb_all_frequencies_3.State(li_ItemCount) = 1 THEN

        lb_tch_frequencies.additem(lb_all_frequencies_3.text(li_ItemCoun
t))

        tch_frequency_list[long(lb_all_frequencies_3.text(li_ItemCount))]
= true
          lb_all_frequencies_3.deleteitem(li_ItemCount)
          li_ItemCount = li_ItemCount - 1
        END IF
NEXT
end event

type cb_remove_tch from commandbutton within w_afp
int X=471
int Y=2580
int Width=288
int Height=144
int TabOrder=260
string Text="<= Remove"
int TextSize=-8
int Weight=400
string FaceName="Arial TUR"
FontCharSet FontCharSet=TurkishCharSet!
FontFamily FontFamily=Swiss!
FontPitch FontPitch=Variable!
end type

event clicked;integer li_ItemTotal, li_ItemCount

// Listedekilerin sayisini bul
li_ItemTotal = lb_tch_frequencies.TotalItems( )

FOR li_ItemCount = 1 to li_ItemTotal

// Kaynak listedeki secildiyse Hedef listeye gonder
        IF lb_tch_frequencies.State(li_ItemCount) = 1 THEN
```

```
        lb_all_frequencies_3.additem(lb_tch_frequencies.text(li_ItemCoun
t))
          tch_frequency_list[long(lb_tch_frequencies.text(li_ItemCount))]
= false
          lb_tch_frequencies.deleteitem(li_ItemCount)
          li_ItemCount = li_ItemCount - 1
        END IF

NEXT
end event

type d_afp_cell_frequencies from datawindow_common within w_afp
int X=2546
int Y=996
int Width=782
int Height=464
int TabOrder=60
boolean Visible=false
boolean BringToTop=true
string DataObject="dwe_afp_cell_frequencies_best"
boolean TitleBar=false
BorderStyle BorderStyle=StyleBox!
boolean ControlMenu=false
boolean MinBox=false
boolean MaxBox=false
boolean HScrollBar=false
boolean VScrollBar=false
boolean Resizable=false
boolean LiveScroll=false
end type

type st_initial_cost from statictext within w_afp
int X=2976
int Y=1896
int Width=571
int Height=80
boolean Enabled=false
boolean BringToTop=true
```

boolean Border=true
Alignment Alignment=Right!
boolean FocusRectangle=false
long TextColor=33554432
long BackColor=15793151
int TextSize=-10
int Weight=700
string FaceName="Arial TUR"
FontCharSet FontCharSet=TurkishCharSet!
FontFamily FontFamily=Swiss!
FontPitch FontPitch=Variable!
end type

type st_initial_cost_label from statictext within w_afp
int X=2478
int Y=1904
int Width=480
int Height=80
boolean Enabled=false
boolean BringToTop=true
string Text="Initial Cost Total"
Alignment Alignment=Right!
boolean FocusRectangle=false
long TextColor=33554432
long BackColor=82042848
int TextSize=-10
int Weight=700
string FaceName="Arial TUR"
FontCharSet FontCharSet=TurkishCharSet!
FontFamily FontFamily=Swiss!
FontPitch FontPitch=Variable!
end type

type cb_optimize from commandbutton within w_afp
int X=2917
int Y=1640
int Width=631
int Height=80
int TabOrder=70

```
boolean Enabled=false
boolean BringToTop=true
string Text="Optimize Imported Plan"
int TextSize=-8
int Weight=400
string FaceName="Arial TUR"
FontCharSet FontCharSet=TurkishCharSet!
FontFamily FontFamily=Swiss!
FontPitch FontPitch=Variable!
end type

event clicked;decimal result_cost,best_cost,total_cost,bcch_cost
string nul
long optimize_times,i,o,m,p,j,k,n
long frequency_no,src_cll_ind
decimal co_affect,adj_affect


enabled = false
SetPointer(HourGlass!)

setnull(nul)
st_status.text = nul

// Best Cost varsa kaldigi yerden devam etsin----------------------------------
------------------------------
if isnull(st_best_cost.text) and isnull(st_best_bcch_cost.text) and
isnull(st_best_tch_cost.text) then

        // Initial Cost un best lere kopyalanmasi
        if cbx_tch_plan.checked and cbx_bcch_plan.checked then
          best_cost = Dec(st_initial_cost.text)
        elseif cbx_bcch_plan.checked then
          best_cost = Dec(st_initial_bcch_cost.text)
        elseif cbx_tch_plan.checked then
          best_cost = Dec(st_initial_tch_cost.text)
        else
          messagebox("Warning","TCH, BCCH or both checkboxes must
be selected")
```

```
          return
        end if

end if


if cbx_bcch_plan.checked then
        best_cell_freq_bcch = cell_freq_bcch
        For p = 1 to cell_count
          For j = 1 to random_bcch_count
                  temp_freq_cost_table[p,random_bcch_list[j]] =
freq_cost_table[p,random_bcch_list[j]]
          Next
        Next
end if

if cbx_tch_plan.checked then
        For p = 1 to cell_count
          For j = 1 to cell_tch_number[p]
                  best_cell_freq_tch[p,j] = cell_freq_tch[p,j]
          Next
        Next
        For p = 1 to cell_count
          For j = 1 to random_tch_count
                  temp_freq_cost_table[p,random_tch_list[j]] =
freq_cost_table[p,random_tch_list[j]]
          Next
        Next
end if

// Optimizasyona baslanmasi----------------------------------------------------
-----------------

if len(sle_optimize_times.text) > 0 then
        optimize_times = long(sle_optimize_times.text)
else
        optimize_times = 1
end if
```

```
For n = 1 to optimize_times

        st_status.text = "Wait for Optimizing! (Remaining times:" +
string(optimize_times - n + 1) + ")"

        //Cosite Cost larini temizleme
        For i = 1 to cell_count

          if cbx_bcch_plan.checked then
                  if fixed_bcch_list[i] <= 0 then
                          freq_cost_table[i,cell_freq_bcch[i]] -= 100000
                          if cell_freq_bcch[i] > 1 then
freq_cost_table[i,cell_freq_bcch[i] - 1] -= 100000
                          if cell_freq_bcch[i] < 124 then
freq_cost_table[i,cell_freq_bcch[i] + 1] -= 100000
                  end if
          end if

          if cbx_tch_plan.checked then
                  For k = 1 to cell_tch_number[i]
                          if fixed_tch_list[i,k] <= 0 then
                                  freq_cost_table[i,cell_freq_tch[i,k]] -=
100000
                                  if cell_freq_tch[i,k] > 1 then
freq_cost_table[i,cell_freq_tch[i,k] - 1] -= 100000
                                  if cell_freq_tch[i,k] < 124 then
freq_cost_table[i,cell_freq_tch[i,k] + 1] -= 100000
                          end if
                  Next
          end if

        Next

        //BCCH
        if cbx_bcch_plan.checked then
        For i = 1 to cell_count

                  if fixed_bcch_list[i] <= 0 then
```

```
                    frequency_no = f_get_best_bcch(i) //En iyi Frekans
Bul

                    if (cell_freq_bcch[i] <> frequency_no) then //
Oncekinden Farkli Frekans ise

                        if cell_freq_bcch[i] > 0 then //
Frekans degisimi

                            For o = 1 to
cell_interfered_count[i]

