İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

TELEKOMÜNİKASYON YÖNETİM ŞEBEKESİ TMN

YÜKSEK LİSANS TEZİ

Müh. Kutlay TETİK
(504971060)

142894

Tezin Enstitüye Verildiği Tarih : 24 Aralık 2002

Tezin Savunulduğu Tarih : 13 Ocak 2003

Tez Danışmanı : Prof. Dr. Günsel DURUSOY 4.2.03

Diğer Jüri Üyeleri : Prof. Dr. Ümit AYGÖLÜ

Yrd. Doç. Dr. Demir ÖNER (İ.Ü.)

OCAK 2003

ÖNSÖZ

Yaklaşık üç yıldır Nortel Networks/Netaş'ta TASMUS projesi kapsamında ATM santralinin TMN ile yönetim projesi üzerinde çalışmaktayım. Bu konuda tez hazırlamamda beni teşvik eden tez danışmanım Prof. Dr. Günsel Durusoy'a, Netaş'taki kaynaklardan yararlanmamı sağlayan Uygulama Geliştirme Departmanı Müdürü Sayat Arslanlıoğlu'na, TMN standartları konusunda her türlü kaynağı bana temin eden Netaş'ta sistem mühendisi olarak görev yapan Orhan Uçar'a teşekkür ederim. Son olarak aileme, bana her türlü destekte bulundukları için teşekkür ederim.

Ocak 2003 Kutlay TETİK

İÇİNDEKİLER

KISALTMALAR

ACSE	Association Control Service Element
ASN.1	Abstract Syntax Notation 1
ATM	Asenkron Transfer Mod
BER	Basic Encoding Rules
CMIP	Common Management Information Protocol
CMISE	Common Management Information Service Element
CORBA	Common Object Request Broker Architecture
EFD	Event Forwarding Discriminator
FTAM	File Transfer Access Module
GDMO	Guidelines for The Definition of Managed Objects
GSM	Global System for Mobile Communications
IDL	Interface Definition Language
ISDN	Integrated Services Data Network
ITU-T	International Telecommunications Union-Telecommunications
MF	Mediation Function
MIB	Management Information Base
NE	Network Element
NEF	Network Element Function
OAM&P	Operation, Administration, Maintenance & Provisioning
OS	Operasyon Sistemi
OSF	Operasyon Sistem Fonksiyonu
OSI	Open System Interface
PDU	Protocol Data Unit
QAF	Q Adaptor Function
QOS	Quality of Service
ROSE	Remote Operations Service Element
SAP	Service Access Point
SDH	Synchronous Digital Hiearchy
SMASE	Systems Management Application Service Entity
SNMP	Simple Network Management Protocol
TCP/IP	Transmission Control Protocol/Internet Protocol
TMN	Telecommunications Management Network
WSF	Work Station Function

TABLO LİSTESİ

ŞEKİL LİSTESİ

TELEKOMÜNİKASYON YÖNETİM ŞEBEKESİ – TMN

ÖZET

Haberleşme şebekesi yönetimi, bir şebekeyi kontrol edebilmek ve şebeke servislerini sağlayabilmek amacıyla, operasyon, işletim, bakım ve önlem fonksiyonlarını yerine getirebilmelidir. Günümüzde haberleşme şebekeleri, farklı firmaların farklı teknolojiyle üretilmiş ürünlerinden oluşmaktadır. Böyle bir şebekeyi yönetebilmek için şebeke yönetim fonksiyonlarını yerine getirebilen, standart bir yönetim biçimine ihtiyaç duyulmaktadır. Telekomünikasyon firmaları, sektörde yaşanan rekabete uyum sağlayabilmek için, şebekelerine kolay ve etkili bir şekilde müdahale edebilmeli, bunları yaparken maliyetleri düşük tutabilmelidir. TMN bu ihtiyaçları karşılayan, yazılım ve donanım birimlerinden oluşan, standart bir haberleşme yönetim şebekesidir.

Bu çalışmada TMN'in katmanlı yapısı, bu yapı içinde gerçekleştirilen TMN fonksiyonları, fonksiyonlara hizmet veren servisler ve servis mesajlarını taşıyan protokoller incelenmiştir. TMN fonksiyonlarının kullanımı bir şebeke çeşidi için ele alınmıştır. Diğer şebeke yönetim biçimleriyle karşılaştırılarak TMN'in avantajları ve dezavantajları vurgulanmıştır. Şebeke elemanları üzerinde TMN yazılımının nasıl gerçeklenebileceğini göstermek amacıyla, bir bilgisayar programı yazılmıştır.

TMN, yerel alan ve özellikle geniş alan şebekelerinin yönetiminde tercih edilen bir yönetim biçimidir. Mevcut teknolojileri kullanan şebekelerin sayısının artması ve gelişmekte olan teknolojileri kullanan şebekelerin ortaya çıkmasıyla beraber, TMN standartları da gelişmekte ve yeni ilerlemelere ayak uydurabilmektedir. TMN uygulamaları, yeni ortaya çıkan programlama dilleriyle ve yeni programlama teknikleriyle yapılmakta, çalışma platformundan bağımsız olarak kullanılmaktadır. Bu ilerlemelerin getirdiği kolaylıklar sayesinde, şebeke yönetimi için TMN daha fazla tercih edilen bir şebeke yönetim biçimi olacaktır.

TELECOMMUNICATION MANAGEMENT NETWORK-TMN

SUMMARY

Telecommunication network management, is supposed to realize functions such as operation, administration, maintenance and provisioning, in order to control a network and supply network services. Today telecommunication networks are formed by different products of companies which use different technologies. In order to manage such a network, a standart network management method is needed which can realize network management functions. Telecommunication companies have to manage their networks efficiently and with low cost, so that they can adapt to the competition in sector. TMN is a standart network management method which is formed by software and hardware to satisfy these needs.

In this study, the layered architecture of TMN, functions related to this layered architecture, services that form functions and protocols which carry data of these services are examined in detail. The use of TMN functions are examined for a specific network type. By making a comparision with other types of network management methods, the advantages and disadvantages of TMN are pointed. In order to explain how a TMN software is implemented on a network element, a computer program is written as a part of this study.

TMN is used to be prefered for managing local area and especially wide area networks. According to the development of networks in size and types which are using existing and new technologies, TMN standarts are currently being developed and renewed. TMN applications are programmed by using new programming methods and new programming languages and also they are going to be platform-independent programs. These developments will definitely make TMN more preferable network management method in the future.

1. GİRİŞ

TMN, İngilizce Telecommunications Management Network (Haberleşme Yönetim Şebekesi) kelimelerinin kısaltılmış halidir. ITU-T kuruluşunun, şebeke yönetiminde standartlaşma sağlayabilmek amacıyla ortaya koyduğu, şebeke yönetim protokolleri ve servislerinden oluşan bir şebeke yönetim biçimidir. OAM&P (Operation, Administration, Maintenance & Provisioning), yani bir yönetim şebekesinin fonksiyonları olan operasyon, işletim, bakım ve önlem işlerini gerçekleştirir. TMN, haberleşme şebekesine bağlı birimlerin uzaktan yönetimini sağlayarak, şebeke yönetimi için harcanan maliyetleri önemli ölçüde azaltmaktadır.

TMN uygulamaları, ITU-T standartlarına göre geliştirilmelidir. TMN standardizasyon çalışmaları ilk olarak 1985 yılında ITU-T kuruluşuna bağlı "Çalışma grubu IV" adı verilen grup tarafından başlatılmıştır. İlk TMN standardı 1988 yılında basılan M.30'dur. Daha sonraki yıllarda M üç bin serisi adı altında standartlar çıkmıştır. Bu standartlarda TMN fonksiyonları ve TMN mantıksal katman mimarisi anlatılır. ISDN protokolüne özel olan TMN standartları da tanımlanmıştır. Ayrıca yeni ortaya çıkan teknolojiler için de standartlaştırma çalışmaları yapılmaktadır.

TMN, haberleşme servislerinin verimli ve düzenli çalışması için kullanılan bir yönetim şebekesidir. ATM, SDH, GSM gibi geniş alan şebekelerini yönetebildiği gibi, yerel alan şebekelerini de yönetebilir. Tek bir şebekenin veya entegre edilmiş birden fazla şebekenin yönetiminde kullanılabilir. İnternet kullanımının yaygınlaşmasıyla beraber, kullanıcıların internet üzerinden ses ve görüntünün kaliteli olarak taşınması isteği, telekomünikasyon servis sağlayıcı firmaları bu yönde yatırım yapmaya zorlamaktadır. Bir telekomünikasyon firmasının yönetimi altında, değişik teknolojilere sahip olan haberleşme şebekeleri bulunabilir. Müşterilere sunulan haberleşme servislerinin verilebilmesi için, bu farklı sistem konfigürasyonları entegre edilebilmelidir. Farklı ürünler için farklı yönetim metotlarının uygulanması, yönetimde karmaşıklığa yol açmaktadır. Bir arada kullanılan değişik konfigürasyonların kontrol edilebilmesi için standart bir haberleşme yönetim

şebekesinin kurulması gereklidir. TMN bu ihtiyacı karşılayan bir yönetim şebekesidir.

TMN, katmanlardan, bu katmanlı yapı içerisinde çalışan fonksiyonlardan, fonksiyonları gerçekleştirmek için kullanılan servislerden ve bu servislerle ilgili verileri şebekeye taşıyan protokollerden oluşur. Yazılım açısından bakıldığında, bu birimleri gerçekleyebilmek için oldukça karmaşık programların yazılması gerekebilir. Bu tür programların yazımında kolaylık sağlayan programlama araçlarının yanı sıra, nesneye yönelik programlama tekniklerinin kullanılması, nesnelere daha az sayıda protokol üzerinden erişilmesi gibi yöntemler bu karmaşıklığı azaltmak için kullanılan seçeneklerdir. Ancak, daha basit ve anlaşılır programlar geliştirilebilmesi için uygulanan tekniklere her geçen gün bir yenisi eklenmektedir. Java gibi yeni programlama dilleri ile yazılan ve platformdan bağımsız çalışabilen TMN uygulamaları mevcuttur.

Farklı şebeke yönetim biçimleriyle karşılaştırıldığında, TMN daha karmaşık görünmesine rağmen kullanımı çok daha etkili bir yöntemdir. Kullanılan komut çeşidi ve fonksiyon sayısı açısından diğer yöntemlere üstünlük sağlamaktadır.

Bu tezin amacı TMN kavramını tanıtmak, endüstrideki uygulamaları ortaya koymak ve bir TMN uygulamasının örneğini verebilmektir. Pratikte karşılaşılan zorlukları aşabilmek için bazı yöntemler de sunulmuştur.

İkinci bölümde TMN kavramı ve TMN'i oluşturan birimler üzerinde durulmuştur. Yönetim katmanları ve fonksiyonlarına değinilmiş, TMN'in şebekede yönetilmek istenen yazılım ve donanımlarla kurduğu ilişki incelenmiştir.

Üçüncü bölümde, yönetim için kullanılan yazılım ve donanım çeşitlerinden bahsedilmiş, bu birimlerle nasıl bağlantı kurulduğu açıklanmıştır.

Dördüncü bölümde TMN yönetim servisleri incelenmiş, TMN fonksiyonları ve bu fonksiyonların TMN katmanlarıyla ilişkisi ele alınmıştır.

Beşinci bölümde TMN protokollerinde kullanılan mesajlar incelenmiş, bu protokolleri gerçekleyebilmek için kullanılan programlama dillerinden bahsedilmiştir.

Altıncı bölümde bir ATM şebekesinin, konfigürasyon, hata ve performans fonksiyonları göz önüne alınarak, TMN ile yönetimi anlatılmıştır.

Yedinci bölümde TMN'in diğer yönetim protokolleri ile karşılaştırılması yapılmış, avantajları ve dezavantajları yorumlanmıştır.

Sekizinci bölümde TMN'in yönetilen şebeke elemanlarıyla bağlantısını sağlayan bir ajan uygulama programının nasıl gerçekleştirildiği anlatılmıştır. Bu program tezle birlikte verilen CD'de mevcuttur.

Son bölümde, bu çalışmada açıklanan TMN ile şebeke yönetiminin genel bir yorumu verilmiş, gelecekte şebeke yönetiminde nasıl bir yola gidileceği tartışılmıştır.

2. TMN KAVRAMI

TMN, haberleşme şebekelerini ve servislerini yönetebilmek amacıyla kullanılan bilgileri taşımak, saklamak ve işlemek için kurulan bir haberleşme yönetim şebekesidir[1]. Birden fazla şebeke birbirlerine bağlanarak TMN ile yönetilebilir. TMN ile yönetilebilecek şebekelere örnek olarak analog şebekeler, sayısal şebekeler, bağlaşma sistemleri, iletim sistemleri, haberleşme yazılımı ve şebeke kaynakları olarak adlandırılan elektronik devreler, haberleşme yolları, abone servisleri verilebilir. TMN, ITU-T standartlarına uygun olarak oluşturulması gereken bir şebekedir.

TMN şebekesi oluşturulduğu zaman, sisteme giriş yetkisi olan bir operatör, sistem üzerinde tam bir kontrole sahip olmaktadır. Bu operatörün yapabileceği işler TMN standartlarında tanımlanmış işlevler olmaktadır. Bu işlevler ve TMN şebekesi ile ilgili genel kavramlar bu bölümde anlatılmıştır.

2.1 TMN'in İşlevleri

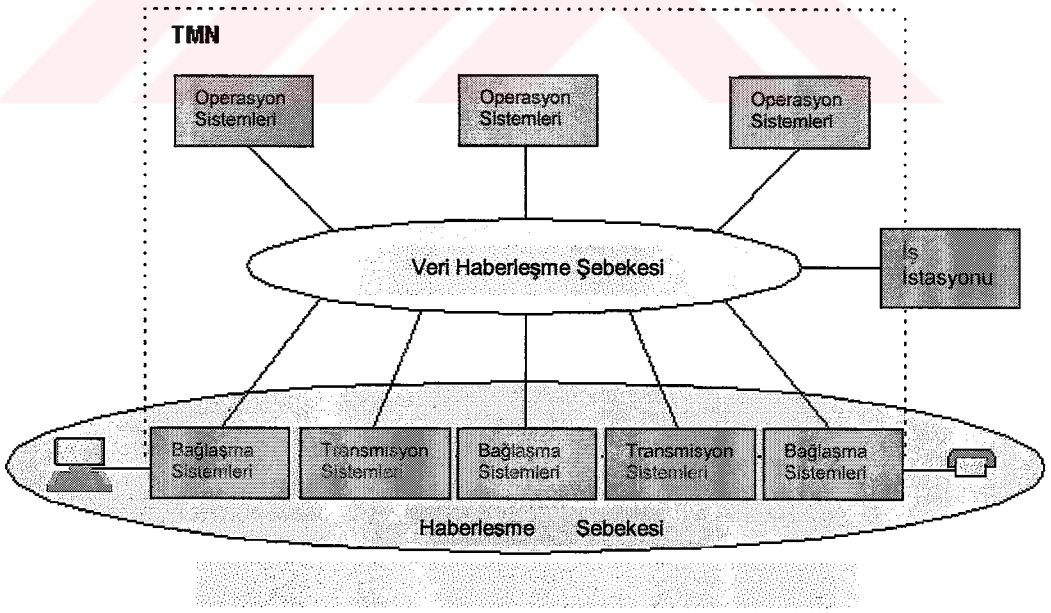
TMN katmanlı bir mimariye sahiptir. Haberleşme şebekesindeki kaynakların gözlem, koordinasyon ve kontrolünü sağlar. Kaynaktan kastedilen şey cihazlar yani donanım, yazılım veya müşteriyle ilgili herhangi bir bilgi olabilir. Yönetilen bazı donanım çeşitleri aşağıdadır[2]:

- Sayısal terminaller
- Özel ve genel olarak kullanılan şebekeler
- Transmisyon terminalleri ve sistemleri
- Dijital veya analog santraller
- Devre bağlaşmalı veya paket bağlaşmalı veri türü taşıyan donanımlar

- İşaretleşme terminal ve sistemleri
- Bağlaşma cihazları

Müşteri ihtiyaçlarını karşılayabilmek için donanımın yanında yazılımın da kontrol edilmesi gerekir.Yazılımla kontrol edilen bazı özellikler şunlardır.

- Abone bilgileri
- Sistemdeki trafik bilgileri
- Yazılımla ilgili link bilgileri
- Alarmlar
- Güvenlik kontrolü
- Sisteme özel uygulamaların kontrolü
- Tarife ve ücretlendirmeler
- Servis kalitesi ve şebeke performansı



Şekil 2.1 TMN'in genel yapısı[2]

2.1.1 TMN Yönetim Katmanları

Haberleşme sistemlerinde karmaşık sistemleri fonksiyonlara bölerek çalıştırmak pratik açıdan tercih edilen bir yöntemdir. TMN'de de aynı yöntem seçilmiş ve katmanlı bir mimari tercih edilmiştir. Her katmanın kendi görevleri ve diğer katmanlarla ortak sağladığı fonksiyonlar vardır. TMN katmanlarına **Fonksiyonel Yönetim Katmanları** adı verilir[2]. Bu katmanlar şunlardır:

- İş Yönetim Katmanı
- Servis Yönetim Katmanı
- Şebeke Yönetim Katmanı
- Eleman Yönetim Katmanı

TMN ile ilgili uygulamalarda bazı kuruluşlar eleman katmanını **şebeke elemanları yönetim katmanı** ve **şebeke elemanları katmanı** olarak ikiye ayırırlar. Bu katmanların görevlerini şu şekilde sıralayabiliriz:

İs Katmanı: Yönetilen şebekenin planlaması bu katmanda yapılır. Şebeke operatörleri arasındaki uzlaşmayı sağlar. Yani şebekenin belli bölümlerinde aynı değerlerin uygulanmasını sağlar. Giriş katmanı olması dolayısıyla ayrıntıların uygulanmasından ziyade şebekenin yönetiminde güdülen amaca genel anlamda hizmet eder. Her katmanın gerçekleştirmesi gereken fonksiyonları vardır. İş katmanı kendi görevlerini yerine getirmek için alt katmanlarla bilgi alışverişi yapmak zorundadır. Aslında bu her katman için geçerlidir.

Servis Katmanı: Servis katmanı genel olarak müşteriyle ilgili görevleri üstlenir. Bu işlemler yeni abone profili oluşturmak ve kapatmak, müşteri şikayetlerini çözümlemek ki bu fatura bilgilerini, aboneye has hata raporlarını ve iletilen verinin servis kalitesine uygun olmasını temin etmektir. Bu katmanın fonksiyonları fiziksel birimlerin bakımından sorumlu değildir.

Şebeke Katmanı: Bu katmanı şebeke yöneticisinin yöneticisi olarak adlandırabiliriz. Eleman katmanından aldığı bilgileri şebekenin genel durumunu ortaya koyacak

şekilde düzenler. Eleman katmanıyla sıkı bir ilişki içerisindedir. Şebeke içerisinde bulunan daha küçük boyuttaki şebekelerin kontrolünden de sorumludur.

Eleman Katmanı: Bu katman yönetim katmanlarının işçisidir. Ajan gibi davranır. Şebekeye bağlı fiziksel birimler üzerinde çalışır. Sistemin performans bilgilerinin toplanması, şebeke üzerindeki elemanların performanslarının kontrolü, alarmlarla ilgili verileri toplama ve alarmları sezme, trafik bilgilerinin toplanması, adres ve protokollerin istenilen şekle çevrilmesi ve veri analizi gibi işlevlere sahiptir.

2.1.2 TMN Yönetim Fonksiyonları

Bütün bir sistemin yönetimini sağlayabilmek için fonksiyonel bir yaklaşım uygulanır. Yönetimi sağlayabilmek amacıyla oluşturulan fonksiyonlara TMN yönetim fonksiyonları adı verilir. Bu fonksiyonlar TMN yönetiminin yapması gereken işleri tanımlar. Bu fonksiyonlar ve görevleri şu şekilde tanımlanır:

Konfigürasyon Yönetim Fonksiyonu: Yönetilen kaynakların izlenmesi ve kaynaklarla ilgili ayrıntıların gözlenmesi görevini üstlenir. Burada kullanılan konfigürasyon kelimesi, kaynak bilgilerini konfigüre etmekten daha fazla anlam taşır. TMN anlamında konfigürasyon yönetimi sistemin gözlenmesi, sistemdeki kaynakların yerlerinin belirlenmesi yani topoloji yönetimi, yazılım yönetimi ve envanter yönetimi anlamına gelir.

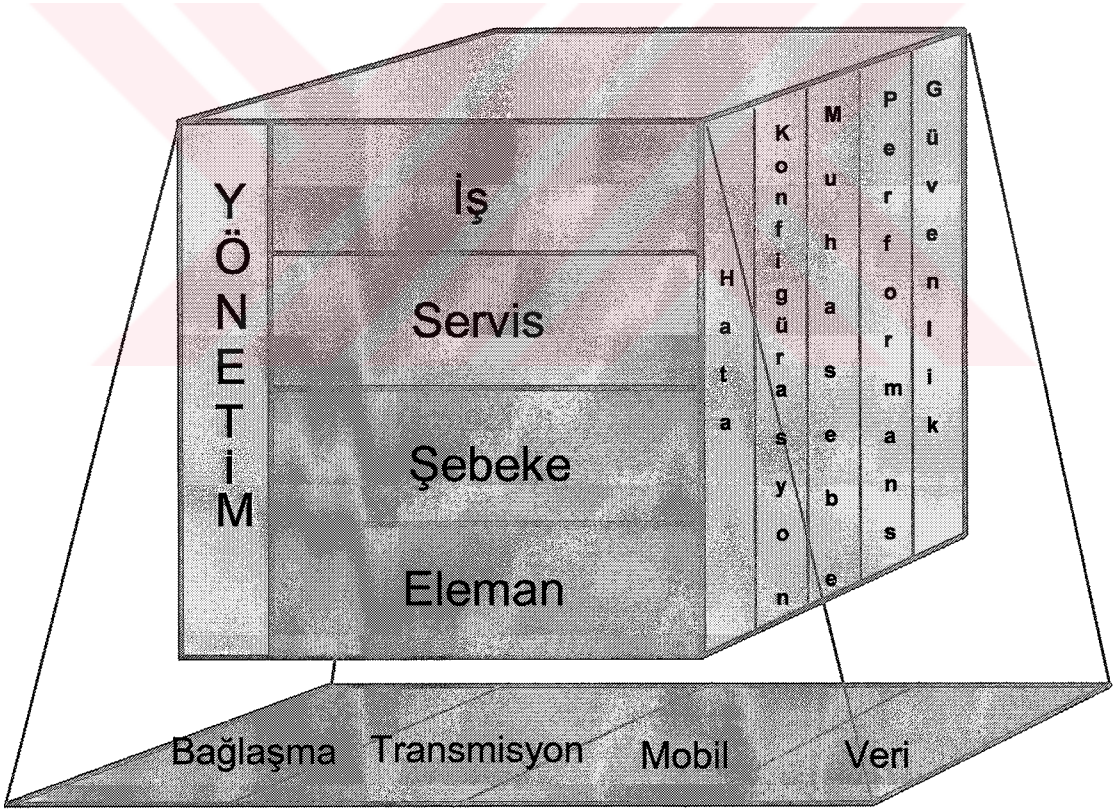
Hata Yönetim Fonksiyonu: Haberleşme yönetiminin sıra dışı davranışlarını sezme, hataları düzeltme ve analiz etme görevlerini üstlenir. Hata yönetimi hataların tarihçesini de gözlemek zorundadır. Problemlerin raporlanması, sezilmesi ve problem takibi görevlerini karşılar.

