EGE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ

(YÜKSEK LİSANS TEZİ)

DAĞITIK PACS SİSTEMLERİ ÜZERİNDEN TIBBİ GÖRÜNTÜLERE ERİŞİMİN SAĞLANMASI

Tolga Utku ONBAY

Tez Danışmanı: Doç. Dr. Aylin KANTARCI

Bilgisayar Mühendisliği Anabilim Dalı

Bilim Dalı Kodu : 619.01.00
Sunuş Tarihi : 10/09/2009

Bornova-İZMİR

2009

Tolga Utku ONBAY tarafından yüksek lisans tezi olarak sunulan “Dağıtık PACS Sistemleri Üzerinden Tıbbi Görüntülere Erişimin Sağlanması” başlıklı bu çalışma E.Ü. Lisansüstü Eğitim ve Öğretim Yönetmeliği ile E.Ü. Fen Bilimleri Enstitüsü Eğitim ve Öğretim Yönergesi’nin ilgili hükümleri uyarınca tarafımızdan değerlendirilerek savunmaya değer bulunmuş ve 10 Eylül 2009 tarihinde yapılan tez savunma sınavında aday oybirliği ile başarılı bulunmuştur.

Jüri Üyeleri:		**İmza**
Jüri Başkanı	**: Doç.Dr. Aylin KANTARCI**	**.....**
Raportör Üye	**: Yrd.Doç.Dr. Rıza Cenk ERDUR**	**.....**
Üye	**: Yrd.Doç.Dr. Şen ÇAKIR**	**.....**

ÖZET

DAĞITIK PACS SİSTEMLERİ ÜZERİNDEN TIBBİ GÖRÜNTÜLERE ERİŞİMİN SAĞLANMASI

ONBAY, Tolga Utku

Yüksek Lisans Tezi, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi: Doç. Dr. Aylin Kantarcı
Eylül 2009, 137 sayfa

PACS (Picture Archiving and Communication System); sağlık merkezleri içerisinde bulunan görüntüleyici iş istasyonlarını, görüntü arşivlerini ve görüntüleme cihazlarını, ağlar ve yazılımlar ile birbirine bağlayan sistemin tamamına verilen genel bir isimdir. Sağlık merkezlerindeki görüntüleme cihazlarında çekilen görüntüler, görüntü arşivlerine uzun süre saklanmak üzere gönderilir. Doktor ise çekilen bu görüntüleri iş istasyonu üzerindeki görüntüleme yazılımı ile inceleyebilir. Günümüzde birçok sağlık merkezi film baskısı yerine bu sistemi kullanarak; zamandan, mekandan ve paradan tasarruf etmektedir.

Birçok ülkede tıbbi verilerin ortak bir havuzda tutulduğu günümüzde, doktorlar tek bir sistem üzerinden hastanın tüm tıbbi geçmişini öğrenebilmektedir. Bu sayede daha doğru teşhis konulabilmekte ve tedavi süreci hız kazanmaktadır. Bu tip sistemlerin yararları günden güne artarken; tıbbi görüntüler, sağlık merkezleri içerisinde saklı kalmaktadırlar. Hastalar görüntülerini farklı doktorlara gösterebilmek için film baskıları veya görüntüleri içeren CD'leri taşımak durumunda kalmaktadırlar.

Geliştirilen dağıtık PACS sistemi ile beraber sağlık merkezleri arasında tıbbi görüntülerin aktarımı sağlanarak; PACS arşivlerinde bulunan tıbbi görüntülerin, ortak bir sistem üzerinden sorgulanabilmesi ve görüntülenebilmesi hedeflenmiştir. Bu sayede; tek bir sorguyla, sisteme dahil olan tüm sağlık merkezlerindeki görüntü arşivleri sorgulanabilir ve sonuçlar anında doktorun önüne dökülebilir. Böylece; hasta hangi sağlık merkezine giderse gitsin, tüm tıbbi görüntü geçmişi doktorlar tarafından erişilebilir olacaktır.

Sistemin hedefi doğrultusunda; iletişim standardı olarak DICOM (Digital Imaging and Communications in Medicine) kullanıldığı için, halihazırda sağlık merkezlerinde bulunan PACS'lar içerisindeki görüntü arşivlerinin ve iş istasyonlarının, başka bir yazılım veya donanıma gerek duymaksızın sisteme dahil olabilmesi sağlanmıştır.

Anahtar sözcükler: PACS, DICOM, Dağıtık Sistemler, Tıbbi Görüntü Aktarımı, Tıbbi Görüntü Arşivleri, Tıbbi Görüntüleme Sistemleri

ABSTRACT

ACCESSING MEDICAL IMAGES THROUGH DISTRIBUTED PACS SYSTEMS

ONBAY, Tolga Utku

MSc in Computer Engineering
Supervisor: Doç. Dr. Aylin Kantarcı
September 2009, 137 pages

PACS (Picture Archiving and Communication System) connects image viewer workstations, image archives and imaging modalities together with networks and softwares. Images taken with imaging modalities such as MR (Magnetic Resonance), CT (Computed Tomography), CR (Computed Radiography) is sent to the image archives to store them for a long time. Physicians can view these images using viewer softwares in their workstations. Lots of hospitals use these type of systems instead of printed films to save time, space and money.

These days countries collect medical informations in one common system. Physicians use these type of systems to look their patients' medical history. So they can find right diagnosis and shorten treatment period. Many of us get more benefits with these systems. But there is no such system medical images. Patients get printed films or CDs to show their images to different physicians.

With help of distributed PACS system that is developed, medical images can be transferred through hospitals. So medical images which stored on local PACS archives can be queried and accessed by using one common system. In short, all PACS systems of the participant hospitals can be queried with just one query and the results is listed right away to the physician's workstation.