                                src_cll_ind =
icdm_interfered_ind[i,o]

                                co_affect =
icdm_co_int[i,o]
                                adj_affect =
icdm_adj_int[i,o]


      freq_cost_table[src_cll_ind,frequency_no] += (co_affect * 10)
                                if frequency_no > 1
then freq_cost_table[src_cll_ind,frequency_no - 1] += (adj_affect * 10)
                                if frequency_no < 124
then freq_cost_table[src_cll_ind,frequency_no + 1] += (adj_affect * 10)


      freq_cost_table[src_cll_ind,cell_freq_bcch[i]] -= (co_affect * 10)
                                if cell_freq_bcch[i] >
1 then freq_cost_table[src_cll_ind,cell_freq_bcch[i] - 1] -= (adj_affect *
10)
                                if cell_freq_bcch[i] <
124 then freq_cost_table[src_cll_ind,cell_freq_bcch[i] + 1] -= (adj_affect
* 10)

                            Next
                        else // ilk Frekans
```

```
                                                For o = 1 to
cell_interfered_count[i]

                                                    src_cll_ind =
icdm_interfered_ind[i,o]

                                                    co_affect =
icdm_co_int[i,o]
                                                    adj_affect =
icdm_adj_int[i,o]


        freq_cost_table[src_cll_ind,frequency_no] += (co_affect * 10)
                                                    if frequency_no > 1
then freq_cost_table[src_cll_ind,frequency_no - 1] += (adj_affect * 10)
                                                    if frequency_no < 124
then freq_cost_table[src_cll_ind,frequency_no + 1] += (adj_affect * 10)


                                                Next

                                            end if

                                            cell_freq_bcch[i] = frequency_no
                                end if

                        end if
        Next

        For i = 1 to cell_count
          if fixed_bcch_list[i] <= 0 then
                        freq_cost_table[i,cell_freq_bcch[i]] += 100000
                        if cell_freq_bcch[i] > 1 then
freq_cost_table[i,cell_freq_bcch[i] - 1] += 100000
                        if cell_freq_bcch[i] < 124 then
freq_cost_table[i,cell_freq_bcch[i] + 1] += 100000
          end if
        Next
```

```
end if


//TCH
if cbx_tch_plan.checked then
For i = 1 to cell_count

        For m = 1 to cell_tch_number[i]

            if fixed_tch_list[i,m] <= 0 then

                frequency_no = f_get_best_tch(i) //En iyi
Frekans Bul

                if (cell_freq_tch[i,m] <> frequency_no)
then // Oncekinden Farkli Frekans ise

                    if cell_freq_tch[i,m] > 0 then
// Frekans degisimi

                        For o = 1 to
cell_interfered_count[i]

                            src_cll_ind =
icdm_interfered_ind[i,o]

                            co_affect =
icdm_co_int[i,o]
                            adj_affect =
icdm_adj_int[i,o]


freq_cost_table[src_cll_ind,frequency_no] += (co_affect)
                            if
frequency_no > 1 then freq_cost_table[src_cll_ind,frequency_no - 1] +=
(adj_affect)
                            if
frequency_no < 124 then freq_cost_table[src_cll_ind,frequency_no + 1]
+= (adj_affect)
```

```
        freq_cost_table[src_cll_ind,cell_freq_tch[i,m]] -= (co_affect)
                                                                    if
cell_freq_tch[i,m] > 1 then freq_cost_table[src_cll_ind,cell_freq_tch[i,m]
- 1] -= (adj_affect)
                                                                    if
cell_freq_tch[i,m] < 124 then
freq_cost_table[src_cll_ind,cell_freq_tch[i,m] + 1] -= (adj_affect)

                                                        Next


                                                  else // ilk Frekans

                                                        For o = 1 to
cell_interfered_count[i]

                                                              src_cll_ind =
icdm_interfered_ind[i,o]

                                                              co_affect =
icdm_co_int[i,o]
                                                              adj_affect =
icdm_adj_int[i,o]


        freq_cost_table[src_cll_ind,frequency_no] += (co_affect)
                                                                    if
frequency_no > 1 then freq_cost_table[src_cll_ind,frequency_no - 1] +=
(adj_affect)
                                                                    if
frequency_no < 124 then freq_cost_table[src_cll_ind,frequency_no + 1]
+= (adj_affect)


                                                        Next

                                                  end if
```

```
                                                cell_freq_tch[i,m] =
frequency_no

                                    end if

                                    freq_cost_table[i,frequency_no] += 100000
                                    if frequency_no > 1 then
freq_cost_table[i,frequency_no - 1] += 100000
                                    if frequency_no < 124 then
freq_cost_table[i,frequency_no + 1] += 100000


                              end if
                        Next

        Next
        end if


        //Cost hesabi
        if cbx_bcch_plan.checked then
          result_cost = 0
          For i = 1 to cell_count
                    result_cost = result_cost +
(freq_cost_table[i,cell_freq_bcch[i]] - 100000)
          Next
          bcch_cost = result_cost
        else
          bcch_cost = 0
        end if

        if cbx_tch_plan.checked then
          result_cost = 0
          For i = 1 to cell_count
                    For j = 1 to cell_tch_number[i]
                              result_cost = result_cost +
(freq_cost_table[i,cell_freq_tch[i,j]] - 100000)
                    Next
```

```
        Next
      else
        result_cost = 0
      end if

      total_cost = bcch_cost + result_cost


      if (best_cost > total_cost) then

        best_cost = total_cost

        if cbx_bcch_plan.checked and cbx_tch_plan.checked then

                st_best_bcch_cost.text = string(Round(bcch_cost, 3))
                st_best_tch_cost.text = string(Round(result_cost, 3))
                st_best_cost.text = string(Round(total_cost, 3))

                best_cell_freq_bcch = cell_freq_bcch
                For p = 1 to cell_count
                        For j = 1 to random_bcch_count

      temp_freq_cost_table[p,random_bcch_list[j]] =
freq_cost_table[p,random_bcch_list[j]]
                        Next
                Next

                For p = 1 to cell_count
                        For j = 1 to cell_tch_number[p]
                                best_cell_freq_tch[p,j] = cell_freq_tch[p,j]
                        Next
                Next
                For p = 1 to cell_count
                        For j = 1 to random_tch_count
                                temp_freq_cost_table[p,random_tch_list[j]]
= freq_cost_table[p,random_tch_list[j]]
                        Next
                Next
```

```
        elseif cbx_bcch_plan.checked then
                st_best_bcch_cost.text = string(Round(total_cost, 3))

                best_cell_freq_bcch = cell_freq_bcch
                For p = 1 to cell_count
                        For j = 1 to random_bcch_count

      temp_freq_cost_table[p,random_bcch_list[j]] =
freq_cost_table[p,random_bcch_list[j]]
                        Next
                Next

        elseif cbx_tch_plan.checked then
                st_best_tch_cost.text = string(Round(total_cost, 3))