Performans Yönetim Fonksiyonu: Sistem performansının gözlenmesi ve gerektiği yerlerde müdahale edilmesi işlevini yerine getirir. Performans verilerinin toplanması, analizi, performansla ilgili aksaklıkların raporlanması, kaynakların verimliliğinin ölçülmesi görevlerini üstlenir.

Muhasebe Yönetimi: Kaynakların kullanımından doğan maliyetlerin hesaplanması görevini üstlenir. Ücretlendirme işini yapar.

Güvenlik Yönetimi: Güvenlik ihlallerini sezme, izleme ve raporlama görevlerini üstlenir. Şifreleme, şifre anahtar kontrolleri ve erişim kontrollerini sağlar. Sistemde kullanılan parolaların dağıtılması görevini üstlenir.

Yönetici ve ajan, şebeke yönetimi açısından önemli iki kavramdır. Yönetici tüm sistemin operatörüdür. Ajanlar kaynaklar üzerinde bulunurlar ve yöneticiden gelen mesajları cihazlara iletirler. Bu cihazlara genel anlamda kaynak diyoruz. Ajanlar kaynaklarda meydana gelen olayları da yöneticiye bildirirler. Yönetici farklı ajanların gözlenmesi ve kontrolü açısından anahtar bir rol üstlenir. Ajanlar yöneticiden gelen komutları yönetilen kaynakların anlayabileceği mesajlara çevirirler. Yönetici ve ajanlar yönetim bilgileri denilen standart olarak tanımlanmış bilgiler vasıtasıyla anlaşırlar. TMN ve veri haberleşme şebekeleri, yönetilen kaynakların gösterimi ve bilgi alışverişi için nesneye yönelik bir program yapısına sahiptir.



Şekil 2.2 Katman ve fonksiyon yapısı

2.1.3 Yönetici ve Ajan Kavramları

Yönetici ajanlardan bilgi toplamak için istediği kaynak bilgileriyle ilgili verileri ajanlara gönderir. Ajanlar bu kaynaklarla ilgili bilgileri kaynaklardan toplarlar ve tekrar yöneticiye onay mesajlarıyla iletirler. Bu işlemler sırasında herhangi bir alarm oluşursa ajanlar uyarı mesajlarıyla bunu yöneticiye bildirirler. Bu uyarı mesajlarına olay raporu adı verilir. Yönetici ve ajanlar arasındaki mesajlar standart yönetim protokolleri aracılığıyla iletilir. Bu protokoller bölüm 5'de ayrıntılı bir şekilde anlatılmıştır.

Yönetici ve ajanlar arasındaki mesajlar standart bir arabirim tarafından iletilir. Genel olarak bu arabirim OSI (Open System Interface) arabirimidir. Ancak TCP/IP (Transmission Control Protocol/Internet Protocol) ve IPX/SPX (Internet Packet Exchange/Sequenced Packer Exchange) gibi arabirimler de kullanılabilir.

Yönetici uygulamaları Sun firmasının Solaris, Hewlet Packard'ın HP-UNIX, Microsoft'un Windows NT, Windows 95 ve sonrası işletim sistemleri üzerinde kurulabilir. Endüstriyel uygulamalarda genellikle Windows işletim sistemleri tercih edilmektedir.

2.1.4 Yönetilen Nesne, Sınıf ve Bilgi Modeli Kavramları

2.1.4.1 Yönetilen Nesne

TMN'de yönetilen birimlere kaynak diyoruz. Kaynaklar fiziksel ve mantıksal kaynaklar olarak sınıflandırılabilir. Örneğin bir PBX, PBX içindeki bir abone kartı veya bir yazıcı fiziksel kaynaklardır. Kaynak üzerinde çalışan yazılım uygulamaları veya kayıtlar için tutulan dosyalar mantıksal kaynaklardır. TMN anlamında bütün bunlar yönetilen nesnelerdir. Yani ajan yazılımında şebeke üzerinde yönetilecek olan kaynaklar temsili olarak yazılımda yönetilen nesneler şeklinde gösterilir. Bir kaynak birden fazla nesne şeklinde gösterilebilir. Bir nesne kaynaklar arasındaki ilişkiyi göstermek için de kullanılabilir. Örneğin numara 7 sinyalleşmesi için kullanılan linkler birer nesne olarak ajan yazılımında oluşturulur. Yazılım anlamında bir nesnenin içeriğindeki veriler şu şekilde tanımlanmıştır[4]:

Özellik (Attribute): Bir nesne içerisinde birden fazla özellik bulunabilir. Özellikler kaynağın işlevlerini tanımlar. Örneğin abone kartının tipi bir özellik tarafından gösterilebilir (analog veya sayısal).

İşlem (Operation): Yönetici tarafından ajan üzerinde yapılan işlemlerdir. Bir nesne yaratmak veya bir özelliğin değerini almak birer işlemdir. Örnek olarak abone profili yaratmayı verebiliriz.

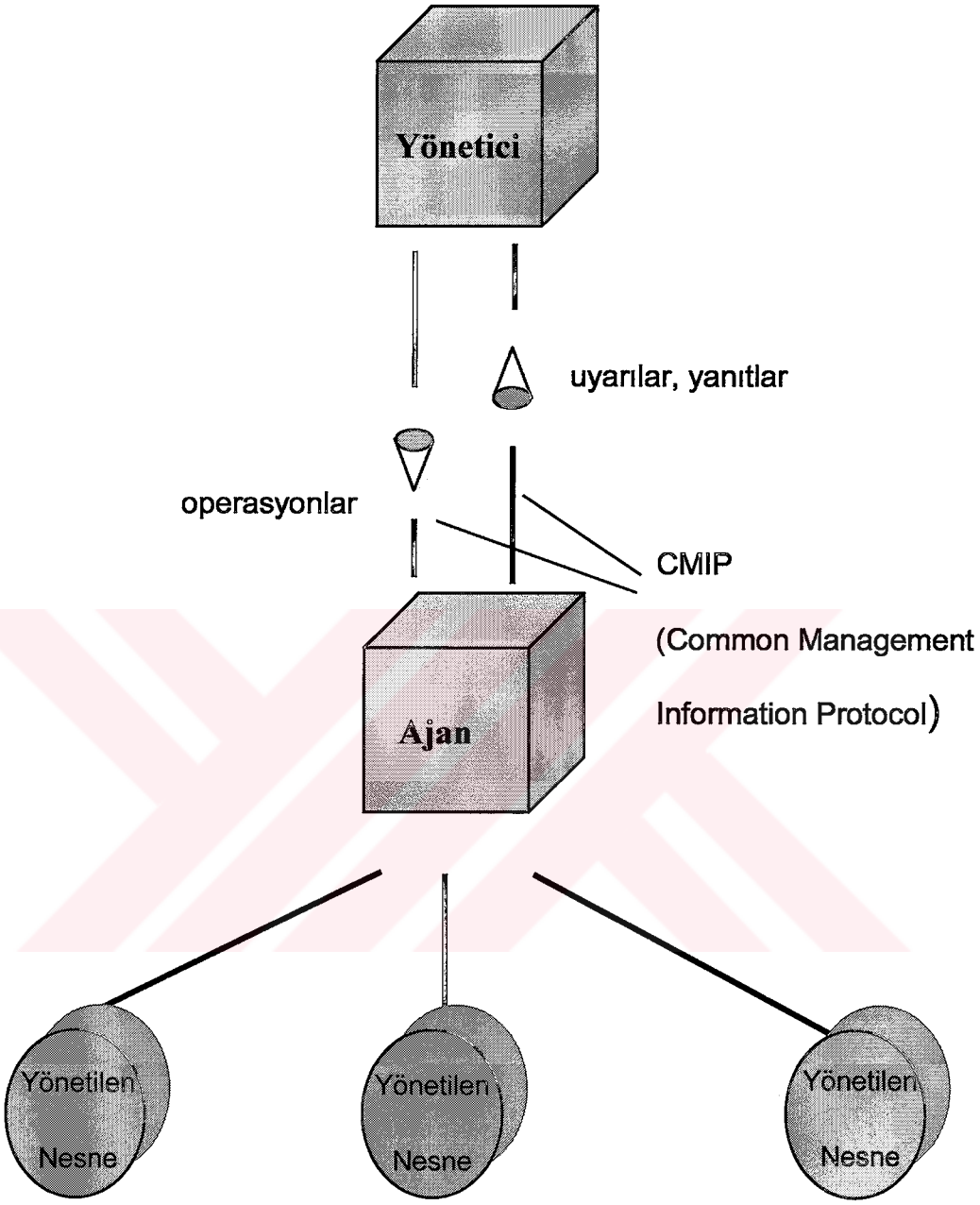
Uyarı (Notification): Ajan tarafından yöneticiye gönderilen bir mesajdır. Mesajda uyarının sebebi, nerde oluştuğu ve kimle alakalı olduğu belirtilir. Örneğin bir santralde abone kartında oluşan bir alarm için uyarı mesajı atılır.

Davranış (Behaviour): İçerisinde özellik, işlem ve uyarı bilgileri bulunan bir nesne tanımlandığında, o nesnenin işlevini anlatmak için davranış bilgileri de nesne içerisine dahil edilir. Davranış sözel bir açıklamadır.

2.1.4.2 Yönetilen Nesne Sınıfı

TMN anlamında sınıf kavramı aynı özellikleri taşıyan yönetilen nesneler için kullanılan ortak bir bildirimdir. Örneğin, bir santral üzerinde bir çok kart çeşidi kullanılır. Kartlar birer yönetilen nesnedir ve ajan yazılımı üzerinde gösterilirler. Kartlar için kullanılan tek bir sınıf vardır. Bu sınıf kullanılarak kart nesneleri üretilir. Ancak bu nesnelerin özellikleri kartları ayırt edecek şekilde düzenlenir. Genellikle bu özelliklere kart isimleri yazılır.

Sınıflar X.720 standardında tanımlanan Yönetim Bilgi Modeli'ne göre düzenlenir. Sınıfları tanımlamak için GDMO (Guidelines for the Definition of Managed Objects) ve ASN.1 (Abstract Syntax Notation 1) diye adlandırılan sınıf kalıbı tanımlama dilleri kullanılır. GDMO X.722[14], ASN.1 X.208[9] standardında tanımlanmıştır. Bu dillerle ilgili bilgiler 5. bölümde verilmiştir.



Şekil 2.3 Yönetici ve Ajan arasındaki ilişki

2.1.4.3 Bilgi Modeli

Yönetim sınıflarını ve bu sınıflar arasındaki ilişkiyi tanımlayan yapıya bilgi modeli adı verilir. Bilgi modeli standart nesneleri ve de kullanıcı tarafından sağlanan, yönetilen cihazlarla ilgili nesneleri tanımlar. Bilgi modeli iki şekilde tanımlanabilir. İlki jenerik bilgi modeli, ikincisi özelleştirilmiş bilgi modelidir. Jenerik bilgi modeli standart nesneleri tanımlar. Spesifik bilgi modeli, kullanılan arabirimi destekleyen nesneleri tanımlar. Örneğin OSI arabirimi olan Q3 arabirimini kullanıyorsak, bu arabirimi destekleyen nesneler spesifik bilgi modelinde tanımlanır.

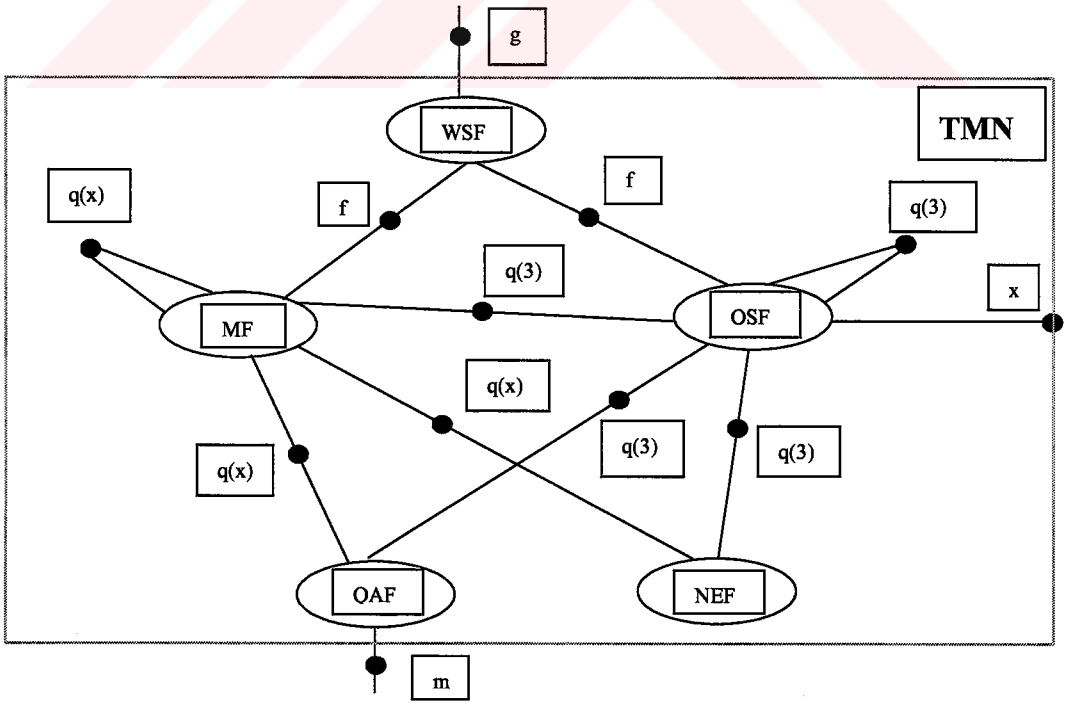
TMN modeli nesneye dayalı bir modeldir. Bilgi modeli sınıflardan oluşur. Sınıflar özellik, işlem, uyarı ve davranışlardan meydana gelir. Sınıflar arasında hiyerarşik bir ilişki söz konusudur. Örneğin bir santral üzerindeki elektronik kart gösterilmek istendiğinde, takılı olduğu rafı temsil eden nesnenin altında yaratılmalıdır.

3. TMN'İN GENEL YAPISI

TMN kavramı bir çok fonksiyonel birimin bir arada çalışması esasına dayanır. TMN standartlarında çokça kullanılan bu terimlere burada değinmek gerekir. TMN mimarisi denildiğinde bir çok farklı işlevi olan birimin bir arada çalışması ile oluşan bir çalışma sistemi söz konusudur. Bu yapıyı üçe ayırabiliriz: Fonksiyonel yapı, fiziksel yapı, veri yapısı. Bunlar haricinde bağlantı yapısı da bunlara eklenebilir.

3.1 Fonksiyonel Yapı

TMN bir arada çalışan bir çok fonksiyonun gerçekleştirilmesiyle oluşan bir yapıdır. Bu fonksiyonlar farklı şekillerde birbirleriyle ilişkilendirilerek karmaşık işlemlerin gerçekleştirilmesi sağlanabilir. Aşağıdaki şekil bu fonksiyonel yapıyı ve ara bağlantılarını göstermektedir[2].



Şekil 3.1 TMN'in fonksiyonel yapısı

Yukarıdaki şekilde fonksiyon blokları birbirlerine çeşitli arabirimlerle bağlanmaktadır. Bu fonksiyon bloklarının ve ara birimlerin açıklamaları şu şekildedir.

OSF, Operasyon Sistem Fonksiyonları, haberleşme şebekesinin yönetimi için gerekli olan yönetim ve planlama fonksiyonlarını sağlar. TMN katmanlarının her biri için tanımlı bir veya daha fazla OSF bulunur. Örneğin iş katmanının kullanabileceği bir OSF fonksiyonu mevcuttur ve iş katmanı OSF diye isimlendirilir. OSF'ler bağlantı kurulan diğer uçlardaki OSF'lere x arabirimi ile ulaşır. Kendi alt fonksiyonlarına ise q(3) arabirimi ile ulaşır[15-16]. X arabirimi yöneticilerin birbirlerine bağlanmasını sağlayan bir arabirimdir. Örneğin A haberleşme şebekesinin yönetimini A1 yöneticisi yapıyorsa, B haberleşme şebekesinin yönetimini B1 yöneticisi yapıyorsa, A1 B1'e x arabirimi ile bağlanır[6].

Q3 arabirimi ise OSI katmanlarının hepsini kullanan bir arabirim çeşididir. TMN uygulamalarında yaygın bir şekilde kullanılmaktadır.

NEF, Şebeke Elemanları Fonksiyonu, şebeke üzerindeki elemanları yani kaynakları temsil eder. WSF, İş İstasyonu Fonksiyonu, yönetim bilgilerinin kullanıcı tarafından görülmesini sağlayan fonksiyondur. Bunun anlamı yönetim bilgilerinin f arabiriminden g arabirimine çevrilmesidir. F arabirimi diğer fonksiyonların iş istasyonlarına bağlanmasını destekleyen bir arabirimdir. G arabirimi insanlarla iş istasyonundaki verilerin kullanıcı tarafından görülebilmesini sağlar.

MF, Aracı Fonksiyonu, diğer fonksiyonlar arasında değişik türde arabirimler kullanılıyorsa bunları dönüştüren fonksiyondur. QAF, Q Adaptör Fonksiyonu, TMN ile TMN olmayan şebekeler arasında yapılan veri alışverişi QAF üzerinden yapılır. Bunun için m arabirimi kullanılır.

Burada bahsedilen fonksiyon kavramı fonksiyon bloklarını ifade eder. 3. bölümde bahsedilen fonksiyonlar bu fonksiyon bloklarını oluşturur. Bu fonksiyon bloklarını oluşturan fonksiyonel birimler yönetici ve/veya ajanın çalışmasını sağlayan fonksiyonlardır.

3.2 Fiziksel Yapı

Fiziksel yapıdan kastedilen yapı yukarda anlatılan fonksiyon bloklarını fiziksel ortamlarda gerçekleştirme şeklidir. Yani fonksiyon bloklarının çalışacağı fiziksel birimler kastedilir. Bu bölümde bu fiziksel birimlerin görevleri anlatılmaktadır. Yukarıdaki şekilde görüldüğü gibi, fonksiyon blokları birbirlerine arabirimlerle bağlanmalıdır.

Fiziksel birimlerden Operasyon Sistemi (OS), yönetici yazılımının büyük kısmını üzerinde barındıran bir fiziksel cihazdır. Bu cihaz üzerinden şebekenin çalışması ile ilgili işlemler, bakım işlevleri ve şebekenin genel durumu hakkında bilgi edinme gibi işlemler yapılabilir. Normalde bir TMN sisteminde bir tane OS bulunur. Ancak üstlenilecek yük ağırsa birden fazla OS bulunabilir. Örneğin alarm toplama gibi bir görevi ayrı bir OS üstlenebilir.

Veri haberleşme şebekesi cihazları fiziksel yapının bir parçasıdır. Bu cihazlar verileri taşımak ve yönlendirmekle yükümlüdür. Örnek olarak, köprü (bridge), yönlendirici (router) gibi cihazları gösterebiliriz.

Aracı cihazlar protokol çevrimi, mesaj çevrimi, adres dönüştürme ve yönlendirme görevlerini üstlenirler. Veri depolama ve filtreleme gibi görevler de verilebilir.

İş istasyonları operatörün sisteme giriş çıkışını sağlayan cihazlardır. Bu sistemler genellikle üzerinde UNIX işletim sistemi taşıyan bir iş istasyonu veya Windows NT işletim sistemi olan bir kişisel bilgisayardır.

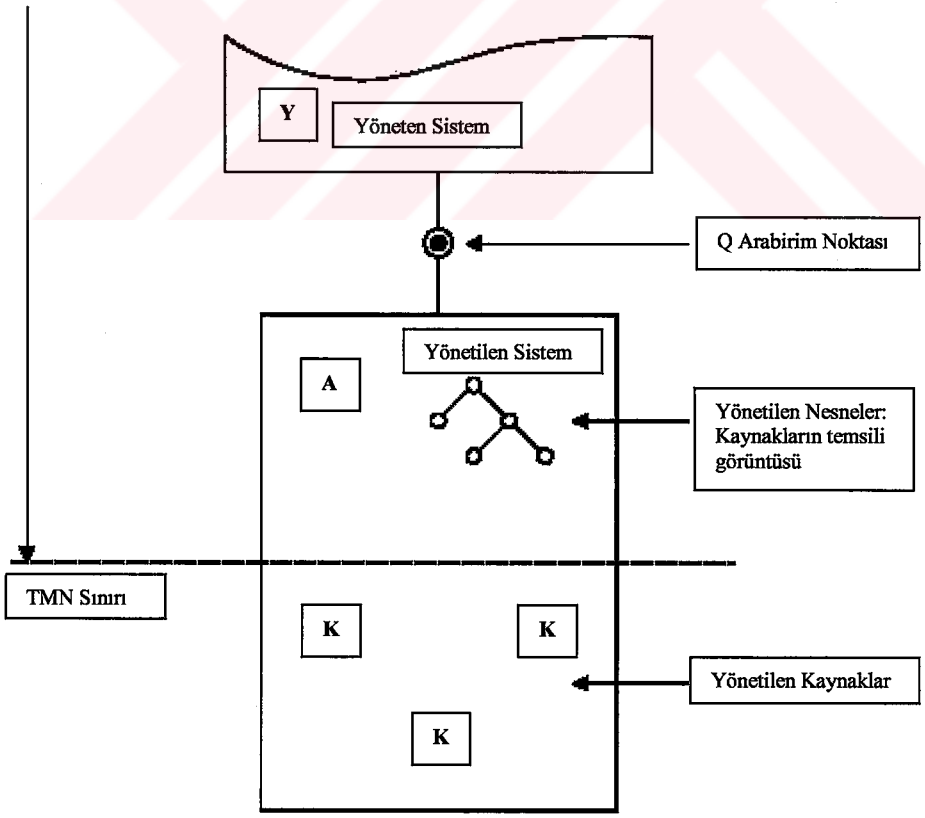
Şebeke elemanları haberleşme sisteminin haberleşmeyi sağlayan elemanlarıdır. Bir şebeke elemanı, santral üzerindeki bir kart, bir güç kaynağı veya sayısal terminal olabilir. Bu cihazlara ajan yazılımı vasıtasıyla erişilir. Ajan yazılımı şebeke elemanlarından birinin üzerinde bulunur. Bu eleman genellikle ana kart olarak adlandırılan bir eleman olmalıdır. Ancak farklı birimler üzerinde de ajan yazılımı gerçekleştirilebilir.