Because of PACS elements are using DICOM (Digital Imaging and Communications in Medicine) standard already, DICOM communication standard was chosen to reach the goal of the system. Therefore without any need of extra software or hardware components, one can use viewers and servers of an installed PACS system.

Keywords: PACS, DICOM, Distributed Systems, Transferring Medical Images, Medical Imaging Archives, Medical Imaging Systems

TEŞEKKÜR

Başta çalışmalarımda her tür desteği sağlayan ve birçok konuda yol göstericim olan değerli hocam Doç.Dr.Aylin KANTARCI olmak üzere gerekli sabrı gösteren anneme ve sonrasında aileme, tezi bitirmemde kullandığım birçok bilgiyi bana aşılayan hocalarıma, çalışmalarımda bana yardımcı olan sevgili arkadaşlarım Kibariye GÜNEY, İsmail TOKMAK ve İrfan ÜNDEVLİ'ye ve lisansüstü çalışmalarımda en az benim kadar özverili davranan EMOT ve Atakalp Hastaneleri yöneticilerine ve çalışma arkadaşlarıma teşekkürü bir borç bilirim.

Son olarak başta dcm4che projesi geliştiricileri ve destekçileri olmak üzere çalışmalarımda gizlice bana yardım eden birçok uygulamasından yararlandığım açık kaynak topluluğuna çok teşekkür ediyorum.

İÇİNDEKİLER

Sayfa

İÇİNDEKİLER (devam)

Sayfa

İÇİNDEKİLER (devam)

Sayfa

İÇİNDEKİLER (devam)

Sayfa

İÇİNDEKİLER (devam)

Sayfa

İÇİNDEKİLER (devam)

Sayfa

ŞEKİLLER DİZİNİ

Şekil Sayfa

ŞEKİLLER DİZİNİ (devam)

Şekil Sayfa

ŞEKİLLER DİZİNİ (devam)

Şekil Sayfa

ÇİZELGELER DİZİNİ

ÇİZELGELER DİZİNİ (devam)

Çizelge Sayfa

KISALTMALAR DİZİNİ

Kısaltmalar	Açıklama
3DES	Triple DES
ACR	American College of Radiology
AE	Application Entity
AES	Advanced Encryption Standard
AS	Age String
AT	Attribute Tag
ATM	Asynchronous Transfer Mode
BT	Bilgisayarlı Tomografi
CD	Compact Disc
CS	Code String
DA	Date
DES	Data Encryption Standard
DICOM	Digital Imaging and Communications in Medicine
DIMSE	DICOM Message Service Elements
DS	Decimal String
DSA	Digital Signature Algorithm
DT	Date Time

KISALTMALAR DİZİNİ (devam)

Kısaltmalar	Açıklama
ECDH	Elliptic Curve Diffie-Hellman
ECDSA	Elliptic Curve DSA
FD	Floating Point Double
FL	Floating Point Single
FTP	File Transfer Protocol
HMAC	keyed-Hash Message Authentication Code
HTML	Hyper Text Markup Language
ID	Identifier
IDEA	International Data Encryption Algorithm
IDL	Interface Definition Language
IE	Information Entities
IOD	Information Object Definition
IP	Internet Protocol
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
IS	Integer String
ISDN	Integrated Services Digital Network

KISALTMALAR DİZİNİ (devam)

Kısaltmalar	Açıklama
ISO	International Organization for Standardization
JDBC	Java Database Connectivity
JPEG	Joint Photographic Experts Group
JRE	Java Runtime Environment
JNI	Java Native Interface
LO	Long String
LT	Long Text
MD5	Message Digest Algorithm 5
MR	Manyetik Rezonans
NEMA	National Electrical Manufacturers Association
OB	Other Byte String
OF	Other Float String
OW	Other Word String
PACS	Picture Archiving and Communication System
PbK	Public Key
PDU	Protocol Data Unit
PDV	Protocol Data Value

KISALTMALAR DİZİNİ (devam)

Kısaltmalar	Açıklama
PN	Person Name
PSK	Pre-shared Key
PvK	Private Key
RAID	Redundant Array of Independent Discs
RC4	Rivest Cipher 4
RIS	Radiology Information System
RMI	Remote Method Invocation
RN	Random Number
RPC	Remote Procedure Call
RSA	Rivest Shamir Adleman
RSNA	Radiological Society of North America
RSP	Response
SAN	Storage Area Network
SCP	Service Class Provider
SCU	Service Class User
SH	Short String
SHA	Secure Hash Algorithm

KISALTMALAR DİZİNİ (devam)

Kısaltmalar	Açıklama
SMTP	Simple Mail Transfer Protocol
SOP	Service Object Pair
SRP	Secure Remote Password
SL	Signed Long
SQ	Sequence of Items
SS	Signed Short
SSL	Secured Sockets Layer
ST	Short Text
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TM	Time
UI (UID)	Unique Identifier
UL	Unsigned Long
UML	Unified Modeling Language
UN	Unknown
US	Unsigned Short
UT	Unlimited Text

KISALTMALAR DİZİNİ (devam)

Kısaltmalar	Açıklama
VM	Value Multiplicity
VoIP	Voice over IP
VPN	Virtual Private Network
VR	Value Representation
WADO	Web Access to DICOM Objects
WAN	Wide Area Network
WG	Working Group
XML	Extensible Markup Language

1. GİRİŞ

Sağlık merkezlerindeki görüntüleme cihazlarında çekilen röntgen, MR (Manyetik Rezonans), BT (Bilgisayarlı Tomografi) gibi tıbbi görüntülerin sayısal olarak saklanmasını ve doktorlar, radyologlar gibi kullanıcıların bu görüntüleri iş istasyonlarında görüntülemesini sağlayan sistemlerin bütününe PACS (Picture Archiving and Communication System) adı verilir.