                For p = 1 to cell_count
                        For j = 1 to cell_tch_number[p]
                                best_cell_freq_tch[p,j] = cell_freq_tch[p,j]
                        Next
                Next
                For p = 1 to cell_count
                        For j = 1 to random_tch_count
                                temp_freq_cost_table[p,random_tch_list[j]]
= freq_cost_table[p,random_tch_list[j]]
                        Next
                Next

        end if
      end if

Next // end n


// Best Cost un kopyalanmasi-----------------------------------------------------
----------------------
if cbx_bcch_plan.checked then
      cell_freq_bcch = best_cell_freq_bcch
      For p = 1 to cell_count
        For j = 1 to random_bcch_count
```

```
                freq_cost_table[p,random_bcch_list[j]] =
temp_freq_cost_table[p,random_bcch_list[j]]
          Next
        Next
end if

if cbx_tch_plan.checked then
        For p = 1 to cell_count
          For j = 1 to cell_tch_number[p]
                cell_freq_tch[p,j] = best_cell_freq_tch[p,j]
          Next
        Next
        For p = 1 to cell_count
          For j = 1 to random_tch_count
                freq_cost_table[p,random_tch_list[j]] =
temp_freq_cost_table[p,random_tch_list[j]]
          Next
        Next
end if

result_cost = f_calculate_cost()
st_status.text = "Real Cost : " + string(Round(result_cost, 3))


//Best Plani dw ye atma
f_make_best_plan_dw()

//Best plan check etme
f_check_new_plan()


// Bitirme Islemleri-----------------------------------------------------------
-----------------
st_status.text += " - Optimizing is finished!"

cb_save_best.enabled = true

enabled = true
SetPointer(Arrow!)
```

```
end event

type cb_save_initial from commandbutton within w_afp
int X=3570
int Y=1896
int Width=343
int Height=80
int TabOrder=290
boolean Enabled=false
boolean BringToTop=true
string Text="Save Plan As"
int TextSize=-8
int Weight=400
string FaceName="Arial TUR"
FontCharSet FontCharSet=TurkishCharSet!
FontFamily FontFamily=Swiss!
FontPitch FontPitch=Variable!
end type

event clicked;d_afp_cell_frequencies.SaveAs("",Text!,TRUE)
end event

type st_best_cost_label from statictext within w_afp
int X=2478
int Y=2248
int Width=480
int Height=80
boolean Enabled=false
boolean BringToTop=true
string Text="Best Cost Total"
Alignment Alignment=Right!
boolean FocusRectangle=false
long TextColor=255
long BackColor=82042848
int TextSize=-10
int Weight=700
string FaceName="Arial TUR"
FontCharSet FontCharSet=TurkishCharSet!
FontFamily FontFamily=Swiss!
```

FontPitch FontPitch=Variable!
end type

type st_best_cost from statictext within w_afp
int X=2976
int Y=2240
int Width=571
int Height=80
boolean Enabled=false
boolean BringToTop=true
boolean Border=true
Alignment Alignment=Right!
boolean FocusRectangle=false
long TextColor=33554432
long BackColor=15780518
int TextSize=-10
int Weight=700
string FaceName="Arial TUR"
FontCharSet FontCharSet=TurkishCharSet!
FontFamily FontFamily=Swiss!
FontPitch FontPitch=Variable!
end type

type d_afp_cell_frequencies_best from datawindow_common within w_afp
int X=2784
int Y=1952
int Width=901
int Height=828
int TabOrder=50
boolean Visible=false
boolean BringToTop=true
string DataObject="dwe_afp_cell_frequencies_best"
boolean TitleBar=false
BorderStyle BorderStyle=StyleBox!
boolean ControlMenu=false
boolean MinBox=false
boolean MaxBox=false
boolean HScrollBar=false

boolean VScrollBar=false
boolean Resizable=false
boolean LiveScroll=false
end type

type sle_optimize_times from singlelineedit within w_afp
int X=3570
int Y=1640
int Width=197
int Height=80
int TabOrder=80
boolean BringToTop=true
BorderStyle BorderStyle=StyleLowered!
boolean AutoHScroll=false
boolean RightToLeft=true
string Text="1000"
long TextColor=33554432
long BackColor=15793151
int TextSize=-8
int Weight=400
string FaceName="Arial TUR"
FontCharSet FontCharSet=TurkishCharSet!
FontFamily FontFamily=Swiss!
FontPitch FontPitch=Variable!
end type

type st_times from statictext within w_afp
int X=3785
int Y=1652
int Width=128
int Height=80
boolean Enabled=false
boolean BringToTop=true
string Text="times"
boolean FocusRectangle=false
long TextColor=33554432
long BackColor=82042848
int TextSize=-8
int Weight=400

```
string FaceName="Arial TUR"
FontCharSet FontCharSet=TurkishCharSet!
FontFamily FontFamily=Swiss!
FontPitch FontPitch=Variable!
end type

type st_status from statictext within w_afp
int X=2976
int Y=2612
int Width=937
int Height=164
boolean Enabled=false
boolean BringToTop=true
boolean Border=true
boolean FocusRectangle=false
long TextColor=33554432
long BackColor=67108864
int TextSize=-8
int Weight=400
string FaceName="Arial TUR"
FontCharSet FontCharSet=TurkishCharSet!
FontFamily FontFamily=Swiss!
FontPitch FontPitch=Variable!
end type

type st_status_label from statictext within w_afp
int X=2478
int Y=2616
int Width=480
int Height=80
boolean Enabled=false
boolean BringToTop=true
string Text="Status"
Alignment Alignment=Right!
boolean FocusRectangle=false
long TextColor=33554432
long BackColor=82042848
int TextSize=-10
int Weight=700
```

```
string FaceName="Arial TUR"
FontCharSet FontCharSet=TurkishCharSet!
FontFamily FontFamily=Swiss!
FontPitch FontPitch=Variable!
end type

type d_afp_initial_plan from datawindow_common within w_afp
int X=3886
int Y=44
int Width=727
int Height=1032
int TabOrder=40
boolean BringToTop=true
string DataObject="dwe_afp_cell_frequencies"
boolean TitleBar=false
BorderStyle BorderStyle=StyleLowered!
boolean ControlMenu=false
boolean MinBox=false
boolean MaxBox=false
boolean Resizable=false
end type

type cb_import_initial_plan from commandbutton within w_afp
int X=3909
int Y=1100
int Width=343
int Height=144
int TabOrder=140
boolean BringToTop=true
string Text="Import Plan"
int TextSize=-8
int Weight=400
string FaceName="Arial TUR"
FontCharSet FontCharSet=TurkishCharSet!
FontFamily FontFamily=Swiss!
FontPitch FontPitch=Variable!
end type

event clicked;string file_name,nul
```

```
decimal result_cost,total_cost, bcch_cost, tch_cost
long rc,last_row,p,j,i,o,m
long frequency_no,src_cll_ind,tru_number
decimal co_affect,adj_affect