3.3 Veri Yapısı

TMN yönetiminin veri yapısı, nesne tanımlama diliyle ifade edilen nesne modelini içerir. Nesne modeli, yönetilmesi istenen şebeke elemanlarının yani kaynakların nesne sınıfı olarak tanıtıldığı bir veri dosyasıdır. TMN nesneye dayalı bir veri yapısına sahiptir. Örneğin yazılımda bir santral üzerindeki kart temsil edilmek isteniyorsa nesne modelindeki kart sınıfı kullanılarak bir kart örneği temsili olarak yaratılır. Bu kartla ilgili işlemler bu örnek üzerinden yürütülür.

Nesne modelindeki sınıfların tanıtılması için iki adet programlama dili kullanılır. Bu diller özel bir nesne tanımlama dili olan GDMO ve ASN.1'dir. Bu dillerin kullanımı ile bilgiler 5. bölümde verilmiştir.

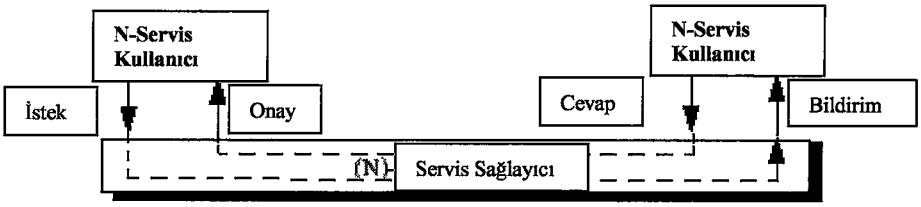
Nesne modelinin yönetici ve ajan tarafından bilinmesi gerekir. Kaynaklarla ilgili mesajlar nesne modeli kullanılarak CMIP protokolüyle yönetici ve ajan arasında taşınır.



Şekil 3.2 Kaynak yönetimi

3.4 TMN Bağlantı Yapısı

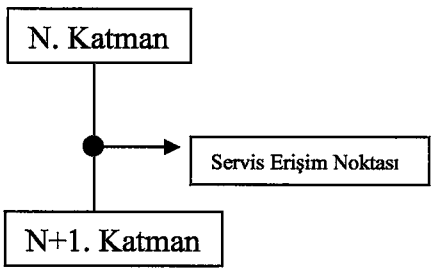
TMN genellikle OSI katmanları üzerine kurulan bir sistemdir. TMN uygulamaları 7. OSI katmanı olan uygulama katmanında bulundurulur. TMN yapısının birer parçası olan yönetici ve ajanlar, şebeke katmanından itibaren katman katman bağlantı kurmak durumundadır. Bu bölümde bağlantı kurmak için kullanılan yöntemler açıklanacaktır.



Şekil 3.3 Servis mesajları

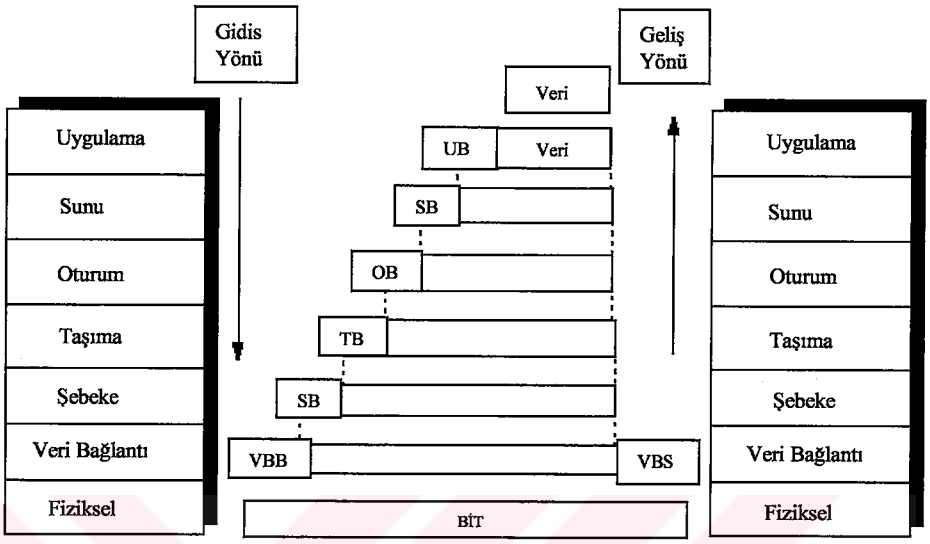
Yukarıdaki şekilde iki OSI katmanı arasındaki ilişkiyi görüyoruz. Servis kullanıcı ve servis sağlayıcı birer katmanı ifade eder. Örneğin uygulama katmanı sunu katmanı için servis kullanıcıdır. Sunu katmanı da uygulama katmanı için bir servis sağlayıcıdır. Bu katmanlar arasındaki haberleşme mesajlar sayesinde olur. Bir servis kullanıcı istek gönderir ve karşı servis kullanıcı bunu bir bildirim mesajı olarak alır. Bu bildirime bir cevap gönderir ve istekte bulunan taraf bu mesajı bir onay mesajı olarak alır. Onay mesajı olumlu anlam taşımayabilir.

Katmanların birbirleriyle bağlantı kurabilmesi için birbirlerine belirlenmiş noktalardan mesaj atmaları gerekmektedir. İki katman arasında geçit görevi görecek bir adrese ihtiyaç vardır. Bu adresin adı Servis Erişim Noktası'dır (SAP-Service Access Point). Bu adres belirli bir isim veya numara olabilir. Örneğin uygulama katmanından itibaren şebeke katmanına kadar "psel, ssel, tsel, 2606558" gibi bir adresleme yapılabilir.



Şekil 3.4 Servis erişim noktası

Şekil 3.5'te bu adreslerin OSI katmanlarında nasıl taşındığı görülmektedir. Mesaj yukarı katmanlardan aşağıya doğru giderken her katman kendi adresini mesajın başına ekler. Mesajın gönderildiği tarafta da mesaj yukarı katmanlara doğru giderken her katman kendi adresini mesajdan çıkararak yukarı katmana aktarır.



UB Uygulama Başlığı
SB Sunu Başlığı
OB Oturum Başlığı
TB Taşıma Başlığı
ŞB Şebeke Başlığı
VBB Veri Bağlantı Başlığı
VBS Veri Başlık Sonu

Şekil 3.5 OSI katman başlıkları

Yapılan bağlantının şekli bağlantılı veya bağlantısız olabilir. Bağlantılı durumda önce bağlantı kurma isteği bağlantı kurulacak taraf gönderilir. Olumlu cevap alınırsa veri transferine geçilir. Bağlantı sona erdirilmek istendiğinde bağlantı çözme isteği taraflardan herhangi biri tarafından gönderilir. Bağlantısız durumda ise sadece bağlantı kurulacak tarafın adresi ve gerekli parametreler gönderilerek veri iletimi yapılır.

4. TMN YÖNETİM SERVİSLERİ VE FONKSİYONLARI

Temel TMN fonksiyonları M-3000 standart serilerinde tanımlanmıştır. Bu standartlarda TMN mimarisi, TMN için kullanılan arabirimler, yönetim servisleri ve fonksiyonları tanımlanır. Mimari yapı ve arabirimler 3. bölümde anlatılmıştır. Bu bölümde TMN yönetim servisleri ve bunların yerine getirdiği fonksiyonlar ele alınacaktır.

TMN yönetim servisleri ITU-T kuruluşunun M.3200 standardında, TMN yönetim fonksiyonları M.3400 standardında anlatılmıştır. Bunlar arasındaki ilişkiye M.3020 standardında değinilmiştir. Bu standartlar tavsiye (recommendation) şeklinde yazılmıştır. Bu çalışmada referans olarak bu standartlar kullanılmıştır.

TMN yönetim servisleri kullanıcı göz önüne alınarak kullanılan servislerdir. TMN yönetim servisleri ile fonksiyonlar arasında hiyerarşik bir yapı mevcuttur. Servisler fonksiyon gruplarının bir araya gelmesinden oluşur. Bir fonksiyon grubu da bir kaç fonksiyonun bir araya gelmesinden oluşur.

Bu servislerin fonksiyonlarının yönetilen sistemlere iletilebilmesi için, servislerin sistem yönetim fonksiyonlarına birebir eşlenmesi gerekir. Sistem yönetim fonksiyonları yönetim protokolü ile gönderilen operasyonlar ve uyarılar vasıtasıyla yönetim fonksiyonları üzerinden yönetim servislerinin çalışmasını sağlarlar. Yönetim protokolü CMIP (Common Management Information Protocol)'dır. CMIP protokolü CMISE (Common Management Information Service Entity) adı verilen servis tarafından sağlanan bir protokoldür. Bu protokole ve mesajlarına 5. bölümde değinilecektir.

4.1 Yönetim Servisleri

Yönetim servislerini şu şekilde sıralayabiliriz[5]:

Abone Yönetim Servisi: Müşteri ihtiyaçlarını karşılamak için kullanılan bir servistir. Bu yönetim servisi servis hazırlık yönetimi, konfigürasyon yönetimi, hata yönetimi, ücretlendirme yönetimi, servis kalite yönetimi, trafik ölçüm yönetimi gibi işlevleri içerir.

Şebeke Hazırlama Yönetim Servisi: Müşteriye sunulacak yeni kaynaklar veya servisler için şebeke kapasitesinin genişletilmesi servisini sunar.

İş Gücü Yönetim Servisi: Sahada çalışan görevlilerin bakım, onarım ve cihazların kurulması gibi işlere atanması servisini verir. Ayrıca müşterinin ihtiyacına göre çalışanların müşteriyle ilişkilendirilmesi sağlanır.

Tarife, Ücretlendirme, Muhasebe Servisi: Müşterilerin ücretlendirilmesi görevini üstlenir. Müşterilerin faturayla ilgili şikayetleri, ödenmemiş faturalarla ilgili bilgileri değerlendirir. Bu servis müşterinin verilen servisleri ne kadar kullandığını belirler ve müşteriye verilen servis için kullanılan kaynakların testini yapar.

Servis Kalitesi ve Şebeke Performansı Yönetim Servisi: Haberleşmede müşteriye güvenilir bir haberleşme şebekesinde yüksek kalitede transmisyon hizmeti vermek esastır. Şebeke üzerinde yapılan analiz ve testlerle şebekenin kalitesi korunur. Bu servisin bazı fonksiyonları trafik kalitesi güvencesi, performans kalitesi güvencesi, güvenilirlik, uygunluk ve kurtarılabilme güvencesi gibi fonksiyonlardır.

Trafik Ölçme ve Analizi Yönetim Servisi: Bazı durumlarda şebeke üzerindeki trafik beklenenden daha fazla veya daha az olabilir. Sisteme eklenen bazı servisler trafiği arttırabilir. Şebeke üzerindeki trafiği planlayabilmek için bu servis kullanılır.

Yön Bulma Ve Dijit Analizi Yönetim Servisi: Yönlendirme tablolarının düzenlenmesi ve değiştirilmesi işlerini yapar.

Bakım Yönetim Servisi: Şebeke üzerinde bir arıza oluştuğunda bunun için bir dosya açılır. Problemin çözümlenme aşaması bu dosya sayesinde gözlenir. Bakım yönetim

servisi düzenli sistem bakımı, şebeke üzerindeki problemlerin sezilmesi, lokalize edilmesi ve düzeltilmesi işlerini yapar.

Güvenlik Yönetimi Servisi: Santrallerle, abonelerle, faturalarla ilgili bilgilerin güvenliği sağlanır. Güvenliğin aşılması sezilir ve bu tip olayları önlemek için önlemler alınır.

Lojistik Yönetimi Servisi: Haberleşme şebekesinde gerekli olan yerlerdeki ekipmanların yenileriyle değişimi ve yeni servislerin verilmesiyle ilgili servistir.

4.2 Yönetim Fonksiyonları

Bu bölümde yönetim fonksiyonlarının şebeke yönetimine nasıl etki ettiği incelenmektedir. Yönetim fonksiyonları performans, hata, konfigürasyon, muhasebe ve güvenlik fonksiyonlarından oluşur.

4.2.1 Performans Yönetimi

Performans yönetim fonksiyonunun başlıca görevlerinden biri, haberleşme şebekesi üzerinde taşınan, performansla ilgili verilerin toplanmasıdır. Kaynaklardan toplanan veriler bir yöneticide veya operasyon biriminde toplandığında analiz edilir. Eğer sistemde sıkışmaya yani bir dar boğaza yol açan bir durum varsa bu durum sezilmeli ve hatta düzeltilmelidir. Performans yönetimi içinde trafik yönetimi de vardır. Bu yönetim fonksiyonuyla ilgili sınıflar ITU-T Q.823 tavsiyesinde tanımlanmıştır. Trafik yönetimini sağlayabilmek amacıyla yazılan uygulama yazılımları bu referansı kullanmak durumundadır.

Performans yönetim fonksiyonu aynı zamanda servis kalitesi ile ilgili verileri de haberleşme şebekesinden toplamak zorundadır. Toplanan veriler değerlendirildikten sonra şebeke üzerinde gerekli iyileştirmeler de yapılmak zorundadır.

Servis kalitesinin değeri veri iletim hızı, sistemin güvenilirlik oranı, bilgi depolama hatası, bağlantı kopması gibi bilgilerin hesaplanmasıyla ortaya çıkar. Servis kalitesiyle ilgili verilerin bir kısmı şu şekilde sıralanabilir:

- Abonelerin bağlantı isteği
- Aboneler arasındaki bağlantı kalitesi
- Ücretlendirmedeki doğruluk
- Sistemin çeşitli durumlardaki tarihçesinin kolaylıkla incelenebilmesi
- Hata yönetimiyle kurulan ilişki sayesinde kaynaklarda oluşan hataların bulunması
- Konfigürasyon yönetimiyle yapılan bilgi alışverişiyle haberleşme yollarının değiştirilmesi
- Servis kalitesi parametrelerinin gözlenmesi için testler uygulanması

Burada dikkat edilmesi gereken bazı durumlar söz konusudur. Eğer performans verileri kısa aralıklarla toplanmaya çalışılırsa bu işlem şebeke üzerindeki trafiği bizzat arttıran bir işlem olacaktır. Aksi durumda eğer yeterli performans verisi toplanmazsa bu verinin yeteri derecede faydası olmayacaktır. Bu durumda performans verilerinin toplanması için iyi bir çizelge hazırlanması gerekliliği ortaya çıkar. Örneğin bir santral üzerinde saatlik, günlük, haftalık ve aylık trafik bilgilerinin toplanması gerekliliği vardır. Bu konuda yazılım tasarımı yapılırken en kısa sürede en fazla verinin toplanması hedef alınmalıdır. Ajan yazılımları bu görevi yerine getirebilmelidir.

Performans yönetiminde dört ana fonksiyon grubu vardır:

Performans Kalitesi: Performans kalitesinin güvenilirliğini sağlamak performans yönetim fonksiyonunun görevidir. QoS amaçlarına ulaşılmalıdır. Bu sebeple şebeke performansı için istenilen koşullar yerine getirilmelidir. Aboneye verilen servisin kalitesi, işletme açısından önemsenmesi gereken bir unsurdur.

Performans İzleme: Bu amaçla şebeke hakkında düzenli olarak bilgiler alınmalıdır. Bu bilgiler performans amacı güderek, sistem, şebeke ve verilen servisler hakkında olmalıdır.

Hata yönetimindeki alarm sezme göreviyle performans verileri toplama görevi ayrı görevlerdir. Alarmların sezilmesi için durumun kronik bir seviyede olması gerekir. Oysa performans verileri en düşük seviyedeki verileri gösterir. Bahsedilen performans bilgileri sistemin trafiği ile ilgili bilgilerdir. Bu bilgiler depolanarak değerlendirilir.

Performans Yönetim Kontrolü: Sistemdeki trafiğin kontrol altında tutulabilmesi için operatörün müdahale edebilmesi gerekir. Bu amaçla sisteme yapılan bilgi girişleri performans yönetiminin görevidir. Örneğin haberleşme şebekesinde bulunan sistemler arasındaki bant genişliğini kontrol edebilmek bir gerekliliktir. Bu kontrolü operatörün yapabilmesi gerekir.

Performans Analizi: Sistemin mevcut durumda taşıdığı trafik göz önüne alınarak, daha sonra meydana gelmesi muhtemel sıkışıklıkların engellenmesi için performans analizi gereklidir. Bu analizler abonenin trafik performansı ve sistemin trafik kapasitesi göz önüne alınarak yapılır.

4.2.2 Hata Yönetimi

Hata yönetimi haberleşme şebekesinin düzgün çalışmasını etkileyen sıra dışı durumların sezilmesinden ve bu durumların giderilmesinden sorumludur. Şebeke üzerindeki cihazlardan gelen alarmlar yöneticide toplanır ve saklanır. Yönetici gerekli müdahalelerde bulunarak bu hataların giderilmesinden sorumludur. Hata yönetimi altı adet fonksiyon grubu içerir:

Güvenilirlik Kalitesi: Haberleşme şebekesindeki cihazların ve kullanılan yazılımın en az hatayı yapması hedef olarak alınmıştır. Bu amaçla verilen servislerin ve şebeke üzerinde kullanılan cihazların güvenilir olması servis sağlayıcının görevi olmalıdır.

Alarm Denetimi: Alarm denetimi gerçek zamanda sistemin gözlenmesi ve şebeke elemanlarının sorgulanmasını sağlar. Bir şebeke elemanında hata oluştuğu zaman, ajan yazılımı bu hatayı, sebebi belliyse belirterek yöneticiye raporlar. Eğer ajanla yönetici arasında kurulmuş bir bağlantı varsa hata hemen yöneticiye bildirilir. Bu bildiriye uyarı denir. Bağlantı yoksa bu bilgi ajanda nesne olarak saklanır. Yönetici bağlantı kurduğunda nesne bilgisine bakarak alarm hakkında bilgi sahibi olur.

Alarmların Lokalize Edilmesi: Bir hata oluştuğunda bu hatanın nerede oluştuğunun bilinmesi gerekir. Problemin oluştuğu yerin ve sebebinin belirlenmesi için bir takım testler uygulanır.

Hata Düzeltme: Hata düzeltme işlemi hatalı ekipmanın tamiri veya değiştirilmesi, eğer yedeği varsa yedeğinin devreye sokulması şeklinde gerçekleştirilir. Hata düzeltme müşteri memnuniyetini sağlayabilmek açısından çok önemli bir görev üstlenir. Gerektiğinde müşteri ile acil temas kurmayı gerektirir.

Test: Cihazların davranışının belirlenmesi ve devrelerin kontrol edilmesi işlemi rutin olarak yapılmalıdır. Testler bizzat yönetici tarafından, ajan yazılımı tarafından periyodik olarak veya cihazların kendisi tarafından başlatılabilir. Elde edilen sonuçlar hemen veya belli süreler içerisinde yöneticiye bildirilir.

Hata Raporlama: Hata raporlama işi operatörün sisteme bilgi girişi yaptığı bir işlemdir. Bu giriş müşterinin şikayeti doğrultusunda veya yöneticiye ajan tarafından bildirilmiş olan hata doğrultusunda yapılır. Sistemde oluşan hatanın düzeltilip düzeltilmediğinin takip edilmesi açısından önemlidir.

4.2.3 Konfigürasyon Yönetimi

Şebeke elemanlarından bilgi toplama ve toplanan bilgilere göre bu elemanlara bilgi yüklenmesi işlemleri konfigürasyon yönetimince yapılır.

Şebeke Planlaması: Sistemin kapasitesinin gözlenmesi, mevcut kapasite gözlenerek gelişen teknolojilere göre gerektiğinde şebekeye yeni ekipmanlar yüklenmesi işleri şebeke planlaması yapılarak gerçekleştirilir. Muhtemel ihtiyaçların değerlendirilmesi, yeni haberleşme yollarının kurulması, şebeke sınırlarının belirlenmesi gibi görevleri yerine getirir.

Kurma İşlemi: Haberleşme şebekesinin çalışabilmesi için şebeke üzerindeki elemanların yazılım ve donanım anlamında işlevlerini yerine getirebilmesi gerekir. Kurma işlevi şebekenin genişletilmesi veya daraltılması gibi görevleri de sağlamak zorundadır. Şebeke üzerindeki elemanların ilk defa çalışması sırasında kullanacağı değerler de konfigürasyon yönetimince belirlenir.

Servis Planlama: Müşterinin talebine göre verilecek servisler planlanır ve kurulur. Verilecek servislerin sınırları da belirlenir.

4.2.4 Muhasebe Yönetimi

Muhasebe yönetimi sistemi işletmenin, servis sağlayıcıya ve sistemi kullanan müşteriye maliyetini hesaplar.

Kullanım Maliyetinin Hesaplanması: Müşteriyi ücretlendirebilmek için gereken bilgiler yönetilen sistem üzerinde depolanır. Ajan yazılımı bu maliyeti yöneticiye bildirir. Müşteri ve işletme açısından meydana gelebilecek aksaklıkları engelleyebilmek için bu bilgilerin güvenilir bir şekilde depolanması gerekir. Ücretin hesaplanabilmesi için yöneticinin tarife ve ücret bilgilerine sahip olması gerekir. Muhasebe yönetimi ayrıca elde edilen ücret bilgilerinin müşteriye fatura edilmesinden de sorumludur.

4.2.5 Güvenlik Yönetimi

Sistem içi güvenliğin sağlanması, müşteriyle sistem arasındaki bağlantı güvenliğinin sağlanması görevini güvenlik yönetimi üstlenir; sisteme izinsiz girişleri saptar ve engeller.