PACS, DICOM (Digital Imaging and Communications in Medicine) standardını kullanarak sağlık merkezi içerisinde tıbbi görüntülerin saklanmasını ve erişilebilmesini etkin ve hızlı bir şekilde sağlamaktadır. Görüntülere sağlık merkezi dışından erişebilmek için, yani tele-radyoloji uygulamaları için ise; Web sunucusu, VPN (Virtual Private Network) gibi değişik çözümler sunulmasına karşın etkin olarak kullanılan ve standart haline gelmiş bir yöntem hala geliştirilebilmiş değildir. Java, Ajax gibi teknolojiler kullanılarak Web sunucusu üzerinden sunulan görüntüler üzerinde yakınlaştırma, uzaklaştırma, parlaklık veya netlik ayarlarını değiştirme gibi temel görüntü işleme işlemleri, iş istasyonlarındaki gibi rahat bir şekilde yapılamamaktadır. VPN çözümünde ise; güvenlik sorunlarının yanı sıra her kullanıcı için ayrı ayrı yapılandırma gerekmektedir. Tele-radyoloji uygulamaları için kullanılan her iki yöntemde de ayrı bir sunucu gerekmekte ve bu da sağlık merkezleri için ayrı bir maliyet anlamına gelmektedir.

Birden fazla sağlık merkezinde çekilen görüntülere, iş istasyonları üzerinden erişmenin Web sunucusu ile karmaşık, VPN ile neredeyse imkansız ve oldukça zahmetli olduğunu söylemek, tek bir sunucuda yaşanılan sorunlar göz önüne alındığında çokta yanlış sayılmaz.

Geliştirilen sistem olan dağıtık PACS ile, birden fazla sağlık merkezinde bulunan PACS arşivlerindeki görüntülerin sisteme dahil olan iş istasyonları tarafından sorgulanması ve erişilmesi amaçlanmaktadır. Ayrıca halihazırda sağlık merkezlerinde var olan PACS arşivlerini ve görüntüleyici iş istasyonlarını kullanabildiği için ek bir maliyet gerektirmemektedir.

Sisteme eklenecek PACS arşivlerine sayısal olarak herhangi bir sınırlandırma getirilmediği için, sistem her ölçekte kullanılabilecektir.

Sistem; coğrafi olarak birbirinden ayrı iki PACS arşivinin ortak olarak sorgulanmasında kullanılabileceği gibi ulusal ölçekte tüm sağlık merkezlerinde bulunan arşivlerin aynı anda sorgulanabilmesinde de kullanılabilir.

Dağıtık PACS, ulusal ölçekte kullanıldığı takdirde bir çok avantaj sağlayabilir. Sistemin avantajları şu maddeler altında incelenebilir:

- *Hastanın tıbbi geçmişi*: Günümüzde batı ülkelerinin birçoğunda hastaların tıbbi geçmişleri tek sistem altında toplanmış ve doktorlar tarafından erişilebilir durumdadır. Doktor hastanın geçmişteki tıbbi bilgilerine bakarak; hem daha doğru hem de daha kısa sürede tanı-tedavi süreci tamamlanmış olmaktadır. Hastanın tıbbi görüntülerinin tek bir sistem üzerinden doktora sunulması durumunda ise tanı-tedavi sürecine daha fazla yardımcı olunacaktır.
- *Film baskısı*: Günümüzde hastalar çekilen görüntüleri farklı doktorlara göstermek isteyebilmektedir. Bu durumda; görüntülerini röntgen filmine bastırıp yanlarında götürmektedirler. Dağıtık PACS ile çekilen görüntülerin filme basılmasına gerek kalmaz. Doktor halihazırda var olan bir PACS görüntüleyici ile hastanın tüm görüntülerini inceleyebilir.
- *Filmin yıpranması/kaybolması*: Filmin yıpranması veya kaybolması durumunda hastalar genellikle filmi çektirdikleri sağlık kurumlarından tekrar baskı almaktadırlar. Dağıtık PACS sistemi ile hastaların film baskısına ihtiyacı kalmadığından dolayı filmleri kaybetmeleri ya da filmlerin yıpranması söz konusu olmaz.
- *Radyolog ihtiyacı*: Radyologlar, çekim sonrası görüntüleri inceleyerek rapor yazarlar. Bu yüzden görüntüleme cihazı bulunan birçok sağlık merkezinde radyolog bulundurulması gerekmektedir. Dağıtık PACS ile sağlık merkezleri çekilen görüntüleri farklı yerlerdeki radyologlara inceleterek rapor yazdırabilirler. Bu sayede halihazırda var olan radyolog açığı azaltılabilir.
- *Maliyet*: Filmin baskısının alınması, filmin kaybolması ya da yıpranması durumunda tekrar baskı alınması, görüntüleme cihazı olan kurumların tümünde radyolog bulundurulması hastalara ve hastanelere maddi yük getirmektedir. Geliştirilen sistem ile bu tip maddi sorunların tümü ortadan kaldırılarak tasarruf edilmesine olanak sağlayacaktır.