//----------------Baslangic Islemler-------------------------------------
setnull(nul)
st_status.text = nul
st_best_cost.text = nul
st_best_bcch_cost.text = nul
st_best_tch_cost.text = nul

setNull(file_name)

same_ch_cost = long(sle_same_ch_cost.text)
adj_ch_cost = long(sle_adj_ch_cost.text)
forbid_ch_cost = long(sle_forbid_ch_cost.text)

rc = d_afp_cells.rowcount()
if rc < 1 then
	Messagebox("Warning","Cell table must be imported")
	return
end if

last_row = d_afp_icdm.rowcount()
if last_row < 1 then
	Messagebox("Warning","ICDM table must be imported")
	return
end if

if not(cbx_tch_plan.checked) and not(cbx_bcch_plan.checked) then
	messagebox("Warning","TCH, BCCH or both checkboxes must be
selected")
	return
end if

d_afp_initial_plan.reset()
d_afp_initial_plan.ImportFile ( file_name )
```

```
if d_afp_initial_plan.rowcount() > 0 then

        enabled = false
        SetPointer(HourGlass!)

        cb_plan.enabled = false
        cb_import_initial_plan.enabled = false
        cb_save_best.enabled = false
        cb_save_initial.enabled = false
        cb_add_bcch.enabled = false
        cb_remove_bcch.enabled = false
        cb_add_tch.enabled = false
        cb_remove_tch.enabled = false

        d_afp_cell_frequencies.reset()
        d_afp_cell_frequencies_best.reset()

        st_status.text = "Wait for importing!"

        d_afp_initial_plan.SetRedraw(false)
        d_afp_initial_plan.SetSort("#1 A")
        d_afp_initial_plan.Sort()

        f_set_random_bcch()
        f_set_random_tch()

        f_calculate_nbr_count()

 //-----------------Initial Olusturma---------------------------------------
        f_copy_initial_to_cell_freq()

        total_cost = 0
        bcch_cost = 0
        tch_cost = 0
        //BCCH
        if cbx_bcch_plan.checked then
          result_cost = 0
```

```
        For p = 1 to cell_count
                result_cost = result_cost +
(freq_cost_table[p,cell_freq_bcch[p]] - 100000)
        Next
        bcch_cost = result_cost
        st_initial_bcch_cost.text = string(Round(result_cost,3))
      end if

      //TCH
      if cbx_tch_plan.checked then
        result_cost = 0
        For p = 1 to cell_count
                For j = 1 to cell_tch_number[p]
                        result_cost = result_cost +
(freq_cost_table[p,cell_freq_tch[p,j]] - 100000)
                Next
        Next
        tch_cost = result_cost
        st_initial_tch_cost.text = string(Round(result_cost,3))
      end if

      total_cost = bcch_cost + tch_cost
      st_initial_cost.text = string(Round(total_cost,3))

      result_cost = f_calculate_cost()
      st_status.text = "Real Cost : " + string(Round(result_cost,3))


      // Initial plani dw ye atma
      For i = 1 to cell_count
        d_afp_cell_frequencies.insertrow(i)

      d_afp_cell_frequencies.setitem(i,"cell_name",lb_cell_name.text(i))
        tru_number = cell_tch_number[i] + 1
        d_afp_cell_frequencies.setitem(i,"tru_number",tru_number)

        bcch_cost = 0
        if cbx_bcch_plan.checked then
```

```
        d_afp_cell_frequencies.setitem(i,"bcch_cost",(freq_cost_table[i,cell
_freq_bcch[i]] - 100000))
                d_afp_cell_frequencies.setitem(i,"bcch",cell_freq_bcch[i])
                bcch_cost = (freq_cost_table[i,cell_freq_bcch[i]] -
100000)
          end if

          tch_cost = 0
          result_cost = 0
          if cbx_tch_plan.checked then
                For j = 1 to cell_tch_number[i]
                        result_cost = result_cost +
(freq_cost_table[i,cell_freq_tch[i,j]] - 100000)

        d_afp_cell_frequencies.setitem(i,("tch"+string(j)),cell_freq_tch[i,j])
                Next
                d_afp_cell_frequencies.setitem(i,"tch_cost",result_cost)
                tch_cost = result_cost
          end if

          total_cost = bcch_cost + tch_cost
          d_afp_cell_frequencies.setitem(i,"total_cost",total_cost)

        Next

        //Initial plan check etme
        f_check_initial_plan()



 //--------------------Fixed frekanslarin atanmasi----------------------------
----------------
        For i = 1 to cell_count

          //Fixed BCCH
          if fixed_bcch_list[i] > 0 then

                frequency_no = fixed_bcch_list[i]
```

```
                if (cell_freq_bcch[i] <> frequency_no) then // Oncekinden
Farkli Frekans ise

                        if cell_freq_bcch[i] > 0 then // Frekans
degisimi

                                For o = 1 to cell_interfered_count[i]

                                        src_cll_ind =
icdm_interfered_ind[i,o]

                                        co_affect = icdm_co_int[i,o]
                                        adj_affect =
icdm_adj_int[i,o]


        freq_cost_table[src_cll_ind,frequency_no] += (co_affect * 10)
                                        if frequency_no > 1 then
freq_cost_table[src_cll_ind,frequency_no - 1] += (adj_affect * 10)
                                        if frequency_no < 124 then
freq_cost_table[src_cll_ind,frequency_no + 1] += (adj_affect * 10)


        freq_cost_table[src_cll_ind,cell_freq_bcch[i]] -= (co_affect * 10)
                                        if cell_freq_bcch[i] > 1 then
freq_cost_table[src_cll_ind,cell_freq_bcch[i] - 1] -= (adj_affect * 10)
                                        if cell_freq_bcch[i] < 124
then freq_cost_table[src_cll_ind,cell_freq_bcch[i] + 1] -= (adj_affect *
10)

                                Next

                                freq_cost_table[i,cell_freq_bcch[i]]
-= 100000
                                if cell_freq_bcch[i] > 1 then
freq_cost_table[i,cell_freq_bcch[i] - 1] -= 100000
                                if cell_freq_bcch[i] < 124 then
freq_cost_table[i,cell_freq_bcch[i] + 1] -= 100000
```

```
                    else // ilk Frekans

                        For o = 1 to cell_interfered_count[i]

                            src_cll_ind =
icdm_interfered_ind[i,o]

                            co_affect = icdm_co_int[i,o]
                            adj_affect =
icdm_adj_int[i,o]


      freq_cost_table[src_cll_ind,frequency_no] += (co_affect * 10)
                            if frequency_no > 1 then
freq_cost_table[src_cll_ind,frequency_no - 1] += (adj_affect * 10)
                            if frequency_no < 124 then
freq_cost_table[src_cll_ind,frequency_no + 1] += (adj_affect * 10)


                        Next

                    end if

                    cell_freq_bcch[i] = frequency_no

                    freq_cost_table[i,frequency_no] += 100000
                    if frequency_no > 1 then
freq_cost_table[i,frequency_no - 1] += 100000
                    if frequency_no < 124 then
freq_cost_table[i,frequency_no + 1] += 100000

                end if

        end if

        //Fixed TCH
        For m = 1 to fixed_tch_count[i]
```

```
                frequency_no = fixed_tch_list[i,m]


                if (cell_freq_tch[i,m] <> frequency_no) then //
Oncekinden Farkli Frekans ise

                        if cell_freq_tch[i,m] > 0 then // Frekans
degisimi

                                For o = 1 to cell_interfered_count[i]

                                        src_cll_ind =
icdm_interfered_ind[i,o]