5. TMN PROTOKOLLERİ

Yönetici ile ajan arasında bilgi alışverişi yapılabilmesi için özel bir çerçeve formatı kullanılmalı, ilgili servisler bu verileri işleyerek gerekli cevapları verebilmelidir. Bu amaçla ITU-T'nin tavsiyelerinde belirttiği yönetim servisleri ve protokolleri, yine belirtilen kurallar doğrultusunda TMN uygulamalarında kullanılır.

TMN servisleri ve protokolleri OSI referans modelinin uygulama katmanında bulunur. Uygulama katmanında bulunan servisler bağlantı kurma ve çözme, alınan mesajların alınması ve işlenmesi görevini yerine getirir. Bu katmandaki servislerin toplamına **uygulama birimi (entity)**, servislere de **uygulama servis birimi** adı verilir.

5.1 Uygulama Servis Birimleri

Bu bölümde incelenecek servis birimleri şunlardır:

- Bağlantı Kontrolü Servis Birimi (Association Control Service Element (ACSE))
- Uzaktan İşlem Servis Birimi (Remote Operations Service Element (ROSE))
- Ortak Yönetim Bilgi Servis Birimi (Common Management Information Service Element (CMISE))
- Sistem Yönetim Uygulama Servis Birimi (Systems Management Application Service Element (SMASE))
- Dosya Transferi, Erişimi, Yönetimi (File Transfer Access Management (FTAM))

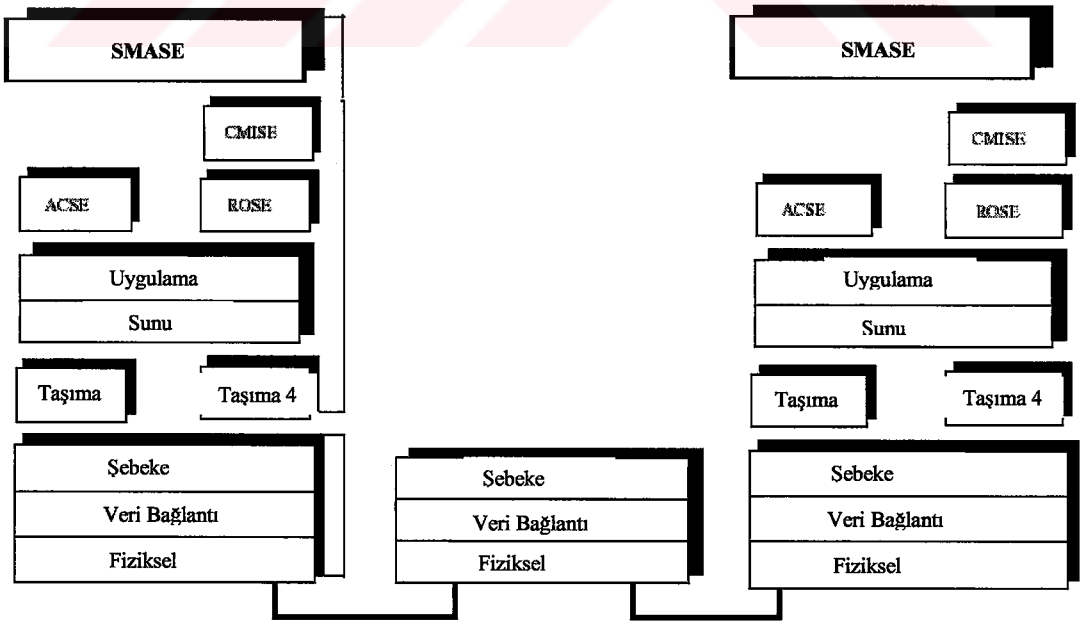
Bunlara ilaveten Güvenilir Transfer Servis Birimi (Reliable Transfer Service Element (RTSE)) de eklenebilir ancak TMN uygulamalarında kullanımı yaygın değildir.

5.1.1 Sistem Yönetim Servisi Elemanları

TMN servisleri arasındaki ve OSI katmanları arasındaki veri transfer işlemleri Protokol Veri Birimleri (Protocol Data Unit (PDU)) aracılığıyla yapılır. Protokol veri birimleri belirli bir servis ile ilişkilendirilirse o servisin adıyla anılır. Örneğin uygulama katmanı ile ilgili veri birimi uygulama protokolü veri birimi olarak (APDU) adlandırılır.

TMN servisleri bir sistem yönetim uygulama birimi tarafından kontrol edilir.

Bu uygulama birimi sistem yönetim uygulaması servis birimini kullanarak diğer servisler üzerinden karşı taraftaki sistem yönetim uygulama birimiyle bağlantı kurup veri alışverişinde bulunur. Bu veri alışverişi CMIP protokolüyle sağlanır. Bu amaçla SMASE servisi CMISE servisinden yararlanır. Kullanıcıdan gelen veriler CMISE servisi tarafından CMIP protokol çerçevelerine çevrilir. Bu çerçevelere CMIP APDU adı verilir. Bu APDU, başına ROSE servisinin başlığı eklenerek sunu katmanına aktarılır. Alt katmanlardan geçerken her katman kendi başlığını ekler ve karşı taraftaki katmanlar bu başlıkları çözerek yukarı katmanlara aktarırlar. Tabi ki karşı tarafla veri alışverişinin başlayabilmesi için öncelikle bağlantı kurulması gerekir. Bağlantı işlemi için SMASE servisi ACSE servisini kullanır.



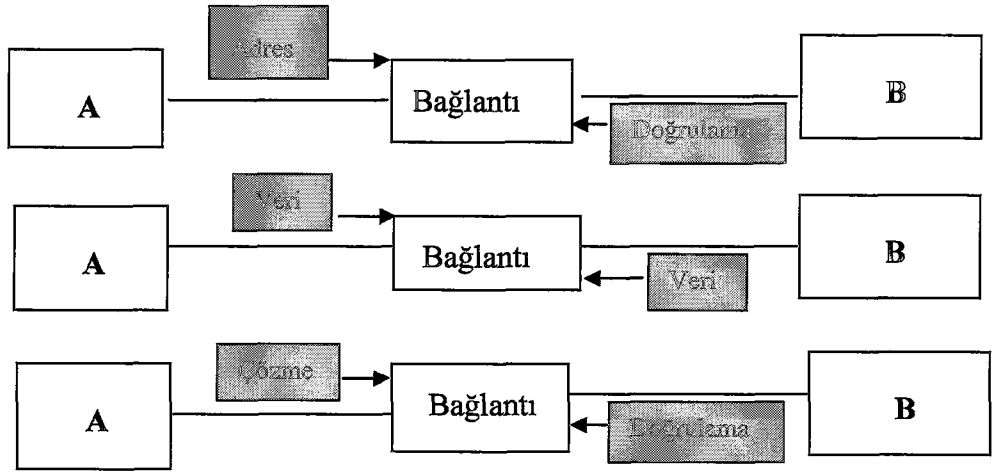
Şekil 5.1 TMN protokolleri

5.1.2 ACSE Bağlantı Servis Birimi

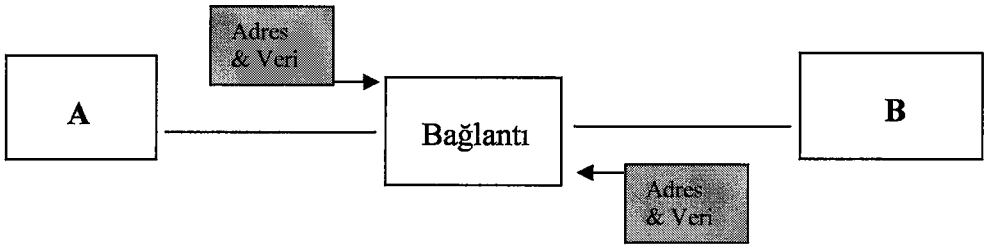
Yönetici ile ajan arasındaki ilişkinin kurulabilmesi için bağlantı kurulması gerekliliği vardır. Bağlantı kurulduktan sonra veri iletimi başlar. Yönetici ajanla devamlı bağlantı durumunda kalmak istemeyebilir. Bu durumda da bağlantıyı çözme gereksinimi vardır. Bütün bu gereksinimlerin karşılanabilmesi için ACSE servisine ihtiyaç vardır[10]. ACSE servisi bağlantılı ve bağlantısız olmak üzere iki türlü çalışır. Bağlantılı çalışmada yönetici veya bazı durumlarda ajan A-ASSOCIATE mesajını karşı tarafın adresini belirterek yollar ve cevap bekler. Bağlantısız durumda cevap beklenmez. Bu ihtiyaçların karşılanabilmesi için ACSE aşağıdaki tabloda gösterilen mesajları kullanır.

Tablo 5.1 ACSE Mesajları[17]

Bağlantı Modu	**Servis**	**Cevap Tipi**
Bağlantılı	A-ASSOCIATE	Doğrulamalı
Bağlantılı	A-RELEASE	Doğrulamalı
Bağlantılı	A-ABORT	Doğrulamasız
Bağlantılı	A-P-ABORT	Kullanıcı tabanlı
Bağlantısız	A-UNIT-DATA	Doğrulamasız



Şekil 5.2 Bağlantılı akış[20]



Şekil 5.3 Bağlantısız akış[20]

ACSE servisleri iki farklı modda çalışır. Bunlar Normal Mod ve X.410-1984 Modu'dur. Normal mod OSI ile TMN yönetiminde kullanılır. Atılan ACSE mesajı içerisinde sunu ve oturum katmanlarının parametreleri de taşınır. X410-1984 modunda sunu ve oturum katmanlarının parametreleri taşınmaz. Bu mod OSI kullanan TMN yönetiminde kullanılmaz.

Bağlantılı ACSE servisleri bazı fonksiyonların yerine getirilmesini sağlarlar. Bu fonksiyonlar, ana fonksiyonları, doğrulama (authentication) denilen şifre doğrulama fonksiyonunu ve de uygulama ismi doğrulama fonksiyonunu içerir.

ACSE mesajları Tablo 5.1'de gösterilmiştir.

Doğrulamanın amacı sisteme izinsiz girişleri engellemektir. Şifre doğrulama metodu doğrulama için kullanılan bir metoddur. Bunu sağlamak için A-ASSOCIATE ve A-ABORT mesajlarına parametre eklenmesi gerekmektedir. A-ASSOCIATE mesajının içerdiği parametreler, doğrulama mekanizması ismi, doğrulama değeri ve ACSE servisi için istek parametreleridir. A-ABORT mesajı tanı (diagnostic) adı verilen doğrulama hatasının sebebini gösteren bir parametre içerir.

Uygulama ismi doğrulama parametresi bir ACSE servis listedir. Bu liste yönetici ve ajan tarafından kullanılacak fonksiyon birimlerinin isimlerini içerir. A-ASSOCIATE mesajının içerisine uygulama isim listesi ve ACSE servisi istek parametresi eklenir. Eğer ajan bu listeyi doğrularsa cevabında liste parametresi bulunmaz. Doğrulamazsa kendi listesini yöneticiye gönderir.

Burada ACSE mesajlarını incelemek faydalı olacaktır.

5.1.2.1 A-ASSOCIATE

Bağlantı durumunda, bağlantı kuracak tarafların, hangi koşullar altında veri transferi yapacaklarına da karar vermesi gerekmektedir. Bu anlaşma A-ASSOCIATE aracılığıyla yapılır. Bağlantı kurmak isteyen taraf parametre listesini karşı tarafa gönderir. Bağlantı kurulacak olan taraf ya bu listeyi kabul eder ya da kendi listesini gönderir. Eğer bağlantı isteyen taraf bu listeyi kullanabiliyorsa bağlantı kurulur. Eğer listeyi kullanamazsa ABORT isteği atılır. A-ASSOCIATE mesajının parametreleri şunlardır:

Mod: Normal veya X.410-1984 modu seçilebilir.

Kullanıcı Bilgisi: Diğer tarafa bilgi taşımak için kullanılır. Şifre veya anahtar gibi güvenlikle ilgili bilgiler gönderilir.

Sonuç: Bağlantı isteğinin sonucu bildirilir. Üç adet değer alabilir. Bunlar kabul edildi, kesin olarak ret ve geçici ret cevaplarıdır. Geçici ret cevabı o an için ilgili tarafın bağlantı kurmaya hazır olmadığı bir durumda gerçekleşir.

Sonuç Kaynağı: Eğer bağlantı kabul edilmişse bu değer kabul eden servisin değerini taşır. Bağlantı kabul edilmemişse bu değer ilgili servis değerini alır.

Tanı: Ret cevabının sebebini bildirir.

Sunu Katmanı Adresi: Sunu servis erişim noktası adresidir.

Servis Kalitesi: Oturum katmanı tarafından verilen servislerin özelliklerini belirtir.

Oturum Katmanı İstekleri: Kullanılacak fonksiyonların kararlaştırılması için kullanılır.

Senkronizasyon Noktası Seri Numarası: Bir oturum kapatılıp tekrar açıldığında, (bağlantı çözülüp kurulması gibi), bu numara arttırılır.

Bayrak Atama Numarası: Senkronizasyon işlemi için gerekli ortamın sağlanması için oturumun özel bir durum alması gerekir. Bu özel durum token denilen bir istekle sağlanır.

Oturum Bağlantı Numarası: Bir bağlantıyı diğerinden ayırmak için kullanılan numaradır.

Normal modda aşağıdaki parametreler ek olarak gelir.

Uygulama Birimi İsmi: Bağlantı isteyen tarafın seçtiği, kullanılacak ACSE uygulamalarının bildirildiği isim listesidir. Bağlantının kurulabilmesi için karşı tarafın aynı isim listesini göndermesi gerekir.

Uygulama İşlem Başlığı: Uygulama işlem başlıklarının depolandığı kütük ismi veya nesne tanımlayıcı değerlerini alabilir.

Uygulama Çağırma Mesaj Numarası: Bir uygulama işlemi birini diğerinden ayırt edecek bir numarayla ayrılır. Bu numara uygulama çağırma mesaj numarasıdır.

Uygulama Birim Numarası: Uygulama biriminin ASN.1 formatındaki numarasıdır.

Sunu Birimi Tanım Listesi: Bu liste kullanılacak ASN.1 tanımlarını ve sunu içerik tanımlayıcısını bildirir.

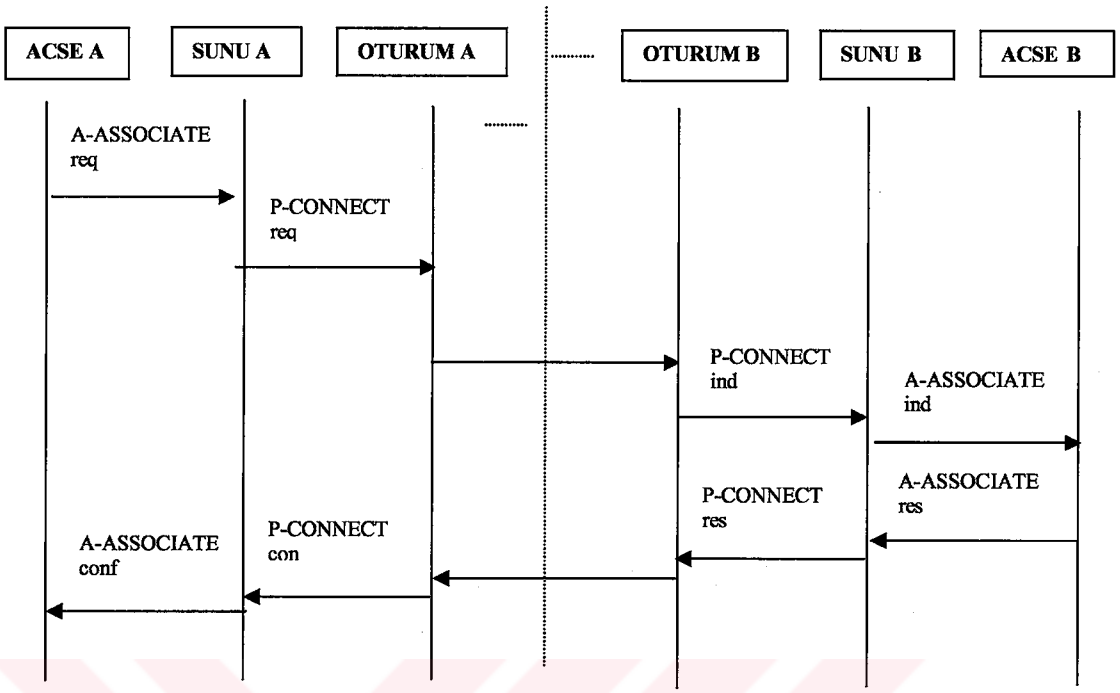
Sunu Birimi Tanım Listesi Sonucu: Bu liste ajanın yöneticiye sunduğu sunu birimi tanım listesidir.

Başlangıç Sunu Birimi İsmi: Sunu birimi listesinin başlangıç değeridir.

Başlangıç Sunu Birimi İsmi Sonucu: Ajanın sunu birimi listesinin başlangıç değeridir.

Sunu Katmanı Özellikleri: Sunu katmanını kullanan uygulamanın ki burada TMN katmanlarıdır, bu katmanda kullanacağı özelliklerdir.

ACSE Özellikleri: Bu parametre doğrulama veya uygulama ismi doğrulama özelliğinin kullanılıp kullanılmadığını belirler. 0 değeri doğrulama kullanılacağını, 1 ise isim doğrulamanın kullanılacağını belirler. Mesajda bu parametre yoksa ana fonksiyonlar haricindeki özellikler kullanılmaz[10].



Şekil 5.4 A-ASSOCIATE akış diyagramı[20]

5.1.2.2 A-RELEASE

Bağlantıyı sona erdirebilmek için bir çözme mesajına ihtiyaç duyulur. Bu mesaj A-RELEASE mesajıdır. Buradaki çözme isteği tamamen kontrollü bir çözme isteğidir. Yani herhangi bir veri kaybından dolayı oluşmaz. Sistem operatörü bağlantıyı koparmak istediğinde bu mesajı gönderir. Bu istek sonucunda ajandan yanıt gelmesi gerekir. A-RELEASE mesajının parametreleri aşağıdadır:

Neden: Bu parametre bağlantının nasıl sona erdirileceğini belirler. Alabileceği değerler normal, acil ve kullanıcı-tanımlı değerleridir. Diğer taraftan bu mesajı alan taraf parametre değeri olarak normal, tamamlanmadı veya kullanıcı-tanımlı bir değer gönderebilir. Bunun anlamı, çözme isteğinin kabul edildiği, ajanda gerçekleşen veri işlemlerinin devam ettiği ve bağlantının çözülemeyeceği veya kullanıcı tarafından belirlenen bir değerin yöneticiye gönderilmesi şeklindedir.

Kullanıcı Bilgisi: Bu parametre kullanıcın isteğine bağlı olarak mesajda göndereceği bilgileri içerir. Kullanıcı kelimesinin anlamı ajan yazılımını kullanan operatördür. Ajan yazılımını üreten firma tarafından müşteri (operatör) isteğine bağlı olarak

sistemle ilgili çözme isteği sırasında oluşan durumu operatöre bildirmek amacıyla kullanılabilir.

Sonuç: Bu parametre, mesajı alan taraf tarafından doldurulur. Çözme isteğinin kabul edilip edilmediğini belirtir.

5.1.2.3 A-ABORT

Bu mesaj ajan veya yönetici tarafından gönderilebilir. İki taraf arasında haberleşme yapılamadığı zaman atılabilir. Haberleşme sırasında veri kaybı oluşması, ajan veya yönetici yazılımlarında telafi edilemeyen bir hatanın oluşması ki bu ürünün son halinde oluşmaması gereken bir hatadır, bu mesajın atılmasına yol açabilir. Cevap mesajı beklenmez. A-ABORT parametreleri aşağıda verilmiştir.

İptal Kaynağı: İptale yol açan tarafın kim olduğunu belirtir. Belirtilen kaynak ACSE gibi bir servis birimini işaret eder ki genellikle böyledir.

Kullanıcı Bilgisi: Bu mesajı alan tarafa açıklayıcı bir bilgi verebilmek amacıyla kullanılır.

5.1.2.4 A-P-ABORT

Bu mesaj Uygulama Servis birimlerinin altındaki uygulama katmanı veya daha alttaki bir katmandan kaynaklanan hata sonucu ilgili tarafın kendi içerisinde atılan bir mesajdır. Yani apartmanın alt katlarından üst katlara gönderilen bir mesajdır. Karşı tarafa gönderilmez.

5.1.2.5 A-UNIT-DATA

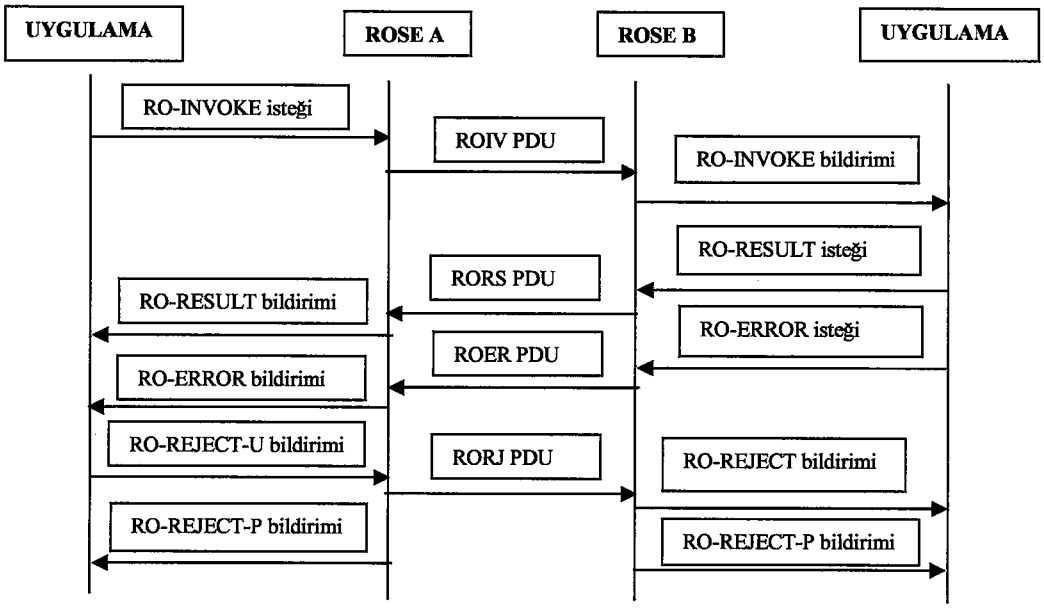
Şu ana kadar anlatılan mesajlar bağlantılı mesajlardı. A-UNIT-DATA bağlantısız durumda atılan bir mesajdır. Bağlantılı durumda uygulama ve oturum katmanlarındaki parametreleri bir defa tanımlamak yeterlidir. Bağlantısız durumdaysa her defasında bu parametreleri bildirmek gerekir. Bu durumda sadece doğrulama fonksiyonu kullanılabilir. Bu fonksiyona ait sadece iki parametre kullanılır ki bunlar doğrulama mekanizma ismi ve doğrulama değeridir. ACSE servisi için kullanılan diğer parametreler ise aynen geçerlidir.