1.1. Benzer Çalışmalar

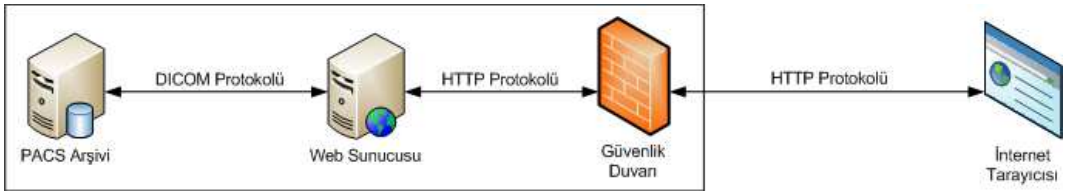
Wahle ve diğerleri (1996) "*Secure Inter-Institutional Image Communication by using DICOM-to-DICOM Gateways*" isimli çalışmalarında iki kurum arasında DICOM görüntü iletimini bir geçit (gateway) kullanarak yapmışlardır. Geçit üzerinde SMTP (Simple Mail Transfer Protocol), FTP (File Transfer Protocol) ve DICOM protokollerini deneyerek herbirinin avantaj ve dezavantajlarını çıkardıkları çalışmalarında DICOM protokolünün diğer iki yönteme göre daha faydalı ve görüntü transferi için daha kullanışlı olduğunu tespit etmişlerdir. Tez çalışması ile benzer özellikleri barındırmasına rağmen bu çalışma sadece iki kurum arasında görüntü iletimini esas almıştır.

"*Interactive DICOM image transmission and telediagnosis over the European ATM network*" isimli çalışmalarında Neri ve diğerleri (1998) tanı koyma amaçlı İtalya ve Brüksel'de bulunan üç hastane arasında ATM (Asynchronous Transfer Mode) ağı üzerinden DICOM iletişimi kullanılarak görüntü iletimini sağlamışlardır. Bu çalışmada görüntü transferinin yanı sıra iletişimin sesli ve etkileşimli yapılması ön plana çıkmaktadır. Daha çok tanı koyma amaçlı olarak düşünülen çalışma ile Avrupa'daki ATM altyapısının radyoloji alanında kullanılabilirliğini denemişlerdir. Wahle ve diğerlerinin (1996) çalışması ile benzerlikleri bulunan bu sistem de iki hastane arasında bağlantı kurmayı esas almıştır.

Becker ve Simon (2001) "*Digital image data transmission over long distances using image compression and broadband ISDN*" ismi ile yürüttükleri çalışmada Neri ve diğerlerinin (1998) yaptığı çalışmada olduğu gibi iki nokta arasında görüntülerin DICOM iletişimi kullanarak iletimini sağlamaya çalışmışlardır. Birden fazla ISDN (Integrated Services Digital Network) bağlantısı birleştirilerek bağlantı hızını yükseltmiş ve JPEG (Joint Photographic Experts Group) ile sıkıştırarak görüntülerin daha hızlı aktarılmasını sağlamışlardır.

Khludov ve diğerleri (2000) "*Internet-orientated medical information system for DICOM-data transfer, visualization and revision*" ismi ile yayınladıkları çalışmalarında DICOM görüntü iletimini İnternet tarayıcısı üzerinden sağlayan Java tabanlı bir uygulama geliştirmişlerdir. Şekil 1.1'de görüldüğü gibi İnternet üzerinden Web sunucusuna erişen istemciler tarayıcı üzerinden görüntülere ulaşabilmektedirler. Web sunucusu DICOM iletişimi ile PACS sunucusundan görüntüleri çekerek istemcide çalışan uygulamaya HTML protokolü ile göndermektedir. Çalışmalarında güvenliğe de değinen yazarlar,

güvenliği Java şifreleme paketini kullanarak RSA algoritması ile sağladıklarını belirtmişlerdir. Bu çalışma, Web sunucusunun sağlık merkezinin içerisinde bulunan PACS arşivleri ile iletişim kuracağı düşünülerek geliştirilmiştir. Birden fazla sağlık merkezindeki sunuculara erişim üzerine durulmamıştır. Daha önceki çalışmalarında ise İnternet tarayıcısı üzerinde çalışan görüntüleyici bir Java Applet geliştirmişlerdir (Khludov at al, 1999).



Şekil 1.1 İnternet Tarayıcısı Üzerinden PACS Arşivine Erişebilen Bir Sistemin Yapısı

"*Communication of Medical Images, Text, and Messages in Inter-Enterprise Systems: A Case Study in Norway*" isimli çalışma ile Balasingham ve diğerleri (2007) kurumlar arası sistemlerde görüntü, metin ve mesaj iletişimi üzerine PACSflow isimli bir uygulama geliştirmişlerdir. Bu Web uygulaması kullanılarak alıcı hastaneye görüntü ile birlikte hastanın tıbbi bilgilerini de içeren bir mesaj gönderilir. Bu bilgiler, şifreli bir VPN bağlantısı üzerinden alıcı tarafa iletilir. Görüntüler için DICOM'da tanımlı C-Move işlemi kullanılırken metin bazlı hasta bilgileri için ise farklı bir sistem kullanılmaktadır. Norveç'te geliştirilen bu sistem hastaneler arası kurulmuş olan 10Mbit'lik bir bağlantı üzerinde çalışmaktadır. Bu sistemin tez çalışmasından farkı, görüntülerin farklı hastanelerden incelenebilmesi için her birine gönderilmesi gerekliliğidir. Ayrıca bu yüzden görüntü gereksiz yere birçok hastanenin PACS arşivinde depolanır.

Morales ve diğerleri (2004) "*Cardiac surgery and cardiological data integration between remote structures*" isimli çalışmalarında görüntüleme ünitelerinden gelen görüntüleri DICOM iletişimi kullanarak DICOM uyumlu iş istasyonlarına gönderme üzerinde durmuşlardır. Diğer çalışmalardan farklı olarak görüntüler çekildiği anda iş istasyonlarına gönderilir. Bu sayede operasyon esnasında diğer hastanelerdeki uzmanlardan destek alınabilir. Görüntüler TLS ile şifrelenerek uzak iş istasyonuna gönderilir. Benzer bir çalışma Gutierrez ve diğerleri (1999) tarafından yapılmıştır. Brezilya'da geliştirilen bu sistemin iletişim altyapısı ATM kullanılarak sağlanmıştır.