                                        co_affect = icdm_co_int[i,o]
                                        adj_affect =
icdm_adj_int[i,o]


        freq_cost_table[src_cll_ind,frequency_no] += (co_affect)
                                        if frequency_no > 1 then
freq_cost_table[src_cll_ind,frequency_no - 1] += (adj_affect)
                                        if frequency_no < 124 then
freq_cost_table[src_cll_ind,frequency_no + 1] += (adj_affect)


        freq_cost_table[src_cll_ind,cell_freq_tch[i,m]] -= (co_affect)
                                        if cell_freq_tch[i,m] > 1 then
freq_cost_table[src_cll_ind,cell_freq_tch[i,m] - 1] -= (adj_affect)
                                        if cell_freq_tch[i,m] < 124
then freq_cost_table[src_cll_ind,cell_freq_tch[i,m] + 1] -= (adj_affect)

                                Next

                                freq_cost_table[i,cell_freq_tch[i,m]]
-= 100000
                                if cell_freq_tch[i,m] > 1 then
freq_cost_table[i,cell_freq_tch[i,m] - 1] -= 100000
```

```
                                        if cell_freq_tch[i,m] < 124 then
freq_cost_table[i,cell_freq_tch[i,m] + 1] -= 100000

                                    else // ilk Frekans

                                        For o = 1 to cell_interfered_count[i]

                                            src_cll_ind =
icdm_interfered_ind[i,o]

                                            co_affect = icdm_co_int[i,o]
                                            adj_affect =
icdm_adj_int[i,o]


        freq_cost_table[src_cll_ind,frequency_no] += (co_affect)
                                            if frequency_no > 1 then
freq_cost_table[src_cll_ind,frequency_no - 1] += (adj_affect)
                                            if frequency_no < 124 then
freq_cost_table[src_cll_ind,frequency_no + 1] += (adj_affect)


                                        Next

                                    end if


                                    cell_freq_tch[i,m] = frequency_no
                                    freq_cost_table[i,frequency_no] += 100000
                                    if frequency_no > 1 then
freq_cost_table[i,frequency_no - 1] += 100000
                                    if frequency_no < 124 then
freq_cost_table[i,frequency_no + 1] += 100000

                end if

          Next

        Next
```

```
 //----------------Bitirme Islemleri--------------------------------------
        cb_optimize.enabled = true
        cb_save_initial.enabled = true
        cb_clear_initial_plan.enabled = true
        cb_check_plan.enabled = true

        d_afp_initial_plan.SetRedraw(true)

        st_status.text += " - Initial Plan imported!"

        SetPointer(Arrow!)

end if
end event

type cb_clear_initial_plan from commandbutton within w_afp
int X=4270
int Y=1100
int Width=343
int Height=144
int TabOrder=150
boolean Enabled=false
boolean BringToTop=true
string Text="Clear Plan"
int TextSize=-8
int Weight=400
string FaceName="Arial TUR"
FontCharSet FontCharSet=TurkishCharSet!
FontFamily FontFamily=Swiss!
FontPitch FontPitch=Variable!
end type

event clicked;string nul

setnull(nul)

d_afp_initial_plan.reset()
```

```
lb_cell_name.reset()
lb_external_name.reset()

cb_plan.enabled = true
cb_optimize.enabled = false

cb_save_initial.enabled = false
cb_save_best.enabled = false
cb_check_plan.enabled = false

cb_import_initial_plan.enabled = true
cb_clear_initial_plan.enabled = false

cb_add_bcch.enabled = true
cb_remove_bcch.enabled = true
cb_add_tch.enabled = true
cb_remove_tch.enabled = true

d_afp_cell_frequencies.reset()
d_afp_cell_frequencies_best.reset()

st_initial_cost.text = nul
st_initial_bcch_cost.text = nul
st_initial_tch_cost.text = nul
st_best_cost.text = nul
st_best_bcch_cost.text = nul
st_best_tch_cost.text = nul

cb_import_initial_plan.enabled = true

st_status.text = "Initial Plan cleared!"
end event

type d_afp_icdm_interferer from datawindow_common within w_afp
int X=2971
int Y=2212
int Width=992
int Height=652
int TabOrder=30
```

```
boolean Visible=false
boolean BringToTop=true
string DataObject="dwe_afp_icdm"
boolean TitleBar=false
BorderStyle BorderStyle=StyleBox!
boolean ControlMenu=false
boolean MinBox=false
boolean MaxBox=false
boolean HScrollBar=false
boolean VScrollBar=false
boolean Resizable=false
boolean LiveScroll=false
end type

type cb_check_plan from commandbutton within w_afp
int X=430
int Y=1100
int Width=384
int Height=144
int TabOrder=280
boolean Enabled=false
boolean BringToTop=true
string Text="View Cell Plan"
int TextSize=-8
int Weight=400
string FaceName="Arial TUR"
FontCharSet FontCharSet=TurkishCharSet!
FontFamily FontFamily=Swiss!
FontPitch FontPitch=Variable!
end type

event clicked;long
selected_row,found_row,last_row,found_row_2,tch_count,bcch,tch,i,j
string cell_name,find_string,interferer_cell,temp_cell_name
decimal co_affect,adj_affect

dr_frequency.reset()

For i = 1 to 124
```

```
        dr_frequency.insertrow(0)
        dr_frequency.setitem(i,"frequency",i)
Next

selected_row = d_afp_cells.getrow()
last_row = d_afp_icdm.rowcount()

cell_name = d_afp_cells.getitemstring(selected_row,"cell_name")

find_string = "source_cell = '" + cell_name + "'"
found_row = d_afp_icdm.Find(find_string,1,last_row)

if found_row > 0 then
        Do
          interferer_cell =
d_afp_icdm.getitemstring(found_row,"interferer_cell")
          co_affect =
d_afp_icdm.getitemdecimal(found_row,"traf_affected_co_percent")
          adj_affect =
d_afp_icdm.getitemdecimal(found_row,"traf_affected_adj_percent")

          find_string = "cell_name = '" + interferer_cell + "'"
          found_row_2 =
d_afp_cell_frequencies_best.Find(find_string,1,d_afp_cell_frequencies_b
est.rowcount())
          if found_row_2 > 0 then
                bcch =
d_afp_cell_frequencies_best.getitemnumber(found_row_2,"bcch")
                temp_cell_name =
dr_frequency.getitemstring(bcch,"cell_name")

                if len(temp_cell_name) > 0 then
                        temp_cell_name = temp_cell_name + "| " +
interferer_cell + "(" + string(co_affect) + "/" + string(adj_affect) + ")"
                else
                        temp_cell_name = interferer_cell + "(" +
string(co_affect) + "/" + string(adj_affect) + ")"
                end if
                dr_frequency.setitem(bcch,"cell_name",temp_cell_name)
```