5.1.3 ROSE Uzaktan İşlem Servis Birimi

ROSE'un açılımı İngilizce Remote Operations Service Entity (Uzak Operasyonlar Servis Birimi)'dir. Bir A uygulama birimi (yönetici veya ajandaki servislerin toplamı) diğer taraftaki B uygulama birimine iş yaptırmak istediğinde ROSE servisini kullanır. Alacağı cevap yine B tarafının ROSE servisi tarafından verilir. A tarafına işlemi başlatan anlamında isteyen (invoker), B tarafına ise uygulayan anlamında uygulayıcı (performer) adı verilir. Aşağıdaki tabloda ROSE mesajlarını ve akış diyagramı görülmektedir[11].

Tablo 5.2 ROSE Mesajları[17]

Servis	Cevap Tipi
RO-INVOKE	Doğrulamasız
RO-RESULT	Doğrulamasız
RO-ERROR	Doğrulamasız
RO-REJECT-U	Doğrulamasız
RO-REJECT-P	Kullanıcı Tabanlı



Şekil 5.5 ROSE mesajları akış diyagramı[20]

5.1.3.1 RO-INVOKE

RO-INVOKE mesajı yönetici tarafındaki uygulama biriminin ajan tarafındaki uygulama birimine iş yaptırmak amacıyla kullandığı bir mesajdır. Bu işlem kaynak üzerinde yapılmak istenen herhangi bir işlem olabilir. Bu mesajın cevabı RO-RESULT mesajıyla gönderilir.

Bu mesajın parametreleri şu şekildedir.

Çağırma Numarası: Atılan mesajı ayırt etmek amacıyla kullanılan bir numaradır.

Bağlantı Numarası: Eğer bir RO-INVOKE mesajı, öncesinde atılan bir RO-INVOKE mesajı ile bağlantılıysa bu numara kullanılarak daha önce önceki mesajın çağrı numarası belirtilir.

İşlem Değeri: Yönetici ile ajan arasında kararlaştırılmış olan işlem numarasıdır.

Argüman: İşlem değeri verilmiş olan işlemin çeşidini belirten bir değer taşır.

İşlem Sınıfı: İşlemler asenkron veya senkron olarak çalışırlar. Senkron çalışmada atılan mesajın cevabı gelmeden başka bir işleme geçilmez. Asenkron çalışmadaysa mesaj atıldıktan sonra cevabın gelmesi şart değildir. Uygulama birimi diğer işlemlere geçebilir.

Beş çeşit işlem sınıfı bulunur:

Sınıf 1: Atılan mesaja sonuç veya hata cevabının alındığı senkron çalışma sınıfıdır.

Sınıf 2: Atılan mesaja herhangi bir anda sonuç veya hata cevabı atıldığı asenkron çalışma sınıfıdır.

Sınıf 3: Atılan mesaja sadece hata cevabının alındığı asenkron çalışma sınıfıdır.

Sınıf 4: Atılan mesaja sadece sonuç cevabının alındığı asenkron çalışma sınıfıdır.

Sınıf 5: Atılan mesaja cevap alınmadığı asenkron çalışma sınıfıdır.

Öncelik: Mesajın önceliğini belirleyen parametredir. Bu parametre yönetici ve ajan arasında haberleşmeyi sağlayan sistemin kapasitesine ve kullanılan yazılımın yeterli

olmasına bağlıdır. Kaç mesajın aynı anda işlenebileceği ve hangisinin öncelikli olarak atılabileceği önemlidir. Küçük numaralar daha önceliklidir.

5.1.3.2 RO-RESULT

RO-INVOKE mesajı ile yerine getirilen başarılı bir operasyonun sonucu olarak RO-RESULT mesajı atılır. Bu mesajın parametreleri çağrı numarası, sonuç, işlem değeri ve önceliktir. Çağrı numarası dışındaki parametreler zorunlu değildir. Buradaki çağrı numarasının değeri gelen RO-INVOKE mesajının çağrı numarasıyla aynı değerdedir. Ancak sonuç parametresi mevcutsa işlem değeri parametresi de mevcuttur.

5.1.3.3 RO-ERROR

Bu bir hata mesajıdır. RO-INVOKE mesajını alan taraf istenilen işlemi gerçekleştirmeye çalışırken bir hata oluşursa, bu durumu bir hata mesajıyla atan tarafa bildirir. RO-ERROR mesajının parametreleri çağrı numarası, hata değeri, hata bilgisi ve önceliktir. Çağrı numarası ve hata değeri zorunlu parametrelerdir. Buradaki çağrı numarasının değeri de gelen RO-INVOKE mesajının çağrı numarasıyla aynı değerdedir. Hata ile ilgili bilgiler hata bilgisi parametresine koyulur.

5.1.3.4 RO-REJECT

RO-INVOKE mesajının işlenmesiyle ilgili bir problem oluşursa bu durum RO-REJECT mesajıyla yöneticiye bildirilir. İki tür RO-REJECT mesajı vardır.

RO-REJECT-U mesajı çağrı mesajını alan taraf tarafından atılan bir mesajdır. Buradaki U (user) ajan yazılımını kullanmak isteyen kullanıcı anlamındadır. Yani ilgili çağrı mesajını atan tarafa doğru ret mesajının atıldığını bildirir. Bu mesajın atılma sebebine örnek vermek gerekirse, ajan programında karşılaşılabilen olumsuz durumları söyleyebiliriz. Bunlardan bir tanesi kullanılan ekipmana bağlı olarak oluşan hafıza yetersizliği olabilir. Bir ajan yazılımı ortalama 3500-4000 arası nesneyi yaratmak ve kontrol etmek durumunda kalabilir. Ortalama bir nesnenin boyu 350 byte civarındadır. Bu durumda bir ajan yazılımı nesneler için en az 1.5 Mbyte yer ayırmak zorundadır. Yaratılacak nesne sayısı yazılım konfigürasyonuyla kontrol altına alınabilir. Bu durumda maksimum nesne sayısı aşıldığı zaman RO-REJECT-U mesajı verilecektir. İkinci ve daha çok karşılaşılan durum, elemanları nesneler olan

ağaç yapısı hiyerarşisine uygun olmayan bir nesne yaratma isteğine karşılık atılan RO-REJECT-U mesajıdır.

RO-REJECT-U mesajının parametreleri çağrı numarası, problem ve önceliktir. Zorunlu olan parametreler çağrı numarası ve problem parametreleridir. Problem parametresinin alt parametreleri çağrı problemi, sonuç döndürme problemi ve hata döndürme problemidir.

RO-REJECT-P mesajı RO-INVOKE mesajını atmaya çalışan uygulama biriminin alt yapısında oluşan problemlerden dolayı, o uygulama biriminin ROSE servisi tarafından atılan bir hata mesajıdır. Bölüm 4'te bahsedildiği gibi katmanlar arasındaki mesajlar PDU'lar tarafında iletilir. Eğer PDU yapısında bir bozukluk oluşursa veya uygulama ve daha alt katmanlarda PDU ile ilgili bir sorun oluşursa, uygulama birimine doğru RO-REJECT-P mesajı atılır. Bu mesaj ajan veya yönetici tarafında oluşabilir.

RO-REJECT-P mesajının tek parametresi çağrı numarasıdır. Çağrı numarasının değeri atılmaya çalışılan RO-INVOKE mesajı ile aynı değerdedir.

5.1.4 CMISE Ortak Yönetim Bilgi Servis Birimi

CMISE (Common Management Information Service Element) Ortak Yönetim Bilgi Servis Birimi anlamındadır. Yönetici ile ajan arasındaki yönetim operasyonları ve uyarılarından sorumludur. Bu işlemler CMIP protokolüyle taşınır ve CMISE ile tanımlanır. CMISE tarafından sağlanan servisler ROSE aracılığıyla CMIP protokolüyle taşınır. Bu mesajlar Tablo 5.3'te verilmiştir[12].

Tabloda görülen mesajlardan, cevap tipi doğrulamalı ve doğrulamasız olarak görülen mesajlar, her iki türlü de gerçekleştirilebilecek işlemleri gösterir. Atılan mesajın türüne bağlı olarak, yöneticiye doğrulama mesajı gönderilmeyebilir. Mesajların başına konulan M karakteri yönetim (management) anlamında kullanılır. Burada bahsedilen mesajlar nesneler vasıtasıyla kaynakların yönetimi ile ilgili mesajlardır.

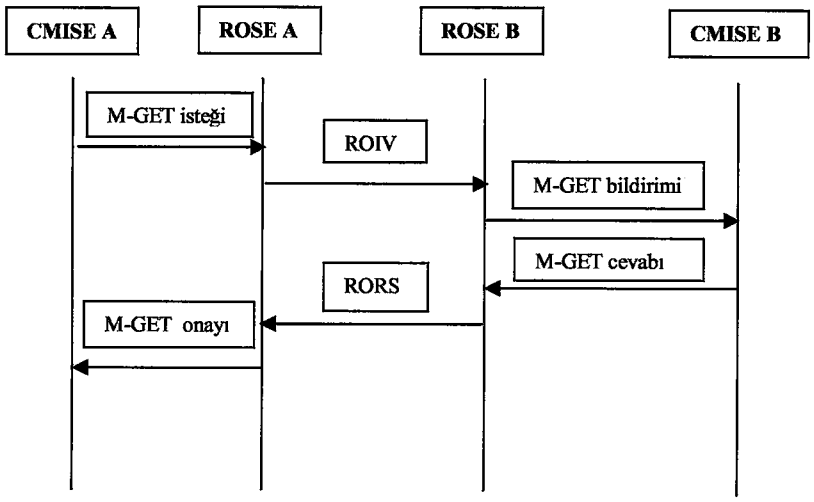
Tablo 5.3 CMISE Mesajları[17]

Servis	Cevap Tipi
M-GET	Doğrulamalı
M-SET	Doğrulamalı ve Doğrulamasız
M-CREATE	Doğrulamalı
M-DELETE	Doğrulamalı
M-ACTION	Doğrulamalı ve Doğrulamasız
M-CANCEL-GET	Doğrulamalı
M-EVENT-REPORT	Doğrulamalı ve Doğrulamasız

5.1.4.1 M-GET

Yöneticinin ajana gönderdiği bir mesaj türüdür. Mesajda belirtilen nesne ile ilgili bilgilerin yöneticiye bildirilmesi sağlanır. Bu mesaj tek bir nesne için kullanılabileceği gibi bir nesne grubunu veya nesne ağacındaki bir düğümün altındaki nesneleri getirebilir. M-CANCEL-GET mesajıyla bu istek yarıda kesilebilir. Eğer bir düğüm altındaki nesne bilgileri istenmişse normalde alınan M-GET cevabı yerine son alınan nesneye kadar LINKED-REPLY mesajı gönderilir. Bu aslında bir mesaj türü değildir. Birden fazla nesne için atılan mesajlarda, link-id kullanıldığı için mesajın çeşidinin LINKED-REPLY olarak belirtilmesine rağmen, aslında M-GET mesajının bir türevidir.

Aşağıdaki şekilde tek bir nesne isteği için gönderilen M-GET mesajının akış diyagramı görülmektedir.



Şekil 5.6 M-GET mesajı akış diyagramı[20]

Birden fazla nesnenin cevap olarak alındığı durumlarda her bir LINKED-REPLY mesajı için cevap olarak ROIV PDU ile M-GET onayı mesajları alınır.

5.1.4.2 M-CANCEL-GET

5.1.4.1'de bahsedildiği gibi M-CANCEL-GET mesajı M-GET mesajını iptal etmek için kullanılır. M-GET mesajının cevabı olarak beklenmedik derecede fazla ve gereksiz LINKED-REPLY mesajı geldiği zaman, M-CANCEL-GET mesaj kullanılarak bilgi akışı kesilebilir.

5.1.4.3 M-SET

Bir nesne tarafından temsil edilen kaynakta yapılmak istenilen bir değişiklik M-SET mesajıyla sağlanır. Link numarası, filtre, içerik, senkronizasyon ve erişim kontrolü gibi parametreleri vardır. M-GET gibi çalışır, ancak nesne özelliklerinin değerini değiştirir. Doğrulamalı veya doğrulamasız çalışabilir.

5.1.4.4 M-CREATE

Bu mesaj nesne yaratmak amacıyla kullanılır. Yaratılan nesnenin amacı bir kaynağı temsil etmek veya o kaynağa bir işlem yaptırabilmek olabilir. Örnek olarak yeni bir abone yaratılması gösterilebilir. Bazı nesneler santral açılışında standart olarak bulunması gereken nesnelerdir ve bunlar ajan yazılımı tarafından yaratılırlar. Raf nesnesini buna örnek olarak verebiliriz. Bazı nesneler de yönetici tarafından yaratılmalıdır. Buna örnek olarak da abone nesnesini verebiliriz.

5.1.4.5 M-ACTION

Bazı yönetim uygulamaları bir işin başlatılıp hemen veya bir süre sonra sonuçlandırılmasını isteyebilir. Bu amaçla M-ACTION mesajı kullanılır. ATM santrallerinde bulunan ajan yazılımlarında VCI ve VPI değerlerini gösteren nesneler bulunur. Bu nesneler üzerinden linklerin güvenilirliğini test edebilmek amacıyla M-ACTION mesajı gönderilir. Mesajın cevabı hemen alınmayabilir, çünkü yapılacak işlemler uzun sürebilir. Bu sırada yönetici başka işlemler yapmak isteyebilir. Test yapıldıktan sonra elde edilen verileri içeren M-ACTION mesajı yöneticiye atılır.

5.1.4.6 M-DELETE

M-CREATE mesajının tersi yani bir nesne silmek amacıyla gönderilecek bir mesajdır. Yönetici artık kullanmak istemediği bir kaynağı sistemden silmek amacıyla M-DELETE isteği gönderebilir. Örnek olarak aboneliği bitmiş olan bir müşterinin aboneliğini temsil eden nesne silinebilir. Ancak bazı nesnelerin M-DELETE mesajı gönderilmesine rağmen silinmemesi istenir. Örneğin raf nesnesinin M-DELETE mesajıyla veya ajan yazılımı tarafından silinmemesi gerekir.

5.1.4.7 M-EVENT-REPORT

Bir nesnenin özelliklerini gösteren değişkenlerde bir değişiklik olduğunda veya o nesneyle ilgili bir hata oluştuğunda M-EVENT-REPORT mesajıyla yöneticiye raporlanır. Mesaj içerisindeki parametrelerde, gönderilen bilginin ne zaman oluştuğu, hangi nesneyle ilgili oluştuğu gibi bilgiler bulunur.

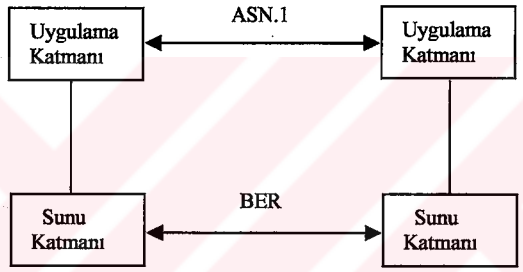
5.1.5 CMIP Yönetim Bilgi Protokolü

CMISE servisi ile ROSE servisi arasında CMIP (Common Management Information Protocol) protokolü işlev görür. CMISE servisi mesajları CMIP protokol veri birimleriyle (PDU) taşınır. CMIP PDU'lar ise ROSE servis veri birimleri içerisine dahil edilerek karşı tarafa aktarılır[13]. Bunu şu şekilde açıklayabiliriz:

CMISE'ın M-GET isteği CMIP'ın m-get PDU'suna aktarılır. Bu m-get PDU'su ROSE servisinin RO-INVOKE PDU'su ile taşınır. Bunun tersi de geçerlidir.

5.2 ASN.1 Ve GDMO Dillerinin Nesne Tanımlanmasında Kullanımı

Sistemlerin birbirleriyle anlaşabilmeleri için aralarında taşınan verinin iki taraf tarafından da anlaşılır olma gerekliliği vardır. Bunu sağlamak için iki tarafta da verilerin tanımlanmasında kullanılan dilin söz diziminin ve anlatımının aynı olması gerekir. TMN uygulamalarında uygulama katmanı seviyesinde soyut söz dizimi (abstract syntax) diye adlandırılan bir tanımlama kullanılır. Bu söz dizimi verilerin nasıl düzenlendiğini ve ne anlam içerdiğini belirler. ASN.1 (Abstract Syntax Notation 1) bu anlamda kullanılan bir dildir[9]. Uygulama katmanında ASN.1 olarak tanımlanan veri sunu katmanında BER (Basic Encoding Rules) denilen bir kodlama tekniği ile var olmalıdır. BER de bir çeşit söz dizimidir. Bu söz dizimine transfer söz dizimi denir.



Şekil 5.7 Katmanlar arası söz dizimi ilişkisi[18]

2. bölümde kısaca açıklanan nesne kavramı şebekedeki kaynakları ifade eden soyut bir kavram olarak nitelendirilmişti. Bu nesneler uygulama katmanı seviyesinde GDMO tanımlama dili ile tanımlanır. Her nesnenin ait olduğu bir sınıf vardır. Bu sınıfları tanımlamak için özellik, davranış, operasyon ve uyarı olarak adlandırılan değişkenler ve açıklamalar kullanılır. GDMO tanımlamaları içerisinde kullanılan değişkenler de ASN.1 söz dizimi ile ifade edilirler.

5.2.1 ASN.1 Veri Tipleri

ASN.1 veri tipleri diğer programlama dilleriyle benzerlik gösterir. Karakter, tam sayı, yapı, bileşim gibi veri tiplerine karşılık düşen veri tiplerine sahip olduğu gibi kendine has veri tiplerine de sahiptir.

ASN.1 veri tiplerini dört sınıfa ayırabiliriz. Bunlar Basit Veri Tipi, Yapısal Veri Tipi, Etiket Veri Tipi ve Alt Seviye Veri Tipi'dir. Veri tipleri bu sınıflar altında toplanırlar.

Basit veri tipleri CharacterString, Boolean, Integer, Real, Bitstring, OctetString, Enumerated tipleridir. ASN.1'de değişkenlerin tip gösterimine örnek olarak şu atamayı gösterebiliriz:

ÖrnekDeğişken ::= INTEGER

Basit veri tiplerinde değinilmesi gereken konu CharacterString tipidir. ASN.1'de bu tiple aynı anlamı taşıyan ancak farklı yerlerde kullanılan tipler vardır. Bunlara örnek olarak NumericString, GraphicString, PrintableString gibi tipleri gösterebiliriz.

Yapısal veri tipleri Sequence, Sequence-Of, Set, Set-Of, Choice, Any veri tipleridir. Buradaki Sequence yapı, Choice bileşim anlamındadır. Any ise VOID anlamında düşünülebilir. Buradaki Of ekleri programlama dillerindeki katar anlamını taşır. Yapısal veri tiplerine örnek olarak aşağıdaki yapı verilebilir:

```
ÖrnekYapı ::= SEQUENCE
{       ÖrnekDeğişken1      INTEGER,
        ÖrnekDeğişken2      INTEGER}
```

Etiket veri tipi yukarda anlatılan veri tiplerinin başına getirilerek kullanılan bir veri tipidir. Etiket veri tipi, yapısal veri tipleri kullanıldığında ortaya çıkan bir sorunu çözebilmek amacıyla kullanılır. Nesne ismini ifade edebilmek için yapısal veri tipi yeterli olmaz. Çünkü nesne ismi aynı zamanda bir nesne ayırıcı numaraya karşılık gelir. Hangi nesne olduğunun deşifre edilebilmesi için veri bloğuna bir etiket vermek gerekir. Etiket tipi açık veya kapalı olabilir. Kapalı etikette veri tipinin karşı tarafa bildirilmesi gerekmez. Açık etiket tipindeyse gerekir. Örnek olarak CharacterString tipinde bir verinin NumericString ya da GraphicString mi olduğunun bilinmesi gerekir.

Kullanıcının tanımlayabildiği dört çeşit etiket tipi vardır. Bunlar **Evrensel** etiket, **Uygulama** etiketi, **İçeriğe Özel** etiket ve **Özel** etiketlerdir.

Evrensel etiket X.208 tavsiyesinde belirtilen yani standart olan tüm veri tipleri için kullanılır. Uygulama etiketi sunu katman biriminin anlayabildiği değişkenler için kullanılır. İçeriğe özel etiketler yapı veri tiplerinin elemanlarının

hangi sırada olduğunu belirtmek için kullanılır. Özel etiketler ise bir firmaya veya bir ülkeye özel veri tiplerini tanımlamak için kullanılır.

Bir nesneyi tanımlamak için OBJECT IDENTIFIER diye adlandırılan bir etiket kullanılır.

ÖrnekNesne OBJECT IDENTIFIER ::= {1 3 5 3 6 137 165}

Zamanla ilgili değişkenleri tanımlamak için de etiket kullanılır.

Alt veri tipleri değişkenlere verilecek değerlerini sınırlarını belirlemek için kullanılır.

ÖrnekDeğişken ::= INTEGER (100..200)

Alt veri tipi diye adlandırılan bir veri tipi de mevcuttur. Bu veri tipinin iki tane kullanım amacı vardır. Birincisi iki veya daha fazla veri tipi benzer özelliklere sahipse ayrı olan özellikler alt tip olarak sınıflandırılır. İkincisi bir değişkenin alabileceği değeri sınırlandırmak için kullanılır. Örnek olarak

ÖrnekDeğişken ::= INTEGER (500..1000) veya

ÖrnekDeğişken ::= INTEGER (500 | 750 | 1000)

ASN.1'de veri tipleri birbirleriyle de tanımlanabilirler.

ÖrnekDeğişken2 ::= ÖrnekDeğişken

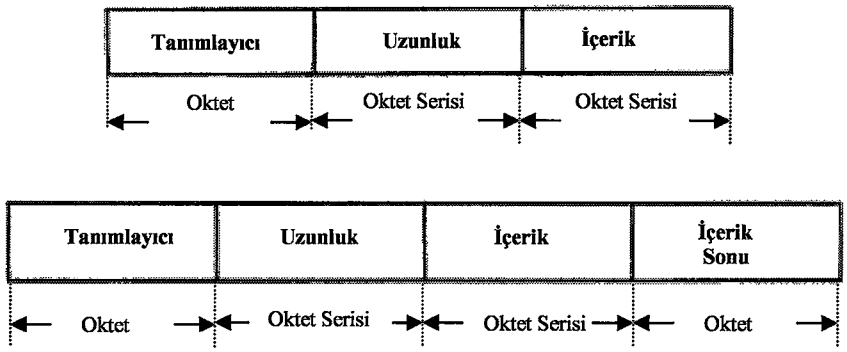
ASN.1 ile daha ayrıntılı bilgi için X.208 ve X.680 tavsiyelerine başvurulabilir.