"*PACS through Web Compatible with DICOM Standard and WADO Service: Advantages and Implementation*" isimli çalışmaları ile Koutelakis ve

Lymperopoulos (2006), DICOM standardının son bölümünde tanımlanmış olan WADO (Web Access to DICOM Objects) erişimi ile PACS arşivinde bulunan görüntüleri İnternet tarayıcısı ile görüntülemeyi amaçlamıştır. Diğer çalışmalardan farkı Web üzerinden erişimi WADO ile sağlamalarıdır. Bu çalışma da tek bir sağlık merkezi içerisindeki PACS sunucusunun görüntülerini İnternet'ten erişime açmaktadır.

Tohme ve diğerleri (2006) "*The Evolution of Distributed Diagnosis: Teleradiology As a Case Study*" isimli çalışmalarında hastaneler arası görüntülerin yönetimi ve dağıtık olarak tanı koyulabilmesi için "*Grid Computing*" yöntemini önermişlerdir. Öneri olarak sunulan bu yöntem oldukça özgün bir çalışma olacaktır.

Literatür araştırması sonucunda bulunan yukarıdaki çalışmalarda ATM, ISDN gibi farklı internet bağlantıları üzerinden görüntü aktarımı, internet tarayıcıları üzerinden teleradyoloji uygulamaları gibi konular üzerinde yoğunlaşılmıştır. "*Grid computing*" üzerine değinen çalışmanın dışındakiler iki sağlık merkezi arasında görüntü aktarımını sağlamaktadır. "*Grid computing*" üzerinde duran çalışma ise öneri olarak sunulmuştur. Fakat bu tarz bir sistemin kullanımı sağlık merkezlerine yeni sunucu bilgisayar almaları gerektirdiği için maddi yük getirecektir.

Dağıtık PACS sisteminin asıl hedefi iki sağlık merkezi arasında görüntü iletimi değil, birçok sağlık merkezinde bulunan PACS sistemlerinin tek bir sistem üzerinden sorgulanabilmesi ve sorgulanan görüntülerin görüntülenebilmesidir. Bunun yanı sıra, dağıtık PACS sistemi, sağlık merkezlerine herhangi bir maddi yük getirmemesi göz önünde bulundurularak tasarlanmış ve geliştirilmiştir. Yani geliştirilen dağıtık PACS sistemi benzerlerinden farklı işlevlere ve hedeflere sahiptir.

Sonraki bölümlerde Dağıtık PACS sisteminin nasıl geliştirildiğinin yanı sıra geliştirme araç ve teknolojileri de anlatılacaktır.

Temel Bilgiler başlıklı 2. bölümde dağıtık PACS sisteminin tasarlanmasında kullanılan PACS, güvenlik, dağıtık sistemler ve teleradyoloji konuları ele alınmıştır. PACS sistemlerinin altyapısını oluşturan DICOM standardı geniş bir şekilde 3. bölümde anlatılmıştır.

DICOM bölümünü izleyen Kullanılan Araç ve Teknolojiler bölümünde dağıtık PACS sisteminin geliştirilmesinde kullanılan dcm4che2 ve log4j kütüphanelerinin yanı sıra Java RMI (Remote Method Invocation) teknolojiside tanıtılmıştır.

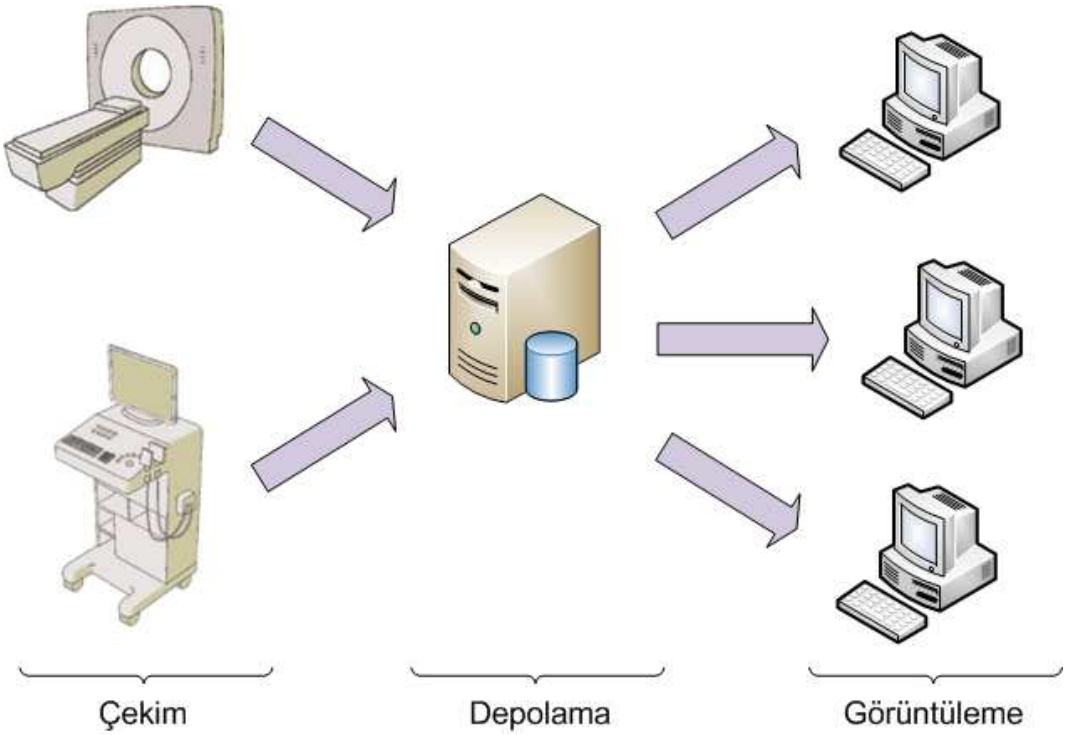
Dağıtık PACS sisteminin mimarisi, işlevleri ve gerçekleştirimi örnek bir senaryo ile birlikte 5. bölümde detaylı bir şekilde anlatılmıştır. Benzer Çalışmalar bölümünde literatürde yapılan çalışmalar incelenmiştir.