```
                tch_count =
d_afp_cell_frequencies_best.getitemnumber(found_row_2,"total_tch_cou
nt")
                For j = 1 to tch_count
                        tch =
d_afp_cell_frequencies_best.getitemnumber(found_row_2,("tch"+string(j
)))
                        temp_cell_name =
dr_frequency.getitemstring(tch,"cell_tch_name")
                        if len(temp_cell_name) > 0 then
                                temp_cell_name = temp_cell_name + "| " +
interferer_cell + "(" + string(co_affect) + "/" + string(adj_affect) + ")"
                        else
                                temp_cell_name = interferer_cell + "(" +
string(co_affect) + "/" + string(adj_affect) + ")"
                        end if

        dr_frequency.setitem(tch,"cell_tch_name",temp_cell_name)
                Next
          end if

          found_row = found_row + 1
          if found_row > last_row then exit
        Loop until (cell_name <>
d_afp_icdm.getitemstring(found_row,"source_cell") )
end if

dr_frequency.show()
end event

type dr_frequency from datawindow_main within w_afp
int X=69
int Y=132
int Width=3337
int Height=2664
int TabOrder=20
boolean Visible=false
boolean BringToTop=true
```

```
string DataObject="dwr_frequency"
boolean TitleBar=true
string Title="Frequency report"
end type

type lb_cell_name from listbox within w_afp
int X=2624
int Y=2220
int Width=686
int Height=500
int TabOrder=350
boolean Visible=false
boolean BringToTop=true
BorderStyle BorderStyle=StyleLowered!
boolean Sorted=false
long TextColor=33554432
int TextSize=-8
int Weight=400
string FaceName="Arial TUR"
FontCharSet FontCharSet=TurkishCharSet!
FontFamily FontFamily=Swiss!
FontPitch FontPitch=Variable!
end type

type lb_external_name from listbox within w_afp
int X=2693
int Y=2272
int Width=686
int Height=500
int TabOrder=340
boolean Visible=false
boolean BringToTop=true
BorderStyle BorderStyle=StyleLowered!
boolean Sorted=false
long TextColor=33554432
int TextSize=-8
int Weight=400
string FaceName="Arial TUR"
FontCharSet FontCharSet=TurkishCharSet!
```

FontFamily FontFamily=Swiss!
FontPitch FontPitch=Variable!
end type

type cbx_bcch_plan from checkbox within w_afp
int X=3557
int Y=1548
int Width=215
int Height=80
boolean BringToTop=true
string Text="BCCH"
BorderStyle BorderStyle=StyleLowered!
boolean LeftText=true
long TextColor=33554432
long BackColor=67108864
int TextSize=-8
int Weight=400
string FaceName="Arial TUR"
FontCharSet FontCharSet=TurkishCharSet!
FontFamily FontFamily=Swiss!
FontPitch FontPitch=Variable!
end type

type cbx_tch_plan from checkbox within w_afp
int X=3557
int Y=1476
int Width=215
int Height=80
boolean BringToTop=true
string Text="TCH"
BorderStyle BorderStyle=StyleLowered!
boolean LeftText=true
long TextColor=33554432
long BackColor=67108864
int TextSize=-8
int Weight=400
string FaceName="Arial TUR"
FontCharSet FontCharSet=TurkishCharSet!
FontFamily FontFamily=Swiss!

```
FontPitch FontPitch=Variable!
end type

type st_best_bcch_cost from statictext within w_afp
int X=2976
int Y=2332
int Width=571
int Height=80
boolean Enabled=false
boolean BringToTop=true
boolean Border=true
Alignment Alignment=Right!
boolean FocusRectangle=false
long TextColor=33554432
long BackColor=15780518
int TextSize=-10
int Weight=700
string FaceName="Arial TUR"
FontCharSet FontCharSet=TurkishCharSet!
FontFamily FontFamily=Swiss!
FontPitch FontPitch=Variable!
end type

type st_best_tch_cost from statictext within w_afp
int X=2976
int Y=2424
int Width=571
int Height=80
boolean Enabled=false
boolean BringToTop=true
boolean Border=true
Alignment Alignment=Right!
boolean FocusRectangle=false
long TextColor=33554432
long BackColor=15780518
int TextSize=-10
int Weight=700
string FaceName="Arial TUR"
FontCharSet FontCharSet=TurkishCharSet!
```

FontFamily FontFamily=Swiss!
FontPitch FontPitch=Variable!
end type

type st_best_bcch_cost_label from statictext within w_afp
int X=2478
int Y=2340
int Width=480
int Height=80
boolean Enabled=false
boolean BringToTop=true
string Text="BCCH"
Alignment Alignment=Right!
boolean FocusRectangle=false
long TextColor=255
long BackColor=82042848
int TextSize=-10
int Weight=700
string FaceName="Arial TUR"
FontCharSet FontCharSet=TurkishCharSet!
FontFamily FontFamily=Swiss!
FontPitch FontPitch=Variable!
end type

type st_best_tch_cost_label from statictext within w_afp
int X=2478
int Y=2432
int Width=480
int Height=80
boolean Enabled=false
boolean BringToTop=true
string Text="TCH"
Alignment Alignment=Right!
boolean FocusRectangle=false
long TextColor=255
long BackColor=82042848
int TextSize=-10
int Weight=700
string FaceName="Arial TUR"

FontCharSet FontCharSet=TurkishCharSet!
FontFamily FontFamily=Swiss!
FontPitch FontPitch=Variable!
end type
type gb_3 from groupbox within w_afp
int X=14
int Width=4622
int Height=1276
int TabOrder=300
BorderStyle BorderStyle=StyleLowered!
long TextColor=33554432
long BackColor=67108864
int TextSize=-8
int Weight=400
string FaceName="Arial TUR"
FontCharSet FontCharSet=TurkishCharSet!
FontFamily FontFamily=Swiss!
FontPitch FontPitch=Variable!
end type

type gb_1 from groupbox within w_afp
int X=18
int Y=1284
int Width=2423
int Height=1624
int TabOrder=320
BorderStyle BorderStyle=StyleLowered!
long TextColor=33554432
long BackColor=67108864
int TextSize=-8
int Weight=400
string FaceName="Arial TUR"
FontCharSet FontCharSet=TurkishCharSet!
FontFamily FontFamily=Swiss!
FontPitch FontPitch=Variable!
end type

type cb_save_best from commandbutton within w_afp
int X=3570

```
int Y=2240
int Width=343
int Height=80
int TabOrder=270
boolean Enabled=false
string Text="Save Plan As"
int TextSize=-8
int Weight=400
string FaceName="Arial TUR"
FontCharSet FontCharSet=TurkishCharSet!
FontFamily FontFamily=Swiss!
FontPitch FontPitch=Variable!
end type


event clicked;d_afp_cell_frequencies_best.SaveAs("",Text!,TRUE)
end event

type gb_2 from groupbox within w_afp
int X=2469
int Y=1284
int Width=1495
int Height=1624
int TabOrder=310
BorderStyle BorderStyle=StyleLowered!
long TextColor=33554432
long BackColor=67108864
int TextSize=-8
int Weight=400
string FaceName="Arial TUR"
FontCharSet FontCharSet=TurkishCharSet!
FontFamily FontFamily=Swiss!
FontPitch FontPitch=Variable!
end type