5.2.2 BER (Basic Encoding Rules)

Uygulama katmanında ASN.1 tipinde girilen değerler sunu katmanında BER tipinde şifrelenir. BER temel şifreleme kuralları anlamındadır. Burada kullanılan şifreleme kelimesinin anlamı verilerin bir oktet dizisi şeklinde sıralanması anlamındadır. Kodlama anlamına da gelir.

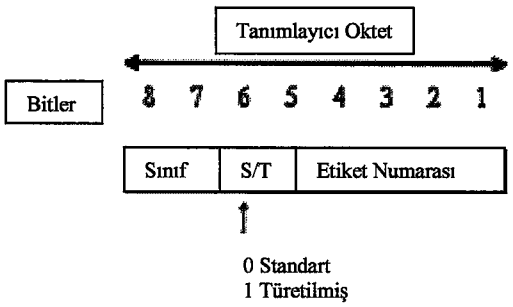
BER ile yapılan bir kodlamaya baktığımız zaman oktet dizisindeki verilerin üç çeşide ayrıldığını görürüz. Bunlar tanımlayıcı, uzunluk ve içeriktir. Tanımlayıcı,

ASN.1 veri tipini; uzunluk, içerik kısmının uzunluğunu belirtir. İçerik kısmı da iletilecek ASN.1 değerlerini içerir. BER formatı Şekil 5.8'de gösterilmiştir.



Şekil 5.8 BER oktet formatları[17]

BER formatı iki çeşittir. İkinci formatın ilkinden farkı içerik oktetlerinden sonra içerik sonu oktetinin bulunmasıdır. İkinci format veri uzunluğunun bilinmediği durumda kullanılır. Tanımlayıcı oktetinin ilk iki biti verinin sınıfını yani etiketini gösterir. Sonraki bit , gönderilen verinin temel bir veri tipi mi, yoksa türetilmiş bir veri tipi mi olduğunu belirtir. Temel veri tipinde içerik oktetleri doğrudan verilerin gerçek değerleri gösterilir. Türetilmiş veri tipinde ise bazı değerler sıkıştırılmış şekilde gösterilir. Bu sayede karşı tarafa gönderilen verilerin boyu kısaltılmış olur. Kalan beş bit ise etiket numarasını gösterir. Ancak etiket numarasının değeri 30'dan daha büyük bir değerse bu beş bitin her biri bir olarak verilir ve etiket numarası gerekli sayıda oktet eklenerek verilir. Şekil 5.9'da tanımlayıcı oktetin formatı gösterilmiştir.



Şekil 5.9 Tanımlayıcı oktet[17]

Uzunluk alanı belirli ve belirsiz olmak üzere ikiye ayrılır. Belirli alan kısa veya uzun olarak ikiye ayrılır.Uzunluk alanının en anlamlı biti 1 ise uzunluk

127'den, uzun 0 ise 128'den küçük anlamındadır. Kısa durumda bir oktet, uzun durumda ise iki veya daha fazla oktet kullanılır.

Belirsiz durum veri uzunluğunun bilinmediği durumdur. Bu durum için ikinci tip BER formatı geçerlidir. Belirsiz veri uzunluğu uzunluk alanının ilk bitinin 1 ve diğer bitlerin 0 olmasıyla anlaşılır. Veri bloğunun bitişi is 0 oktetiyle anlaşılır.

İçerik oktetlerinin temel ve türetilmiş olarak kodlanabildiği belirtilmişti. Temel sınıfta içerik oktetleri kodlanmamış bir şekilde yazılır. Türetilmişte ise bazı oktetler sıkıştırılmış olarak yazılırlar. Örnek olarak nesne tanımlayıcısı tipinde bir veri tipini ele alalım. Nesne tanımlayıcısının değeri { 1 60 8 1 4 5 130 100} olsun. Tanımlayıcının ilk değeri olan 1 sayısı 40 sayısı ile çarpılır ve bir sonraki değer olan 60 ile toplanır. Elde edilen sayı 100'dür. Bu durumda nesne tanımlayıcısının kodlanmış değeri {100 8 1 4 5 130 100} olur. Böylece bir oktet tasarruf edilmiş olur.

BER dışında kullanılan CER (Canonical Enoding Rules) , DER (Distinguished Encoding Rules) ve PER (Packed Encoding Rules) kodlama teknikleri de mevcuttur. Bu teknikler BER kurallarına bazı kısıtlama getiren ve kullanım kolaylıkları sağlayan tekniklerdir.

5.2.3 GDMO (Guidelines for The Definition for Managed Objects)

Kaynakları temsil etmek için ajan yazılımıyla nesnelerin oluşturulduğu belirtilmişti. Her bir nesnenin, sahip olduğu özellikleri gösteren, özelliklerindeki bir değişim veya olağan dışı bir durumdan dolayı yöneticiye bildireceği uyarıları tanımlayan, nesnenin yapabildiği işleri anlatan davranışları bildiren ve yapabileceği işleri tasvir eden elemanları vardır. Bu elemanlar **Yönetilen Nesne Tanımlanması** olarak adlandırılan bir çeşit programlama dili ile tanımlanırlar.

Bir şebeke ile ilgili bütün nesnelerin toplamına **Yönetim Modeli** adı verilir. Yönetim modelini bir ağaç yapısı olarak düşünebiliriz. Nesneler arasında hiyerarşik bir yapı mevcuttur. Nesne ağacının en üstünde kök bulunur. Bu bir nesne değil bir referans noktasıdır. Kökün altında adı **managedElement (yönetilen birim)** olan bir nesne kullanılır. Bu nesne yönetilen bir cihazı temsil eder. Örneğin yönetici bir ATM santralına bağlanıp o santralle ilgili bilgileri ajan yazılımından istediğinde ilk alacağı nesne bu managedElement nesnesi olacaktır. Diğer tüm nesneler bu nesnenin altında

bulunurlar. Ancak istisnai bir durum da söz konusudur. Eğer yönetici başka bir şebeke üzerinden ajan yazılımına bağlanırsa kökten sonra şebeke nesnesi bulunur. Ancak bu nesne yönetilen cihazın herhangi bir birimini değil şebekenin kendisini temsil eder.

ITU-T'nin X.720 ve X.721 standardında yönetim bilgi modeli, yönetilen bir nesnenin nasıl tanımlanacağı hakkında bilgi vermekte ve bazı standart nesneleri de tanımlamaktadır. X.722 tavsiyeleri GDMO'yu tanımlar.

GDMO'nun temel kavramları aşağıda verilen örnek nesne tanımlamasına göre açıklanabilir. Buradaki amaç GDMO dilinin tüm özelliklerini anlatmaktan ziyade bir nesnenin nasıl tanımlandığını açıklayabilmektir.

circuitEndPointSubgroup MANAGED OBJECT CLASS
DERIVED FROM "Recommendation X.721 : 1992":top;
CHARACTERIZED BY
circuitEndPointSubgroupPackage PACKAGE
BEHAVIOUR
circuitSubgroupBehaviour BEHAVIOUR
DEFINED AS
"A set of circuit end points that directly interconnects one exchange with another, having common values for
the attributes listed in this package. Note that the term exchange includes PBX where applicable."
-- Annex A/E.410 defines circuit sub group--
;;
ATTRIBUTES
circuitEndPointSubgroupId GET,
numberOfCircuits GET,
labelOfFarEndExchange GET,
signallingCapabilities GET,
informationTransferCapabilities GET,
circuitDirectionality GET,
transmissionCharacteristics GET,
userLabel GET-REPLACE;
NOTIFICATIONS
"Recommendation X.721:1992": attributeValueChange,
"Recommendation X.721:1992": objectCreation,
"Recommendation X.721:1992": objectDeletion;
;;
REGISTERED AS {m3100ObjectClass 31}; [4]

Bu nesne M.3100 standardında tanımlı olup örneğin iki santralin birbirine bağlı olduğu devreyi (link) gösterir. Bu tanımın bir nesne olduğunu ilk satırdaki MANAGED OBJECT CLASS anahtar kelimesinden öğreniyoruz. İkinci satır bu nesnenin **top** referans noktasından türetildiğini gösterir. Bir nesne başka bir nesneden de türetilebilir. Sonraki satırın anlamı bu nesnenin özelliklerinin paket halinde birleştirildiğidir. Paket, nesne içerisinde referans gösterilerek paket dışında da tanımlanabilir. Çok fazla özelliği olan nesneler için bu yöntem tercih edilmektedir. Paketler duruma bağlı (conditional) olabilir. Bunun anlamı nesnenin bir özelliğine

veya başka nesnelerle ilgili bir duruma bağlı olarak paketin nesneye dahil olması veya olmamasıdır. BEHAVIOUR ile başlayan satırda nesnenin işlevi yazılı olarak anlatılmaktadır. Yeni bir nesne tanımlandığında davranış kısmının anlatılması nesnenin işlevi konusunda kullanıcıya bilgi verecektir.

ATTRIBUTES ile başlayan satırdan itibaren nesnenin özelliklerini ifade eden değişkenler bulunmaktadır. Her nesnenin mutlaka tanımlayıcı bir numarası olmalıdır. Bu nesne için numara circuitEndPointSubgroupId özelliği ile tanımlanır. Diğer özellikler nesnenin genel işlevi açısından önemlidir. Örneğin circuitDirectionality özelliği diğer cihazla olan bağlantının iki yönlü mü tek yönlü mü olduğunu belirtir. NOTIFICATIONS kısmında üç adet uyarı görülmektedir. AttributeValueChange uyarısı nesnenin özelliklerinden herhangi birisi değiştiği zaman ajandan yöneticiye bir uyarı mesajı gönderilmesi gerektiğini belirtir. ObjectCreation ve ObjectDeletion uyarıları nesne yaratıldığında veya silindiğinde ajandan yöneticiye bir uyarı mesajı gönderileceğini belirtir. REGISTERED AS anahtar kelimesi bu nesneye bir nesne tanımlayıcı bir numara atar. Bu nesne için bu numara { 1 130 1 1 3 4 5 6 7}'dır.

Nesne tanımından başka nesne ile ilgili ayrıntılı açıklamalar veren GDMO tanımlamaları da yapılmalıdır. Bunlar ATTRIBUTE, NOTIFICATION ve NAME-BINDING anahtar kelimeleri ile başlayan tanımlamalardır. Bu nesne için tanımlanmasına gerek olmayan, ancak başka nesnelerde tanımlanabilecek ATTIBUTE GROUP ve ACTION tanımlamaları da mevcuttur.

Bir özellik aslında ASN.1 veri tiplerinden bir tanesi ile tanımlanan bir değişkendir. Ancak bunu GDMO tanımlaması ile belirtmek gerekir. Örnek olarak circuitEndPointSubgroupId özelliği şu şekilde tanımlanır:

circuitEndPointSubgroupId ATTRIBUTE
WITH ATTRIBUTE SYNTAX ASN1DefinedTypesModule.NameType;
MATCHES FOR EQUALITY, ORDERING, SUBSTRINGS;
BEHAVIOUR
"Recommendation X.721 : 1992" : rDNIdBehaviour,
-- The above behaviour is defined as part of discriminatorId in Recommendation X.721
circuitEndPointSubgroupIdBehaviour BEHAVIOUR
DEFINED AS
"The circuitEndPointsubgroup Id is an attribute type whose distinguished value can be used as a RDN when
naming an instance of the circuitEndPointsubgroup object class.";;
REGISTERED AS {m3100Attribute 61}; [4]

Nesne tanımlanmasından farklı olarak WITH ATTRIBUTE SYNTAX anahtar kelimesi ile özelliğin NameType ASN tipiyle tanımlanmış bir özellik olduğu

görülür. M.3100 tavsiyesindeki ASN.1 tanımlarında NameType tipi şu şekilde gösterilir:

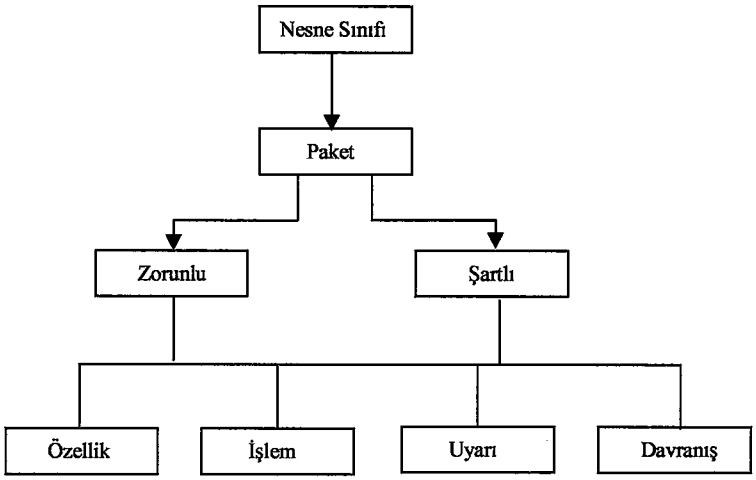
```
NameType ::= CHOICE {
                    numericName INTEGER,
                    pString GraphicString
                    }[4]
```

Yani circuitEndPointSubgroupId özelliği numericName INTEGER değeri ile veya pString GraphicString değerini alabilir.

NOTIFICATION ve ACTION için de benzer kurallar geçerlidir. Bu anahtar kelimeler sırası ile uyarı ve aksiyon (görev) anlamına gelir. Aksiyonlar bir nesne üzerinden ilgili kaynağa, cevabı verilen görev tamamlandığında gönderilmek üzere bir görev verebilmek için kullanılır. Yönetici kaynağa görev verdikten sonra cevap beklemeden başka işlemler yapabilir. Genellikle cihazla veya bağlantılarla ilgili testler bu şekilde başlatılır ve testler bittiği zaman sonuçlar ajan tarafından yöneticiye gönderilir. ATTRIBUTE GROUP anahtar kelimesi ise aynı görev ile ilgili özellik gruplarını belirtir. Bazı durumlarda özellikleri bir grup altında toplamak bir gerekliliktir. Ancak çok sık görülen bir durum değildir.

NAME-BINDING anahtar kelimesi bir nesnenin hiyerarşik olarak altında bulunduğu nesneye olan bağını gösterir. Bu bir isim bağlantısıdır. Örneğin circuitEndPointSubgroup nesnesi hiyerarşik olarak managedElement nesnesine bağlıdır ve circuitEndPointSubgroup-managedElement şeklinde gösterilir.

Şekil 5.10'da GDMO tanımlamalarının yapısı gösterilmektedir.



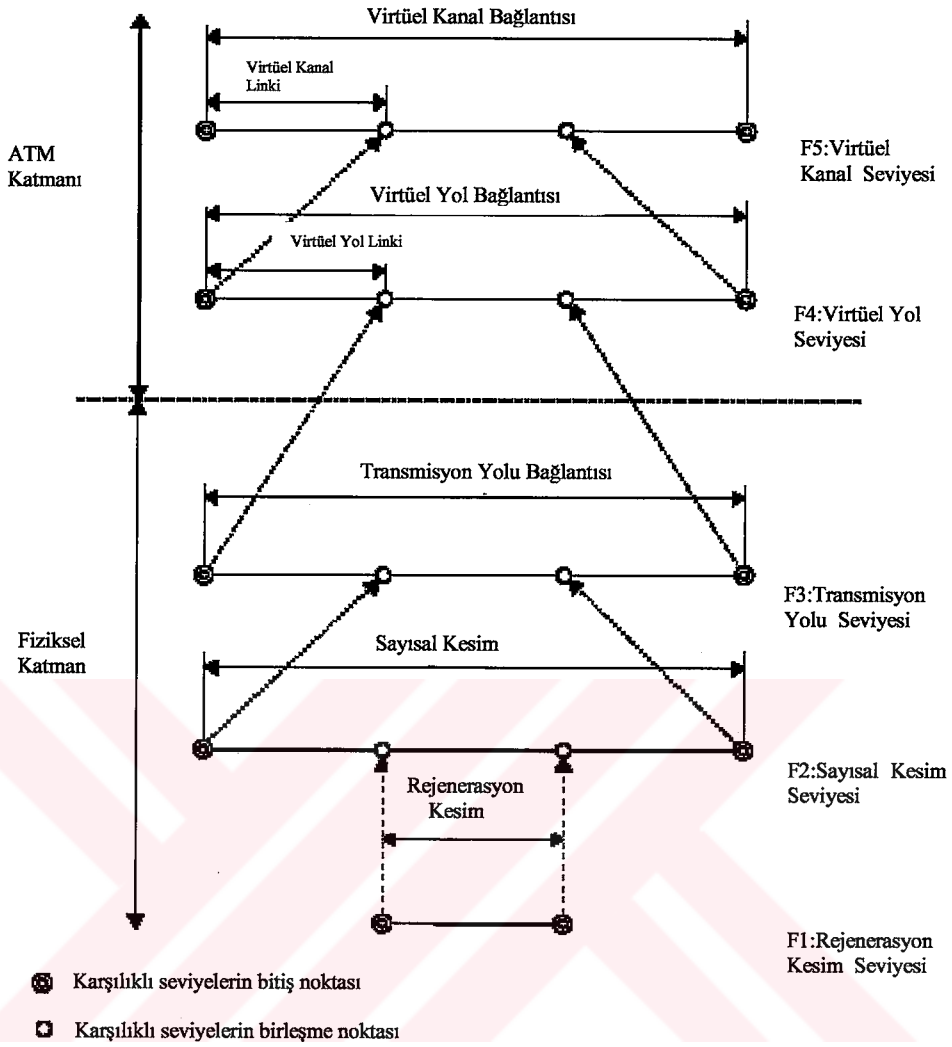
Şekil 5.10 GDMO tanım yapısı

6. TMN İLE ATM ŞEBEKESİ YÖNETİMİ

Günümüzde bakır teller üzerinden ses iletimi sağlayan sistemlere nazaran fiber optik kablolar üzerinden ve/veya havadan ses ve veri taşıyan geniş bantlı (B-ISDN) sistemler geçerlilik kazanmıştır. Bu bölümde ATM gibi geniş bantlı haberleşme alt yapısı üzerine oturan bir sistemin nasıl yönetilebileceği kavramı üzerinde durulmaktadır. Ayrıntılı bilgiler I.610 standardında mevcuttur.

Geniş bantlı sistemlerde B-ISDN özelliklerinin TMN ile yönetilmesinin amacı, şebeke operatörlerinin B-ISDN cihazlarını etkili bir şekilde kontrol edebilmesi, müşterilerin B-ISDN servislerinden yüksek servis kalitesinde yararlanabilmesinin sağlamaktır. Bu amaçla yönetilen sistem üzerinde ve sistemler arasında testler yapmak, alarm raporlamasını sağlamak ve gerekli konfigürasyonları yapabilmek yönetim açısından gerekliliktir. Geniş bantlı sistemlerin yönetimi bu açılardan incelenecektir.

ATM şebeke yönetimi ITU-T ve ATM Forum tarafından belirlenen standart ve tavsiyelere göre yapılır. Şebeke, ATM katman yönetimi, ATM virtüel yol/virtüel kanal yönetimi ve ATM performans yönetimi başlıkları adı altında yönetilir. ATM katman yönetimi, transmisyon yolu, virtüel yol ve virtüel kanalların (VC/VP) konfigürasyonunu ve hata yönetimi fonksiyonlarını içerir. Yol ve kanalların bant genişliklerinin belirlenmesi, virtüel yol tanıtıcısı ve virtüel kanal tanıtıcısı (VPI/VCI) değerlerinin limitlerinin belirlenmesi, ATM katmanlarında oluşan alarmların sezilmesi ve raporlanması işlevlerine sahiptir. ATM virtüel yol/virtüel kanal yönetimi, virtüel yol ve kanal bağlantılarının kurulması ve çözülmesi, VPI, VCI değerlerinin belirlenmesi ve kurulan bağlantıların kontrolünün yapılması görevlerini üstlenir. ATM performans yönetimi, VP, VC ve transmisyon yolunun performansının gözlenmesi amacıyla testler yapmak ve bu testler doğrultusunda sonuçları gösteren sayaçları güncellemekle yükümlüdür. Bu amaçla yapılan devamlılık kontrolü (continuity check) ve performans izleme (performance monitoring) testlerini örnek olarak gösterebiliriz.



Şekil 6.1 ATM ve fiziksel katmanda yönetilen birimler[8]

ATM katmanından başlayarak F seviyelerini özet olarak şu şekilde açıklayabiliriz:

Virtüel Kanal, ortak tek bir değer vasıtasıyla ilişkilendirilmiş ATM hücrelerinin tek yönlü taşınmasını tanımlamakta kullanılan bir kavramdır. Bu değere virtüel kanal tanıtıcısı (VCI) adı verilir ve hücre başlığının bir kısmını oluşturur.

Virtüel Yol, ortak bir tanıtıcı değer ile ilişkilendirilmiş virtüel kanallara ait hücrelerin tek yönlü taşınmasını tanımlamakta kullanılan bir kavramdır. Bu tanıtıcıya virtüel yol tanıtıcısı(VPI) adı verilir ve hücre başlığının bir kısmını oluşturur.

Transmisyon Yolu, bir transmisyon sisteminin yükünü bir araya getiren ve ayıran şebeke elemanları arasında uzanan transmisyon yoludur.

Sayısal Kesim, sürekli bit ve oktet akımlarını bir araya getiren ve ayıran şebeke elemanları arasında uzanır.

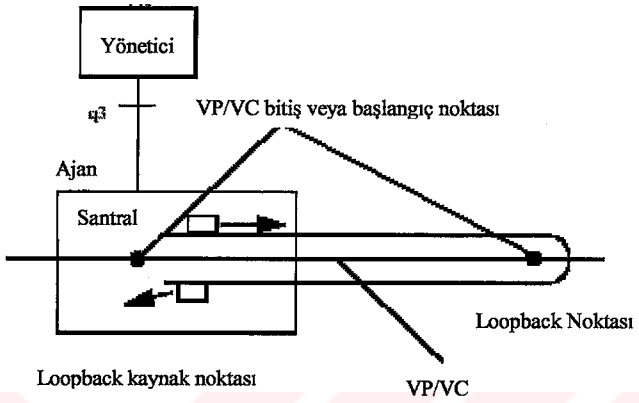
Rejenerasyon Kesimi, bir sayısal kesimin komşu iki rejeneratörü arasında bulunan parçadır.