Sonuç bölümünde ise tez çalışmasının özet olarak nasıl geliştirildiği ve faydaları üzerinde durulmuştur. 8. ve son bölüm olan Öneriler bölümünde projenin devamında neler yapılacağına ve geliştirilen sisteme ne gibi özellikler kazandırılması gerektiğine değinilmiştir.

2. TEMEL BİLGİLER

2.1. PACS

PACS, BT veya ultrason gibi sayısal görüntüleme cihazları (modaliteler), çekilen görüntülerin saklandığı sayısal görüntü arşivleri ve radyologların görüntüleri inceleyebildikleri iş istasyonlarının ağlar ve yazılımlar ile birleşmesinden oluşur. Aslında PACS, sayısal fotoğraf makinelerinin kullanılmasıyla benzer özellikler taşır. Sayısal fotoğraf makinesi (görüntüleme cihazı) ile fotoğraf çekilir, çekilen görüntüler bilgisayarda depolanır (arşiv) ve diğer kişilerle paylaşılabilir (radyolog, doktor) (Pianykh, 2008). PACS'larda kullanılan bileşenler ve çekilen görüntülerin ağ üzerinde izledikleri yol Şekil 2.1'de gösterilmiştir.



Şekil 2.1 PACS'ın Bileşenleri ve Görüntü İletimi (Pianykh, 2008)

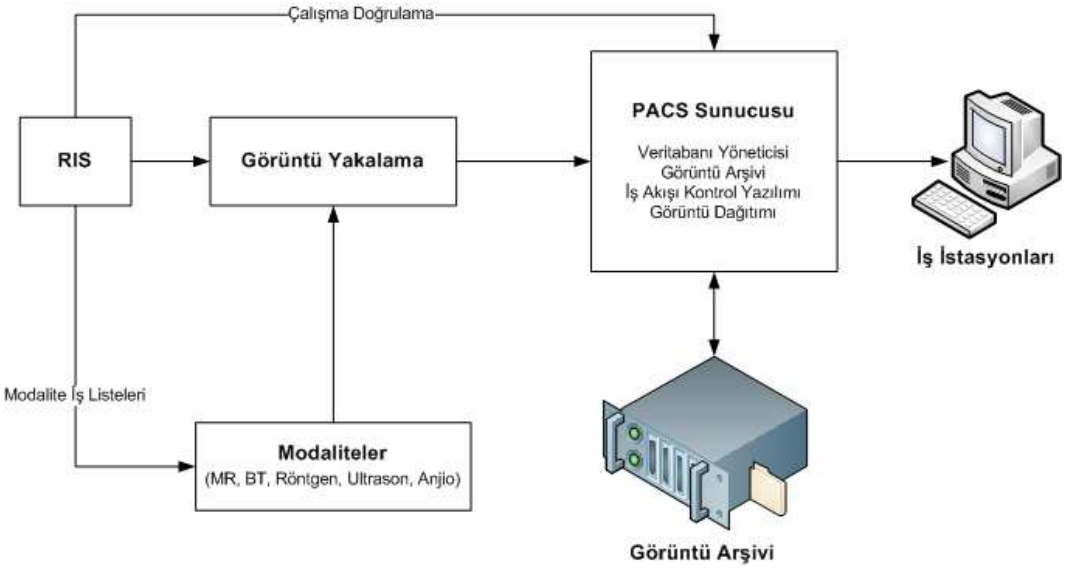
PACS, iletişim ve dosya standardı olarak DICOM'u kullanır. Bu yüzden PACS cihazları veya yazılımları DICOM uyumluluk belgesi ile birlikte gelmektedir.

2.1.1. PACS'ın elemanları

Temel bir PACS şu elemanlardan oluşur (Dreyer et al., 2006):

- Görüntüleme ünitesi
- PACS sunucusu
- Görüntüleyici iş istasyonları

Bu bileşenlerin birbirleri arasındaki etkileşim Şekil 2.2'de gösterilmektedir.



Şekil 2.2 PACS'ı Oluşturan Bileşenlerin Etkileşimi (Dreyer et al., 2006)

2.1.1.1. Görüntüleme ünitesi

Görüntüleme ünitesi çekilen görüntünün PACS'a giriş noktasıdır. Bu yüzden görüntüleme ünitesinde oluşacak hatalar sistemin işleyişine doğrudan etki etmektedir.