type st_initial_bcch_cost_label from statictext within w_afp
int X=2478
int Y=1996
int Width=480
```

int Height=80
boolean Enabled=false
boolean BringToTop=true
string Text="BCCH"
Alignment Alignment=Right!
boolean FocusRectangle=false
long TextColor=33554432
long BackColor=82042848
int TextSize=-10
int Weight=700
string FaceName="Arial TUR"
FontCharSet FontCharSet=TurkishCharSet!
FontFamily FontFamily=Swiss!
FontPitch FontPitch=Variable!
end type

type st_initial_tch_cost_label from statictext within w_afp
int X=2478
int Y=2088
int Width=480
int Height=80
boolean Enabled=false
boolean BringToTop=true
string Text="TCH"
Alignment Alignment=Right!
boolean FocusRectangle=false
long TextColor=33554432
long BackColor=82042848
int TextSize=-10
int Weight=700
string FaceName="Arial TUR"
FontCharSet FontCharSet=TurkishCharSet!
FontFamily FontFamily=Swiss!
FontPitch FontPitch=Variable!
end type

type st_initial_bcch_cost from statictext within w_afp
int X=2976
int Y=1988

int Width=571
int Height=80
boolean Enabled=false
boolean BringToTop=true
boolean Border=true
Alignment Alignment=Right!
boolean FocusRectangle=false
long TextColor=33554432
long BackColor=15793151
int TextSize=-10
int Weight=700
string FaceName="Arial TUR"
FontCharSet FontCharSet=TurkishCharSet!
FontFamily FontFamily=Swiss!
FontPitch FontPitch=Variable!
end type

type st_initial_tch_cost from statictext within w_afp
int X=2976
int Y=2080
int Width=571
int Height=80
boolean Enabled=false
boolean BringToTop=true
boolean Border=true
Alignment Alignment=Right!
boolean FocusRectangle=false
long TextColor=33554432
long BackColor=15793151
int TextSize=-10
int Weight=700
string FaceName="Arial TUR"
FontCharSet FontCharSet=TurkishCharSet!
FontFamily FontFamily=Swiss!
FontPitch FontPitch=Variable!
end type

type cb_import_icdm_msmt from commandbutton within w_afp
int X=2907

```
int Y=1100
int Width=704
int Height=144
int TabOrder=180
boolean BringToTop=true
string Text="Convert ICDM (.msmt) to (.txt)"
int TextSize=-8
int Weight=400
string FaceName="Arial TUR"
FontCharSet FontCharSet=TurkishCharSet!
FontFamily FontFamily=Swiss!
FontPitch FontPitch=Variable!
end type

event clicked;
enabled = false
SetPointer(HourGlass!)
w_afp.Pointer = 'HourGlass!'

string docname, named, target_file, target_file_2, icdm_file_name
integer value
long pos_found

value = GetFileOpenName("Select ICDM File",docname, named,
"MSMT","ICDM Files (*.MSMT),*.MSMT")

IF value = 1 THEN

	icdm_file_name = docname

	pos_found = Pos(icdm_file_name,".",1)
	target_file =
Replace(icdm_file_name,pos_found,(len(icdm_file_name) - pos_found +
1),"-neptune.txt")
	target_file_2 = "c:\neptune_temp_icdm.txt"

	f_change_delimeter_char_for_file(docname,target_file_2,".",",",0)

	if FileExists(target_file) then FileDelete(target_file)
```

```
        f_initialize_icdm_file(target_file_2,target_file)

        if FileExists(target_file_2) then FileDelete(target_file_2)

        icdm_file_name = target_file

        st_status.text = "ICDM converted!"

END IF

w_afp.Pointer = 'Arrow!'
enabled = true
SetPointer(Arrow!)
end event

type sle_same_ch_cost from singlelineedit within w_afp
int X=2025
int Y=1440
int Width=379
int Height=80
int TabOrder=90
boolean BringToTop=true
BorderStyle BorderStyle=StyleLowered!
boolean AutoHScroll=false
boolean RightToLeft=true
string Text="10000"
long TextColor=33554432
long BackColor=15793151
int TextSize=-8
int Weight=400
string FaceName="Arial TUR"
FontCharSet FontCharSet=TurkishCharSet!
FontFamily FontFamily=Swiss!
FontPitch FontPitch=Variable!
end type

type sle_adj_ch_cost from singlelineedit within w_afp
int X=2025
int Y=1528
```

int Width=379
int Height=80
int TabOrder=100
boolean BringToTop=true
BorderStyle BorderStyle=StyleLowered!
boolean AutoHScroll=false
boolean RightToLeft=true
string Text="3"
long TextColor=33554432
long BackColor=15793151
int TextSize=-8
int Weight=400
string FaceName="Arial TUR"
FontCharSet FontCharSet=TurkishCharSet!
FontFamily FontFamily=Swiss!
FontPitch FontPitch=Variable!
end type

type sle_forbid_ch_cost from singlelineedit within w_afp
int X=2025
int Y=1616
int Width=379
int Height=80
int TabOrder=100
boolean BringToTop=true
BorderStyle BorderStyle=StyleLowered!
boolean AutoHScroll=false
boolean RightToLeft=true
string Text="10000"
long TextColor=33554432
long BackColor=15793151
int TextSize=-8
int Weight=400
string FaceName="Arial TUR"
FontCharSet FontCharSet=TurkishCharSet!
FontFamily FontFamily=Swiss!
FontPitch FontPitch=Variable!
end type

type st_1 from statictext within w_afp
int X=1426
int Y=1444
int Width=585
int Height=80
boolean Enabled=false
boolean BringToTop=true
string Text="Cosite Same Channel"
boolean FocusRectangle=false
long TextColor=33554432
long BackColor=80269524
int TextSize=-10
int Weight=400
string FaceName="Arial TUR"
FontCharSet FontCharSet=TurkishCharSet!
FontFamily FontFamily=Swiss!
FontPitch FontPitch=Variable!
end type

type st_2 from statictext within w_afp
int X=1495
int Y=1532
int Width=512
int Height=80
boolean Enabled=false
boolean BringToTop=true
string Text="Cosite Adj Channel"
boolean FocusRectangle=false
long TextColor=33554432
long BackColor=80269524
int TextSize=-10
int Weight=400
string FaceName="Arial TUR"
FontCharSet FontCharSet=TurkishCharSet!
FontFamily FontFamily=Swiss!
FontPitch FontPitch=Variable!
end type

type st_3 from statictext within w_afp

int X=1509
int Y=1620
int Width=498
int Height=80
boolean Enabled=false
boolean BringToTop=true
string Text="Forbidden Channel"
boolean FocusRectangle=false
long TextColor=33554432
long BackColor=80269524
int TextSize=-10
int Weight=400
string FaceName="Arial TUR"
FontCharSet FontCharSet=TurkishCharSet!
FontFamily FontFamily=Swiss!
FontPitch FontPitch=Variable!
end type

type st_4 from statictext within w_afp
int X=2121
int Y=1344
int Width=201
int Height=80
boolean Enabled=false
boolean BringToTop=true
string Text="COSTs"
boolean FocusRectangle=false
long TextColor=33554432
long BackColor=82042848
int TextSize=-10
int Weight=700
string FaceName="Arial TUR"
FontCharSet FontCharSet=TurkishCharSet!
FontFamily FontFamily=Swiss!
FontPitch FontPitch=Variable!
end type

$PBExportHeader$f_initialize_icdm_file.srf

```
global type f_initialize_icdm_file from function_object
end type

forward prototypes
global subroutine f_initialize_icdm_file (string source_file_name, string
target_file_name)
end prototypes

global subroutine f_initialize_icdm_file (string source_file_name, string
target_file_name);
long file_pointer_for_reading , file_pointer_for_writing ,
tab_1_pos, tab_2_pos, position_found
int file_pointer_return_value, i
string a_line_from_source_file

file_pointer_for_reading = FileOpen(source_file_name,
LineMode!, Read!, LockWrite!)