4. bölümde TMN'de konfigürasyon, hata, performans ve güvenlik yönetimlerinin ne ifade ettiği açıklanmıştı. Bu bölümde bu kavramların ATM şebeke yönetimi açısından ifade ettiği kavramlara değinilecektir.

Konfigürasyon yönetimi, şebekenin konfigürasyonunun bilinmesi esasına dayanır. Şebeke konfigürasyonu raflar, kartlar, kart çıkışları, güç kaynakları, fiziksel sonlanma noktaları gibi fiziksel birimlerden oluşur. Bir ATM santrali açıldığında konfigürasyonun santral yazılımı ve de ajan yazılımı tarafından sezilip operatöre gösterilmesi gerekir. Bunun anlamı, ajan yazılımının operatörle bağlantı kurmaya hazır duruma gelmeden önce bu konfigürasyonu öğrenip konfigürasyon ağacını oluşturması gerektiğidir. Ayrıca başlangıçtaki konfigürasyonda bir değişiklik olduğunda bu durum operatöre bildirilmelidir. Bu işlem uyarı mesajlarıyla yapılır. Şebeke konfigürasyonu elemanlarının her birinin nesne karşılığı, gerektiğinde operatöre uyarı gönderebilecek şekilde modellenmelidir. Örneğin bir kart nesnesinde **özellik değeri değişimi** (attribute value change) uyarısı tanımlı olmalıdır ki kart çekildiğinde ya da devre dışı bırakıldığında operatöre uyarı atılabilsin.

Operatör bu konfigürasyon üzerinde bazı ATM bağlantılarını da konfigüre etmek durumundadır. Virtüel yol linklerin ve virtüel kanal linklerin çapraz bağlantılarını kurabilmeli ve kurduğu bağlantıları da çözebilmelidir. Ancak bu işlem pratikte bazı zorlukları da beraberinde getirir. Bir ATM santralinden başka bir ATM santraline doğru tanımlanacak linkler, doğru değerlerle tanımlanmazsa, operatör bir daha bu santrallere TMN üzerinden erişemeyebilir. Bu durumda santral başlangıç değerleriyle bağlantı kurabilmelidir. Kurulan linkler segment olarak ya da uçtan uça kurulabilmelidir. Segment bağlantı bir sonraki sonlanma noktasıyla kurulan bağlantıdır. Uçtan uça bağlantı bir kaç şebeke elemanından sonraki bir şebeke elemanıyla kurulan bağlantıdır.

Hata yönetimi kurulan bağlantı için herhangi bir alarm ya da sınır değerleri aşma gibi durumlar ortaya çıkarsa operatöre rapor göndermekle yükümlüdür. ATM şebekesinde hata yönetiminin en çok kullanıldığı durum ATM bağlantılarının test edilmesidir. Bu testlerden bir tanesi, ATM hücrelerinin bir virtüel yol üzerinde bulunan virtüel kanaldan yollayıp, istenilen bir noktadan geriye döndürülmesini sağlayan **Loopback** testidir. Loopback testinin şeması Şekil 6.2'de verilmiştir.

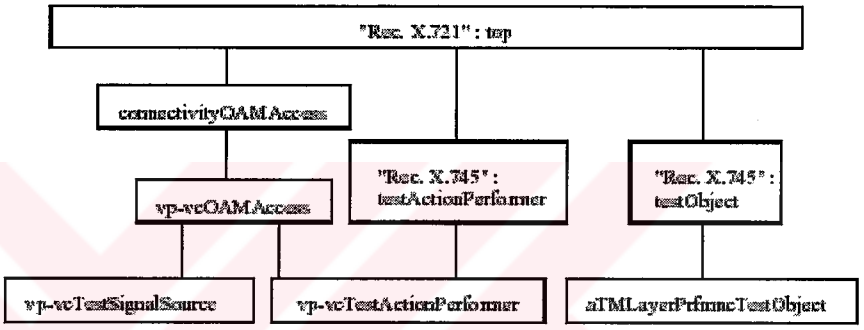


Şekil 6.2 Loopback testi şeması[8]

Bu test için yönetici gönderilecek hücre adedini ve hücrelerin geri dönüş noktasını belirler. Loopback testi için kullanılan nesnelerden iki tanesi **atmLoopbackOAMAccess** ve **atmLoopbackTestObject** nesneleridir. atmLoopbackOAMAccess nesnesi loopback geri dönüş noktasını belirler. Eğer 5 saniye içerisinde hücreler geri alınamazsa testin sonucu başarısız kabul edilir. Bu sonuç için atmLoopbackTestObject nesnesiyle anlaşılır. Santralde yapılan uygulamaya göre bu nesnelerden yeni nesneler türetilerek testler uygulanabilir.

Performans yönetimi trafik yönetimini, şebekeden trafik bilgilerinin alınıp trafik durumunun analiz edilmesini sağlar. ATM performans yönetimi, fiziksel katmanın gözlenmesi ve hücre seviyesi protokolün kontrol edilmesi görevlerini üstlenir. Sınır değerler (threshold) kontrol edilerek transmisyon yolu seviyesi ile ATM katmanları arasında meydana gelebilecek protokol davranış anormallikleri sezilir. Bu tip davranışları sezebilmek için **performans testleri** uygulanır. Bu testler sonucunda ortaya çıkan durumlardan biri transmisyon hatalarından kaynaklanan düzeltilemeyen hücre hataları olabilir ki bu hücreler yok edilir. Hücrelerin yok edildiği ikinci bir durum da protokolde belirlenmiş olan hücre biçiminin dışına çıkılmış olmasıdır.

0Performans testlerinde kullanılan nesneler ilgili VP veya VC nesnesi üzerinden yaratılan nesnelerdir. Bu nesnelerde bulunan özelliklerle istenilen performans değerleri görülebilir. Bu değerler o anda görülen değerler olduğu gibi log olarak tutulan belli zaman aralığındaki değerler de olabilir. ATM performans testleri sonunda ölçülmesi gereken değerler şunlardır: Hücre hata oranı, hücre kayıp oranı, önemli derecede hatalı hücre bloke edilme oranı, hücre yollanamama oranı, hücre transfer gecikmesi, hücre gecikme değişimi. Bu değerlerin ölçülmesi **atmLayerPerformanceTestObject** nesnesi ve bu nesneden türetilen nesneler üzerinden yapılabilir. M.3611E tavsiyesinde kullanılması öngörülen nesne hiyerarşisi Şekil 6.3'te şu şekilde gösterilir.



Şekil 6.3 Performans test nesneleri hiyerarşisi[8]

Bu şekilde gösterilmesine rağmen, 8. bölümdeki uygulamada gösterildiği üzere bu nesnelerden farklı şekilde aynı testleri uygulayan ve gerekli sonuçları elde eden farklı nesneler de bulunur. Örneğin **devamlılık kontrolü (continuity check)** testi için kullanılan bidirectionalPerformanceMonitoring nesnesi sayesinde bir VP veya VC bağlantısı üzerinden belirli süreler boyunca hücre gönderme testi yapılabilir.

Güvenlik yönetimi oturumu başlatan taraf için belirli tanıtıcıların bulunmasını gerektirir. Bunun anlamı, güvenliği sağlamak için ancak belirli VPI/VCI'ların kullanılması gerektiğidir. Bunun için doğrulama mekanizması kullanılır. ATM şebeke elemanlarına ancak onaylanan kullanıcıların erişebilmesinin sağlamak da güvenlik yönetiminin görevidir. Onaylanan ve onaylanmayan kullanıcılar hakkındaki bilgiler de tutulmalıdır. Bu amaçla kayıt tutulmalı ve nesne yaratılmalıdır.ATM şebeke yönetimi ile ayrıntılı bilgiler referanslar bölümünde verilen kaynaklarda bulunabilir.

7. TMN UYGULAMALARINDA KARŞILAŞILAN ZORLUKLAR

Özellikle geniş bantlı sistemlerin yönetimi için TMN en uygun seçenek olarak görünmektedir. Ancak uygulamada bazı zorluklar ortaya çıkmaktadır. TMN yazılımları uygulaması oldukça zor yazılımlardır. Her ne kadar TMN çözümleri sunan firmalar mümkün olduğunca bu zorlukları gidermeye çalışan yazılım ve donanım araçları sunmaktaysalar da, TMN uygulamalarını yazan yazılımcıların bir ön eğitimden geçmeye ihtiyaçları olabilir. Büyük telekomünikasyon firmaları, genellikle ürettikleri haberleşme cihazları üzerindeki yönetici ve ajan yazılımlarını kendileri üretmez. Bunları ikinci bir firmadan satın alarak kendi cihazlarına uyumlu hale getirmek zorundadırlar. Bu bazen TMN çözümü sunan firmanın sattığı yazılım kütüphanelerinde değişiklik yapmayı da gerektirir. Ancak farklı iki firmanın kurduğu şebeke tek bir elden yönetilmeye kalkıldığında, model üzerinde veya kullanılan yazılımlarda oluşabilecek büyük farklılıklar şebekenin doğru bir şekilde yönetilmesini engeller. Bunun nedeni her firmanın sunduğu çözümlerin platform, yazılım dili ve fonksiyonel olarak farklılık göstermesidir.

TMN protokolünün karmaşık bir protokol olduğundan bahsedilmişti. Bu karmaşıklık yüzünden SNMP (Simple Network Management Protocol) ve CORBA (Common Object Request Broker Application) gibi daha basit seçenekleri seçen firmalar da olmuştur. SNMP adından da anlaşılabileceği gibi TMN'e göre daha basit bir protokoldür. Asıl amacı, yönlendirici gibi internet üzerinde kullanılan cihazların yönetimini sağlamaktır. ATM santrallerinin de yönetimini sağlamak amacıyla SNMP içinde MIBII bilgi modeli tanımlanmasına karşın, TMN'in sağladığı etkinliği sağlayamaz. Bunun sebebi SNMP'nin nesne yaratabilme ve aksiyon tanımlama imkanının olmamasıdır. SNMP'nin sadece varolan nesne özelliğini değiştiren set ve nesne bilgilerini alan get özellikleri vardır. Son yıllarda yapılan çalışmalar bu eksiklikleri gidermeye çalışmaktadır. CORBA ise bir nesneyle ilgili işlemleri herhangi bir istekte bulunmadan, nesneye doğrudan erişerek yapabilir. TMN'den farklı bir mimariye sahiptir. Ancak OSI katmanları ve CMIP protokolüyle uyumlu

hale de getirilebilir. Bazı firmalar yöneticinin görevlerini bir kaç tane yöneticiye paylaştırırken CORBA kullanmayı tercih etmişlerdir. CORBA IDL (Interface Definition Language) kullanılarak oluşturulan bir protokoldür. Ancak CORBA, CMIP ve SNMP gibi bir yönetim protokolüne sahip değildir. IDL TMN'in ASN.1 ve GDMO tanımlama dillerine benzer. Nesne uyumludur. Bu yüzden CORBA'nın bazı özelliklerinden yararlanmak isteyen firmalar TMN yapısı üzerine CORBA oturtmuşlardır. En önemli faydası, yöneticinin yükünü farklı yöneticilere dağıtabilmesidir. Ajan açısından faydası, nesne bilgilerinin alınması için ASN.1 veri tipi yerine kullanıcının istediği bir veri tipinin kullanılabilmesidir. SNMP ile TMN'in farkına örnek verirsek, bir ATM santral üzerinde hem SNMP hem de TMN uygulaması geliştirilmek istenilirse, protokoller farklı olduğu için geliştirme ayrı ayrı yapılmak durumundadır. Oysa ki nesne modelleri çok benzerdir. Ancak SNMP ajanıyla bir ATM santralinin yönetimi pek de etkili olmayacaktır. Santral yönlendirici özellikleri gösteriyorsa SNMP kurulması şarttır. Önceden TMN ile kurulan bir sistem varsa bu ikisini entegre etmek oldukça problem yaratacaktır.

TMN uygulamalarını geliştirmek için kullanılan platform sorunları da hala tam olarak giderilememiştir. Kullanılan programlama dillerinin derlendiği derleyiciler ve bağlayıcılar (linker) platforma bağımlıdır. Bazı bilgisayar platformları her programlama dilini desteklemez. JAVA programlama dili kullanımı bu soruna bir çözüm oluşturabilir. Ancak genelde TMN uygulamaları C veya C++ dillerinde gerçekleştirilen uygulamalardır. Bunun dışında ajan ve yönetici yazılımlarının üzerinde geliştirildikleri haberleşme katmanları da platforma bağlı sorunlardan nasibini almaktadır. Örneğin UNIX platformunda çalışan OSI katmanı, MS-WINDOWS platformunda çalışmaz. Bunun tersi de geçerlidir. Bu yüzden hem UNIX, hem de MS-WINDOWS ortamında çalışmak istiyorsanız iki ayrı haberleşme katmanı satın almak veya geliştirmek durumunda kalabilirsiniz. TMN yöneticisi için sisteme eriştiği yer kullanıcı arabirimidir. Kullanıcı arabirimi konusunda bir standardizasyon yoktur. TMN için standardizasyon hala kapanmamış bir konudur. Eğer yönettiğiniz cihazın bazı bölümlerini standarttaki nesnelerle yönetemiyorsanız, bunun için yeni nesneler tanımlayabilirsiniz. Ancak bu işlem standarttan uzaklaşmaya yol açabilir. Farklı yöneticilerin sisteme dahil edilmesine engel teşkil edebilir. Üstelik TMN katmanlarıyla ilgili kesinleşmiş bir standart bulunmadığı için sadece alt seviyelerde değil, üst seviyelerde de problem çıkabilir.

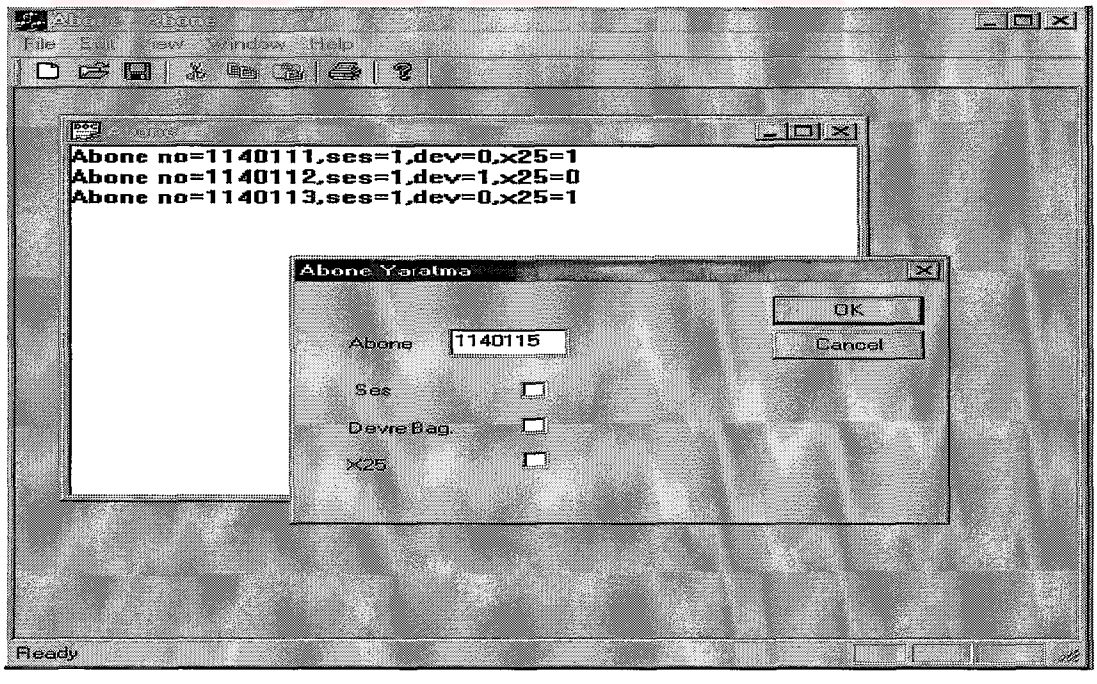
8. TMN İLE SİSTEM YÖNETİMİ UYGULAMASI

Uygulamada bir kişisel bilgisayar kullanılarak, gerçek zamanlı bir işletim sistemi olan pSOS üzerinde çalışan bir Ajan yazılımı gerçeklenmiştir. Yapılan uygulamanın ajan bölümü NORTEL NETWORKS /NETAŞ firması tarafından 1998 yılında satın alınan DSET firmasının ajan yazılımı geliştirme yazılım aracıyla hazırlanmıştır. Ajan yazılımında kullanılan diller ASN.1 ve GDMO tanımlama dilleri ve ASN.C programlama dilidir. Hazırlanan nesne modeli X.721 ve M.3600 tavsiyelerinde verilmiş olan nesnelerden veya bunlardan türetilmiş nesnelerden oluşmaktadır. Ajan yazılımının bir ATM santrali üzerinde nasıl çalıştığını gösterebilmek için Visual C++ Builder 6.0 kullanılarak ABONE isimli bir program yazılmıştır. Bu yardımcı yazılım aracılığıyla Ajan yazılımının kaynaklarla nasıl bir ilişki içinde olduğu gösterilmiştir. Ajan yazılımını test edebilmek ve nesneleri gözlemleyebilmek amacıyla DSET firmasının Agent Tester yazılım aracı kullanılmıştır. Bu yazılım aracıyla ajan yazılımıyla bağlantı kurulabilir, nesneler yaratılabilir, silinebilir, nesne özelliklerinin değeri değiştirilebilir ve nesneye aksiyonlar verilebilir. Uygulamada bu test cihazı üzerinden ajan yazılımına bağlanılarak, yardımcı bir yazılım aracılığıyla temsili kaynaklar üzerinde değişiklikler yapılarak test aracına uyarılar gönderilebilir.

TMN yazılım araçlarının oldukça pahalı araçlar olduğundan bahsedilmişti. Agent Tester ve Ajan geliştirme yazılım araçlarının NETAŞ firmasına toplam maliyeti 1998 yılı fiyatları ile 330.000 Amerikan Doları'dır ve tek bir iş istasyonu için lisans alınabilmiştir. Ajan yazılımı gerçek zamanlı sistemlerde kullanılan pSOS işletim sistemi üzerinde kurulmuştur. Nitekim ATM santrali üzerindeki bir kart üzerinde koşan pSOS işletim sistemi üzerine Ajan yazılımı da benim tarafımdan kurulmuştur. Bu yazılım şu anda Türk Silahlı Kuvvetleri'ne Netaş tarafından satılmış olan TASMUS santrallerinde kullanılmaktadır ve yazılım geliştirme süreci devam etmektedir. Kullanılan ajan geliştirme aracı ajan yazılımını UNIX işletim sisteminde

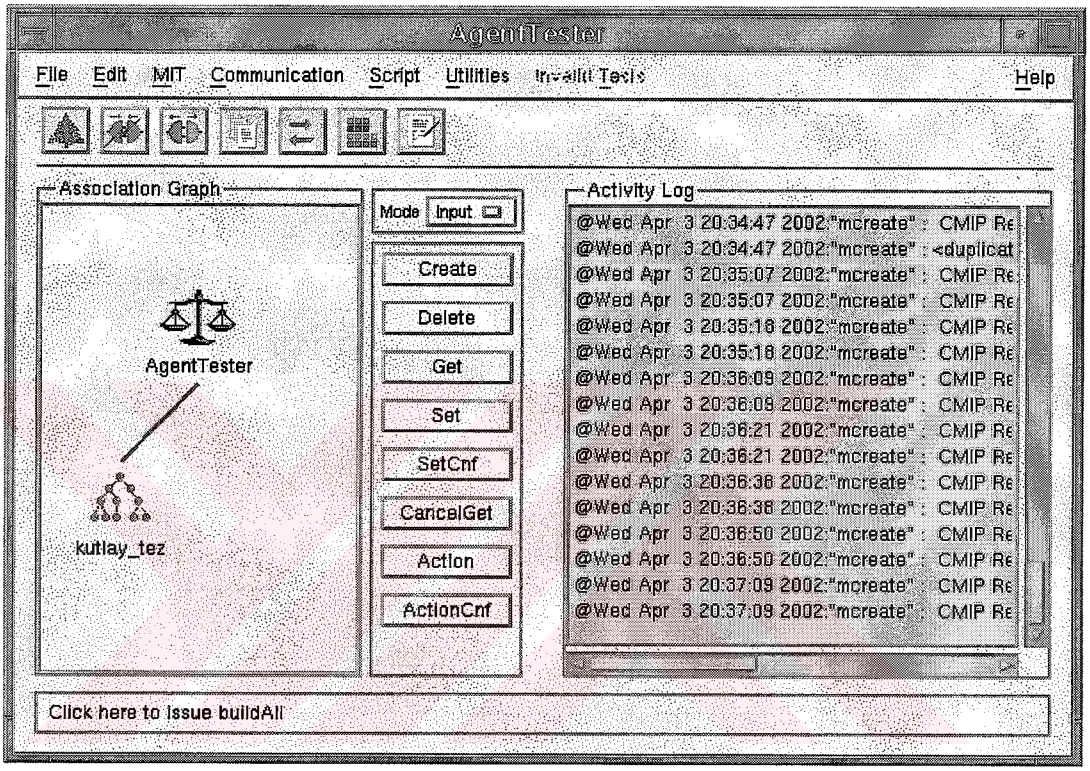
de geliştirme imkanı tanımaktadır. Ancak sadece tek bir iş istasyonu için lisanslı olduğundan geliştirme pSOS işletim sistemiyle yapılmıştır.

Ajan yazılımı Pentium 200 Mhz işlemciye sahip bir kişisel bilgisayarda kurulmuştur. Üzerinde çalıştığı pSOS işletim sisteminin sistem dosyası oluşturulmuş ve bir disket aracılığıyla bilgisayara yüklenmiştir. Kullanılan bilgisayarın C diskinin kapasitesi 512 Mbyte'ı geçmemelidir. Sistem dosyasının ismi **Psosboot.sys**'dir. İlk görevi, işletim sisteminin kullanacağı cihazların sürücülerini bilgisayara yüklemektir. Bu cihazlar ekran kartı, sabit disk, disket sürücü gibi cihazlardır. Bu işlem bittikten sonra ajan yazılımını içeren uygulama C:/ kök dizininden okunup, bilgisayarın hafizasına yüklenmekte ve ajan uygulamasının başlangıç fonksiyonu çağırılmaktadır. Uygulama **Ram.abs** isimli dosyada bulunmaktadır. Program içerisinde cihazlar tekrar yüklendikten sonra OSI katmanları çalıştırılır ve ajan programını başlatan fonksiyon çağrılır. Ajan yazılımı çalışmaya başlayınca, gerekli nesneleri yaratmak için daha önceden abone.exe programıyla bilgisayara yüklenen, abone.dat dosyasında saklanan bilgileri kullanır. Uygulamada ajan test yazılım aracına bağlı kalmamak için gerekli nesnelerin bu bilgilere dayanarak yaratılmaları sağlanmıştır. Bu dosyalara kaynak dosyaları denilmektedir. Kaynak dosyalarına bilgi girişi yapabilmek için abone.exe programı vasıtasıyla gerekli bilgilerin girilmesi gerekmektedir. Abone yaratma ekranı aşağıdaki şekilde gösterilmektedir.