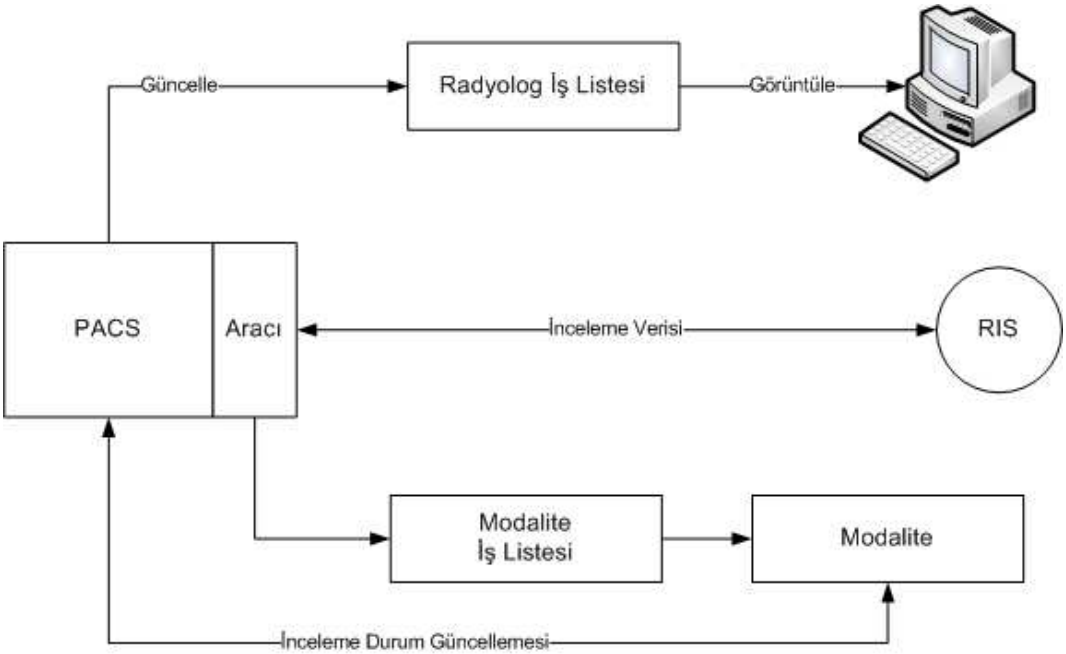
2.1.1.2. PACS sunucusu

PACS sunucusunun, çekilen görüntülerin hatasız bir şekilde depolanması, çağrılması ve getirilmesini sağlamak için doğru bir şekilde yönetilmesi gerekmektedir. Ayrıca PACS sunucusu, yönetmelikler tarafından belirlenen süre boyunca görüntüleri saklamak ile yükümlüdür.

PACS sunucusunun bileşenleri (Dreyer et al., 2006):

- Veritabanı yöneticisi (Oracle, MySQL, MS SQL, ...)
- Görüntü arşivi (RAID, Optik disk kutusu, ...)
- İş akışı kontrol yazılımı (Görüntü yöneticisi)
- Radyoloji bilgi sistemi (Radiology Information System – RIS) arayüzü

Şekil 2.3'te bir PACS ağında bulunan bu bileşenler üzerindeki iş akışının detayları görülebilir.



Şekil 2.3 PACS İş Akışı (Dreyer et al., 2006)

Veritabanı yöneticisi PACS'ın en önemli bileşenidir. Görüntünün sorgulanmasında kullanılmasının yanı sıra görüntü ile arşivdeki konumu arasındaki bağı, veritabanı tutar. Yani sunucunun dosya sisteminde tutulan DICOM dosyasının diskteki adresini (C:\PACS\ABC.DCM gibi) ve hastanın adı, soyadı, çekim tarihi gibi görüntüye ait temel bilgileri saklar. Ayrıca kendisine gelen sorguları inceleyip uygun hastanın görüntülerini geri döndürmelidir. Veritabanının cevaplayacağı sorgular, DICOM standardı içerisinde tanımlanmıştır. Veritabanı yöneticisi olarak ise; çoğunlukla ilişkisel veritabanı yöneticisi kullanılmaktadır.

Görüntü arşivi veritabanı ile birlikte çalışır. Görüntülerin uzun ya da kısa vadede ne şekilde tutulacağını, görüntüleri kurtarma politikalarını belirler. Yakın

zamanda gönderilen verilere anlık ulaşım sağlarken eski görüntüleri optik diskler, SAN (Storage Area Network) gibi yerlerde tutarlar.

Görüntü yöneticisi, PACS'ın işleyişini sağlayan bileşendir. RIS'ten ve PACS sunucusundan gelen verinin birleştiği yerdir. Görüntüleme cihazından geldikten sonra görüntülerin depolanana kadar sistem boyunca izleyeceği yolu belirler. Görüntülerin saklanmasının yanı sıra çağrılmasında da rol alır.

RIS arayüzü çekim bilgilerini ve uygun zaman çizelgesini PACS sunucusuna göndermekle sorumludur. PACS ve RIS'in mimarisine ve yapılandırmasına göre RIS arayüzünü yönetecek bir aracı gerekebilir. Aracının rolü ise PACS ve RIS arasında gerekli olan verinin müzakeresi ve eğer gerekiyorsa verinin dönüşümünü sağlamaktır.

2.1.1.3. Görüntüleyici iş istasyonları

İş istasyonu; doktorun RIS'ten elde edilen çekim bilgilerinin ve çekildikten sonra PACS sunucusundan getirilen görüntünün sonuçlarını gördüğü yerdir. PACS'ta iş istasyonları, tanı amaçlı ve inceleme amaçlı olarak ikiye ayrılır. Bu iki tip iş istasyonu arasında çözünürlük ve işlevsel fark bulunmaktadır. Tanı amaçlı kullanılan iş istasyonunda, radyolog yapılan çalışmayı (examination) yorumlar. Bu yüzden tanı amaçlı iş istasyonları daha yüksek çözünürlüğe, daha yüksek parlaklığa ve daha fazla işleve sahiptirler. Çok fazla özelliğe sahip olduklarından maliyetleri daha yüksektir ve bu tip iş istasyonlarından sağlık kuruluşlarında az sayıda bulunmaktadır.