IF file_pointer_for_reading = -1 THEN
	RETURN
ELSEIF isnull( file_pointer_for_reading ) THEN
	RETURN
END IF

file_pointer_return_value = FileRead(file_pointer_for_reading,
a_line_from_source_file) //basliklari almamak icin bi daha okutup ilk
data satirina atlanmali

IF file_pointer_return_value > 0 THEN

	file_pointer_return_value =
	FileRead(file_pointer_for_reading, a_line_from_source_file)
	file_pointer_for_writing = FileOpen(target_file_name,
LineMode!, Write!, LockWrite!, Replace!)

	DO WHILE ( file_pointer_return_value <> -100 )

	  position_found = Pos(a_line_from_source_file,"UNKNOWN")
```

```
        IF ( position_found = 0 )  THEN // UNKNOWN yok ise
                if Mid(a_line_from_source_file,31,1) <> Char(9) then //
co affect null degilse
                        tab_1_pos =
Pos(a_line_from_source_file,Char(9),31)
                        tab_2_pos =
Pos(a_line_from_source_file,Char(9),(tab_1_pos + 1))
                        if (tab_2_pos - tab_1_pos) > 1 then // adj affect
null degilse
                                if Mid(a_line_from_source_file,31,1) <>
"0" or Mid(a_line_from_source_file,33,1) <> "0" then // co ve adj affect
0 degilse
                                        a_line_from_source_file       =
      Replace(a_line_from_source_file, 1, 8 , "" )
                                        a_line_from_source_file       =
      Replace(a_line_from_source_file, 8, 8 , "" )
                                        FileWrite(file_pointer_for_writing,
a_line_from_source_file)
                                end if
                        end if
                end if
        END IF

        file_pointer_return_value = FileRead(file_pointer_for_reading,
a_line_from_source_file)

      LOOP



ELSEIF file_pointer_return_value = -1 THEN
      RETURN
ELSEIF isnull(file_pointer_return_value) THEN
      RETURN
END IF

FileClose(file_pointer_for_reading)
FileClose(file_pointer_for_writing)
```

```
RETURN
end subroutine


$PBExportHeader$f_get_bcch_from_cell.srf
global type f_get_bcch_from_cell from function_object
end type

forward prototypes
global function long f_get_bcch_from_cell (string cell_name)
end prototypes

global function long f_get_bcch_from_cell (string cell_name);long
bcchno

setnull(bcchno)
SELECT oss_internal_cell.bcchno
INTO :bcchno
FROM oss_internal_cell
WHERE (oss_internal_cell.cell = :cell_name) and
        (oss_internal_cell.configuration = 0 );

return bcchno
end function


$PBExportHeader$f_get_all_tch_from_cell.srf
global type f_get_all_tch_from_cell from function_object
end type

forward prototypes
global function string f_get_all_tch_from_cell (string cell_name)
end prototypes

global function string f_get_all_tch_from_cell (string cell_name);string
freq_ch_0, chgr , tch_list
long cell_freqs[],freq_count, bcchno, tch_freq, i, j, k
boolean tch_ok
```

```
freq_count = 0

setnull(bcchno)
SELECT oss_internal_cell.bcchno
INTO :bcchno
FROM oss_internal_cell
WHERE (oss_internal_cell.cell = :cell_name) and
        (oss_internal_cell.configuration = 0 );

For k = 1 to 3
      chgr = string (k - 1)

      setnull(freq_ch_0)
      SELECT oss_channel_group.dchno
      INTO :freq_ch_0
      FROM oss_channel_group
      WHERE (oss_channel_group.cell = :cell_name) and
            (oss_channel_group.configuration = 0 ) and
            (oss_channel_group.chgr = :chgr);

      if len(freq_ch_0) > 0 then
        freq_ch_0 = mid(freq_ch_0,2,(len(freq_ch_0) - 2))
        For i = 1 to f_get_from_count(freq_ch_0)

            tch_freq = long(f_get_from(freq_ch_0,i))
            if bcchno <> tch_freq then

                  tch_ok = true
                  For j = 1 to freq_count
                        if cell_freqs[j] = tch_freq then
                              tch_ok = false
                              exit
                        end if
                  Next

                  if tch_ok = true then
                        freq_count++
                        cell_freqs[freq_count] = tch_freq
                  end if
```

```
                    end if

                Next
            end if

Next

if freq_count > 0 then
        tch_list = string(cell_freqs[1])
else
        tch_list = ""
end if

For k = 2 to freq_count
        tch_list = tch_list + "," + string(cell_freqs[k])
Next

return tch_list
end function
```

KAYNAKLAR DİZİNİ

Eisenblaetter, A., Kürner T. and Fauss R., 1999, Radio Planning Algorithms for Interference Reduction in Cellular Networks, Communications for the Millenium, Proceedings of COST252/259 Joint Workshop, University of Bradford

Ericsson Radio Systems AB, 1998a, GSM System Survey, EN/LZT 123 3321 R2A

Ericsson Radio Systems AB, 1998b, GSM Cell Planning Principle, EN/LZT 123 3314 R3A

Ericsson Radio Systems AB, 1998c, GSM System Introduction, EN/LZT 123 3641 R2A

Ericsson Radio Systems AB, 1998d, GSM Cell Planning Overview, EN/LZT 123 3313 R3A

Ericsson Radio Systems AB, 1998e, GSM Cell Planning Workshop, EN/LZT 123 3315 R3A

Halonen, T., Romero, J., and Melero, J., 2003, GSM, GPRS and EDGE Performance Second Ed., John Wiley&Sons, 0-470-86694-2

Harputluoğlu, C., 2000, GSM'in Altyapısı ve Gelişimi

Lundqvist F. and Raismaa S., 1999, Performance Evaluation of a Closed Loop Frequency Optimisation Algorithm for GSM, Master Thesis performed in Automatic Control, Linköping Institute of Technology

Mishra, A. R., 2004, Fundamentals of Cellular Network Planning & Optimisation, John Wiley&Sons, 0-470-86267-X

ÖZGEÇMİŞ

Adı Soyadı : Serkan Kayacan

Doğum Tarihi : 05.09.1982

Doğum Yeri : Konak/İZMİR

Medeni Hali : Bekar

Uyruğu : T.C.

Kariyer

2006-... : Turkcell İletişim Hiz. A.Ş. - İzmir, Yazılım Mühendisi

2004-2006 : Turkcell İletişim Hiz. A.Ş. - İzmir, Hücre Planlama ve Optimizasyon Mühendisi

Eğitim

2004-... : Ege Üniversitesi, Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı, Y.Lisans

2000-2004 : Ege Üniversitesi, Mühendislik Fakültesi Bilgisayar Mühendisliği Bölümü, Lisans

1993-2000 : İzmir Bornova Anadolu Lisesi

Yabancı Dil : İngilizce, Almanca