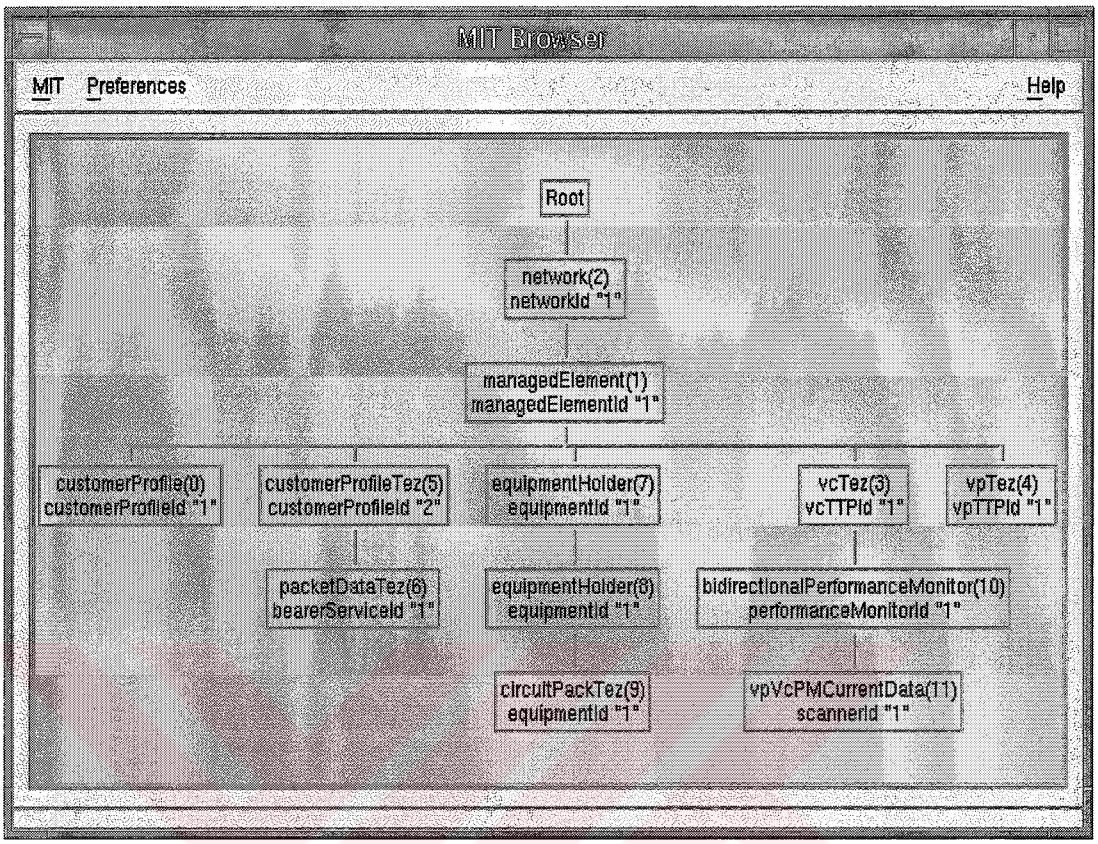
Şekil 8.1 Abone yaratma ekranı

Ajan yazılımı bu bilgileri C:/Abone dizini altındaki Abone.dat dosyasından okuyarak customerProfile standart nesnesinden türetilen customerProfileTez nesnesini bearerService nesnesinden türetilen packetDataTez ve circuitSwitchedTez nesnelerini ve de ses için yaratılan circuitSpeechTez nesneleri yaratılır. Bu nesnelerin yaratıldığı ajan ekranından gözlenebilir. Ajan test aracı ekranından gözlenen veriler Şekil 8.2-4 'de görülebilir.

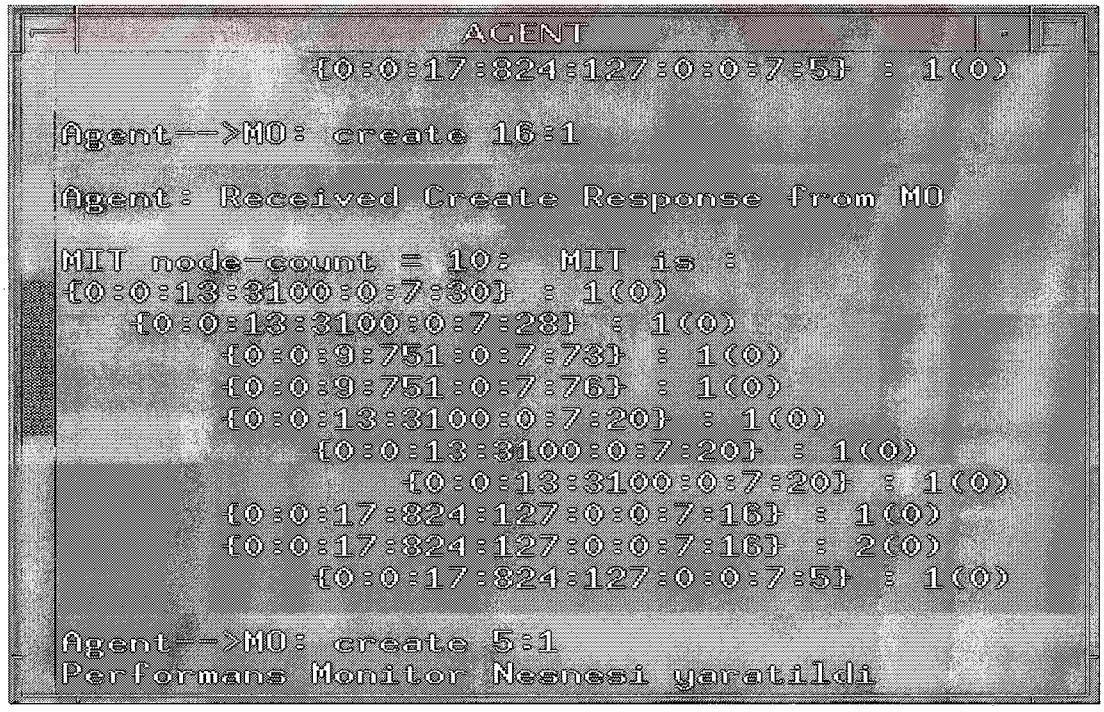


Şekil 8.2 Ajan test aracı

Şekilde görüldüğü gibi ajan test aracı kutlay_tez olarak isimlendirilen ajan yazılımına bağlanmakta ve ajandan topladığı bilgileri Şekil 8.3'de görüldüğü gibi ağaç yapısı şeklinde gösterebilmektedir. Burada görülen network ve managedElement nesneleri standart nesnelerdir. Hangi şebekedeki hangi cihaza ulaşıldığı bu nesnelerin numaralandırılmasına bakılarak anlaşılabilir. customerProfileTez nesneleri ve bunun altındaki nesneler ajan yazılımı tarafından kişisel bilgisayar üzerindeki bilgiler kullanılarak yaratılmış olan nesnelerdir. Sekil 8.4'de ise Unix üzerinde yaratılmış ajan ekranı gösterilmektedir. Ajan yazılımı nesne yaratırken eğer debug modundaysa bu tür bilgiler ekrana basılır. Uygulamadaki ajan yazılımı normal moddadır.



Şekil 8.3 MIT nesne görünümü



Şekil 8.4 Ajan nesne yaratılış ekranı

9. SONUÇLAR VE TARTIŞMA

Bu çalışmada, TMN ile şebeke yönetimi ayrıntılı olarak incelenmiştir. Yapılan uygulama, ajan yazılım programlanmasını açıklayan bir örnektir. Gerçek zamanlı bir işletim sistemi üzerinde kurulan ajan yazılımı, üzerinde çalıştığı kişisel bilgisayarda kayıtlı bilgileri kullanarak nesne yapısını oluşturmaktadır. Ajan test yazılımı kullanılarak, yazılan program doğrulanmış, uygulanabilirliği kanıtlanmıştır.

Yazılan programda, bir ATM şebekesinin kullanıcı bilgilerinin ajan yazılımında nesne olarak oluşturulması sağlanmıştır. Ajan yazılımının bu kaynaklarda yapılan değişikliklerde nasıl bir davranış sergilediği gösterilmiştir. Herhangi bir yönetici yazılımı veya bir ajan test eden programla bu nesnelere ulaşılabilir, yaratılabilir ve kaynak değerleri değiştirilebilir.

TMN geniş bantlı şebekelerin yönetimi için uygun bir yöntemdir. Katmanlı mimari yapısı ve gerçekleştirdiği fonksiyonlar açısından esneklik getirir. Bazı fonksiyonlar tek bir katmanda çalışırken, bazı fonksiyonlar değişik katmanların yönetimi altında olabilir. Fonksiyonel yapı şebeke yönetimini görevlere bölerek karmaşıklığın azalmasını sağlar. TMN'in etkili bir yöntem olmasını sağlayan bu fonksiyonel yapısıdır.

Fonksiyonel yapının karmaşıklığı azaltmasına rağmen, TMN protokollerini gerçekleştirmek pek de kolay değildir. ASN.1 ve GDMO dilleri ile tanımlanmış nesnelerle program yazabilmek, pek çok programcı için kolay bir işlem olmayabilir. Ayrıca ajan ve yönetici yazılımları, bir çok fonksiyonu yerine getirmek durumunda olduklarından karmaşık yazılımlardır. Fakat telekomünikasyon firmaları genelde bu tür yazılımları ikinci şirketlerden alır veya yaptırırlar. Bu tip firmalar gerekli yazılımları kullanarak yönetici ve ajan yazılımlarını kendi şebekelerine uygun hale getirirler.

Şebeke yönetimi için kullanılan yegane metot TMN değildir. Özellikle yerel alan şebekelerinin yönetiminde kullanılan SNMP protokolünün, geniş alan

şebekelerinde de kullanılabilmesi için SNMP nesne modeline eklemeler yapılmıştır. SNMP, TMN'e göre daha basit bir protokoldür. Bunun sebebi daha az mesaj içermesi ve bu mesajların içeriğinin de daha basit olmasından kaynaklanır. Ancak, yine bu sebepten SNMP ile gerçekleştirilebilecek işlemler TMN'e göre kısıtlı işlemlerdir. Örneğin bir ATM santralindeki bilgileri alabilmek için TMN yöneticisi ajana tek bir mesaj gönderir ve ajan nesne bilgilerini sırayla yöneticiye gönderir. Yönetici her nesne için mesaj göndermek zorunda değildir. SNMP yöneticisi ise her bir nesne bilgisi için bir mesaj göndermek zorundadır. Karmaşık şebekeler oldukça fazla değişken parametrelerle çalıştıkları için, TMN protokolüyle yönetimi SNMP'ye göre avantajlı olacaktır. Buna karşın protokol gerçeklemesi SNMP ve türevlerine göre karmaşık olacaktır. Bu karmaşıklığı azaltabilmek amacıyla, CORBA benzeri yazılımlar TMN protokolüne uygulanarak, nesnelere protokolden bağımsız olarak doğrudan erişim sağlanabilmektedir. Sistem güvenliği açısından bakıldığında, TMN bağlantılı olduğu ve güvenlik doğrulama fonksiyonlarını gerçekleştirdiği için avantajlıdır.

Uygulamada da görüldüğü gibi ajan yazılımı gerçek zamanlı bir işletim sistemi üzerinde çalışmaktadır. Pratikte, ATM santrallerinde ve SDH gibi transmisyon sistemlerinde bu tip işletim sistemleri kullanılmaktadır. Ancak TMN sadece geniş alan şebekelerinde değil, yerel alan şebekelerinin yönetiminde de kullanılan bir protokoldür. Bu yüzden sistemde kullanılan cihazın çeşidine göre Windows veya UNIX tabanlı işletim sistemi kullanan sistemlerde de ajan yazılımı kurulabilmektedir.

Dünyadaki telekomünikasyon şirketlerinin, yer yer cihazlara özel yönetim şekilleri kullanmasına rağmen, TMN giderek artan yoğunlukta kullanılan bir yönetim şeklidir. Gelişen yeni teknolojiler için de çözüm olarak TMN sunulmaktadır. Örnek olarak UMTS (Universal Mobile Telecommunications System) için TMN ile yönetim standartları konusunda çalışmalar yapılmaktadır.

Ülkemizde TMN ile şebeke yönetimi, farklı firmalara ait olan GSM şebekelerinde ve Türk Telekom'a ait SDH transmisyon sistemlerinde uygulanmaktadır. Bu ürünlerdeki TMN çözümleri Ericsson, Alcatel, Siemens, Nortel/Networks gibi firmalara ait çözümlerdir. Yine bu firmalar tarafından üretilmiş, Türk Telekom'a ait sayısal santrallerin yönetiminde, pek yaygınlaşmamış

olsa da, TMN ile şebeke yönetimi yer yer uygulanmaktadır. Ayrıca Aselsan ve Nortel/Networks Netaş firmaları tarafından ortak olarak gerçekleştirilen, Türk Silahlı Kuvvetleri için yapılan TASMUS projesi kapsamında, ATM santrali ve santraller arasındaki bağlantıyı sağlayan radyo cihazlarının yönetimi TMN ile yapılmaktadır. Bu projede yapılan TMN yazılımının büyük bir kısmı Aselsan ve Netaş mühendisleri tarafından yazılmıştır. Ülkemizde yapılan ilk TMN yazılımı özelliğini taşımaktadır.

Bir TMN projesinin gerçekleştirilebilmesi için gerekli olan ilk şey, yönetilmek istenen cihaza göre nesne modelinin oluşturulmasıdır. Standart nesne modelleri, yönetilecek cihaz için kullanılacak parametreler açısından yetersiz kalabilir. Böyle durumlarda ITU-T, standart nesne sınıfı esas alınarak yeni bir nesne sınıfı oluşturulmasını ya da yeni bir nesne sınıfı tanımlanmasını tavsiye etmektedir. Oluşturulacak yeni model iyi planlanmalı, projenin gelişmesine bağlı olarak, yeni eklemelerin yapılmasına olanak sağlamalıdır. Örneğin Tasmus projesinde yapılan çalışmalarda, standart abone nesneleri ile yaptığımız ilk çalışmaların performans açısından çok yetersiz kaldığı görülmekteydi. Tek bir abonenin ATM santralinde tanıtılması ortalama üç dakika kadar sürmekteydi. Gerekli model değişiklikleri yapılarak bu süre yirmi saniyeye düşürülmüştür. Bu duruma göre, standart olarak (tavsiye edilen) verilen nesnelere bire bir bağlı kalmak yerine, yönetilecek cihazın davranışına göre standart nesneden yeni bir nesne türetmek daha geçerli bir çözümdür.

Yöneticinin ajanlarla bağlantısı yan etkenlerin değişimine göre her zaman sağlıklı olmayabilir. Yönetilen birim üzerinde oluşan arıza durumunda veya rutin bakım çalışmalarında bağlantı yapılamayabilir. Bu şartlarda ajan yazılımının, yönetici ile bağlantısının olmadığı süre boyunca, üzerinde çalıştığı cihazdaki değişiklik bilgilerini saklı tutabilmesi gerekir. Yönetici bağlantı sağladığında bu bilgileri elde etmek isteyecektir.

Bağlantı kurulması gereken cihaz sayısının artışına paralel olarak yöneticinin yükü artacak ve performansı düşecektir. Bunu azaltabilmek için yöneticinin iş yükünün dağıtılması gereklidir. Örneğin konfigürasyon yönetimi, hata yönetimi, güvenlik yönetimi ve performans yönetimi için ayrı ayrı yöneticiler kullanılabilir. Ajan yazılımı da birden fazla adrese uyarı gönderilebilecek şekilde konfigüre edilebilmelidir.

Ajan yazılımlarının kaynaklarla ilişkisi olabildiğince basit olmalıdır. Kaynak bilgileri ajan yazılımının kolayca ulaşabileceği bir yerde tutulmalıdır. Tercihen ajan yazılımı yönetilen cihazın ana biriminde çalıştırılmalıdır (ATM santralinin ana kartı gibi). Bunun mümkün olmadığı durumlarda çevresel bir kart aracılığıyla ana kartla mesajlaşarak kaynak bilgilerine ulaşılabilir. Ancak bu durumda işlem süresi uzayacaktır. Kaynaklarla ilişkide dikkat edilmesi gereken bir başka nokta periyodik veri kontrolüdür. Ajan yazılım fonksiyonları yönetilen cihazda kullanılan işletim sistemi tarafından çağrılamaz. Bu yüzden örneğin bir alarm oluştuğunda, cihazdaki işletim sistemi tarafından yöneticiye alarm göndermek için ajan fonksiyonları çağrılamaz. Ajan yazılımı, alarm durumu gibi bazı verileri periyodik olarak kontrol etmek durumundadır. Bu periyotlar yöneticiden gelen isteklerin zamanıyla çakışabilir. Bu gibi durumlarda önceliğin yöneticiden gelen isteklere bırakılması gerekir. Yöneticinin isteği karşılandıktan sonra ajan yazılımı periyotları çalıştırılır ve uyarı oluşmuşsa yöneticiye gönderilir.

Birkaç yıl öncesine kadar TMN yazılımları C/C++ programlama dillerinin kullanıldığı yazılımlardı. Bu eğilim gittikçe Java tabanlı yazılımlara doğru kaymaktadır. Bunun sebebi Java'nın web ortamında da çalışabilir olmasıdır. Web sunucularının kullanılabilir olması, gerektiğinde müşteri isteklerinin internet üzerinden karşılanabilir olmasını sağlayacaktır. Örneğin kullanıcı istediği zaman kullanmak istediği servisi web sunucusu üzerinden işaretleyerek aktif olmasını sağlayabilecektir. Şu anda bu tip istekler internet üzerinden yapılabilse de otomatik değildir ve operatörler tarafından yapılmaktadır. Java tabanlı uygulamalar platformdan bağımsız uygulamalar olduğu için gerektiğinde bir iş istasyonu veya kişisel bilgisayar kullanılarak sistem yönetimi sağlanabilecektir. Daha önce C++ tabanlı TMN çözümleri sunan firmalar da artık programlarını Java tabanlı olarak değiştirmektedir.

Katmanlı mimarisi ve fonksiyonel yapısı sayesinde getirdiği faydalar nedeniyle TMN, karmaşık şebekelerin yönetimi için kullanılması uygun bir protokoldür. Yazılım uygulamalarında çeşitli zorluklar içermesine rağmen, karmaşık bir şebekenin yönetimi için gerekli olan tüm özelliklere sahiptir.

KAYNAKLAR

[1] M.3000, 1994. Overview of TMN Recommendations, *ITU-T Tavsiyesi*

[2] M.3010, 1996. Principles of Telecommunications Management Network, *ITU-T Tavsiyesi*

[3] M.3020, 1997. TMN Interface Spesification Methodology, *ITU-T Tavsiyesi*

[4] M.3100, 1999. Generic Network Information Model , *ITU-T Tavsiyesi*

[5] M.3200, 1997. TMN Management Services and Telecommunications Managed Areas:Overview, *ITU-T Tavsiyesi.*

[6] M.3320, 1997. Management requirement frameworks for the TMN X-Interface , *ITU-T Tavsiyesi*

[7] M.3400, 1997. TMN Management Functions, *ITU-T Tavsiyesi*

[8] M.3611, 1997. Test Management of the B-ISDN ATM layer using the TMN, *ITU-T Tavsiyesi*

[9] X.208, 1998. Specification of Abstract Syntax Notation One, *ITU-T Tavsiyesi*

[10] X.217, 1996. Service Definition for the Association Control Service Element, *ITU-T Tavsiyesi*

[11] X.219, 1988. Remote Operations:Model, Notation and Service Definition, *ITU-T Tavsiyesi*

[12] X.710, 1997. Common Management Information Service, *ITU-T Tavsiyesi*

[13] X.712, 1996. Common Management Information Protocol, *ITU-T Tavsiyesi*

[14] X.722, 1992. Guidelines for The Definion of Managed Objects, *ITU-T Tavsiyesi*

[15] Q.811, 1997. Lower layer protocol profiles for Q3 and X interfaces, *ITU-T Tavsiyesi*

[16] Q.812, 1997. Upper layer protocol profiles for Q3 and X interfaces, *ITU-T Tavsiyesi*

[17] **Divakara K. Udupa**, 1999. TMN Telecommunications Management Network, McGraw-Hill, New York.

[18] **Salah Aidarous and Thomas Plevyak**, 1994. Telecommunications Network Management into the 21'st Centuary, IEEE Press, New York.

[19] **Thomas Meserole**, 1997. ATM Network Management, *ATM Year 97*, San Jose CA, USA, June 23-27.

[20] **Kutlay Tetik ve Basri Serdaş**, 1999. Telecommunications Management Network, *Sunum,* Nortel Networks Netaş, İstanbul

ÖZGEÇMİŞ

Kutlay TETİK 1975 yılında Adana'da doğdu. İlk öğrenimini Adana Ziyapaşa İlkokulu'nda, orta öğrenimini Adana Anadolu Lisesi'nde tamamladı. Lise öğrenimimin iki yılını Kayseri Fen Lisesi'nde tamamladıktan sonra son yılını Adana Bilfen Lisesi'nde tamamladı. 1993 yılında başladığı İstanbul Teknik Üniversitesi Elektronik ve Haberleşme Bölümü'ndeki mühendislik eğitimini 1997 yılında tamamladı ve aynı yıl yüksek lisans öğrenimine başladı. 1997 Kasım ayından itibaren Nortel Networks/Netaş'ta yazılım tasarım mühendisi olarak çalışmaktadır. Netaş'ta DRX4-Dicle kırsal alan ISDN santralinde Türk Telekom için geliştirilen Metalic Test Access projesinde, Kara Kuvvetleri Komutanlığı için geliştirilen Tasmus ATM-ISDN santrali projelerinde görev yaptı. Üç yıldır ATM santral yönetimi üzerine Netaş'ta görev yapmaktadır.