İnceleme amaçlı iş istasyonları tanı amaçlılar kadar güçlü değillerdir. İkisi arasındaki farklar donanımsal (çözünürlük gibi), yazılımsal ya da her ikisi olabilir. İnceleme amaçlı iş istasyonları doktorların görüntülere erişimini sağlar. Görüntülerin kalitesi doktorların yorum yapabilmesi için idealdir. Doktorların görüntüleri radyoloji raporu ile birlikte incelemesine ve paylaşmasına olanak sağlar.

2.1.2. PACS'ın kullanım nedenleri

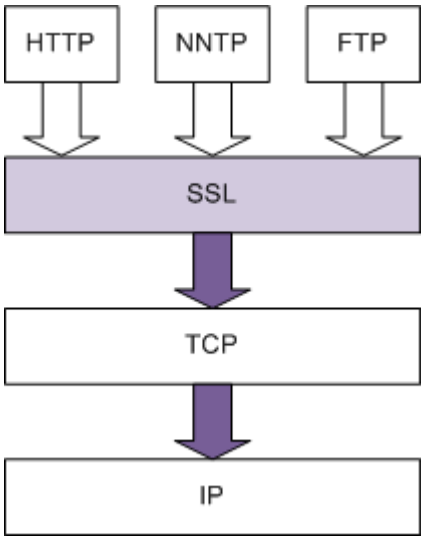
PACS'ın iki kullanım nedeni bulunmaktadır. Bunlar (Wikipedia, 2009a):

- *Çıktı almamak*: PACS ile beraber tıbbi görüntülerin çıktı alınmasına gerek kalmaz. Bu sayede hem çıktıların saklanması için gereken alandan hem de çıktı üretmek için harcanacak paradan tasarruf edilmiş olur.
- *Uzaktan erişim*: Görüntülere, hastanenin dışından erişilebilmesine olanak sağlar. Bu sayede görüntülerin hastane dışından incelenerek raporlanması ve farklı yerlerdeki uzmanların görüntülere erişimi sağlanır.

2.2. Güvenlik

TLS (Transport Layer Security) ve atası SSL (Secure Sockets Layer) İnternet gibi bilgisayar ağlarında güvenlik ve veri bütünlüğü sağlayan şifreleme protokolleridir. TLS ve SSL İnternet protokolünün (Internet Protocol – IP) iletim katmanında (Transport Layer) uçtan uca kurulan ağ bağlantılarını parça parça şifrelemekten sorumludur. Şekil 2.4 üzerinde uygulama katmanı protokollerinden gelen verilerin TCP (Transmission Control Protocol) katmanına ulaşmadan nasıl şifrelendiği gösterilmektedir. E-posta, anlık mesajlaşma, VoIP (Voice over IP) gibi uygulamalarda çok yaygın olarak kullanılmaktadır.

SSL/TLS, tıbbi görüntülerin IPv4 üzerinden DICOM iletişimi kullanılarak iletiminde en uygun yöntem olarak belirlenmiştir. IPv6'da ise IPSec yöntemi daha verimli olarak çalışmaktadır (Zhang et al, 2007). IPv4 günümüzde daha yaygın olarak kullanıldığı için tez projesinde güvenlik protokolü olarak SSL/TLS seçilmiştir.



Şekil 2.4 Ağ Üzerinden Gönderilecek Verilerin SSL ile Şifrelenmesi (Thomas, 2000)

TLS protokolü istemci/sunucu arasında kurulan iletişimde iletişimin dinlenmesini, verinin değiştirilmesini, sahte mesajların gönderilmesini önlemek için tasarlanmıştır (Wikipedia, 2009b). Çizelge 2.1'de TLS'nin kullanım nedenleri verilmiştir.

Çizelge 2.1 TLS'nin Kullanım Nedenleri (Thomas, 2000)

Kullanım	**Servis**	**Korunma**
Gizli tutmak	Gizlilik	Dinlenmek
Kimlik ispatı	Doğrulama	Sahtecilik
Bilginin doğrulanması	Mesajın doğruluğu	Değişiklik

TLS üç aşamadan oluşmaktadır. Bunlar (Thomas, 2000):

1. Algoritma desteğinin sağlanıp sağlanmadığını öğrenmek için müzakere yapılması
2. Anahtar değiş tokuşu ve kimlik doğrulaması
3. Simetrik şifrenin şifrelenmesi ve mesaj doğrulaması

İlk aşamada istemci ve sunucu kullanılacak şifrelemeyi, anahtar değiş tokuşunu, doğrulama algoritmalarını ve mesaj doğrulama kodlarını tanımlayan şifre takımını (cipher suite) müzakere ederler. Anahtar değiş tokuşu ve doğrulama algoritmaları tipik olarak genel anahtar algoritmalarıdır. Mesaj doğrulama kodları ise şifreleme alanında kullanılan hash fonksiyonlarından üretilir.

Kullanılan algoritmalar (Wikipedia, 2009b):

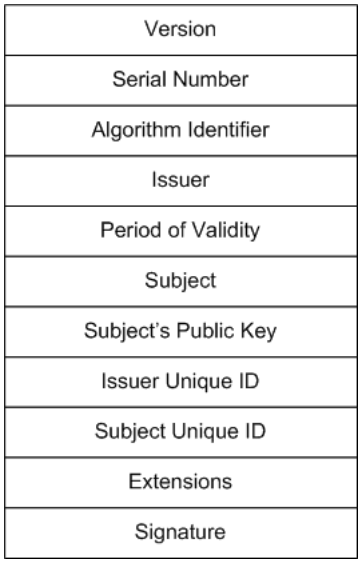
- *Anahtar değiş tokuşu için*: RSA, Diffie-Hellman, ECDH, SRP, PSK
- *Doğrulama için*: RSA, DSA, ECDSA
- *Simetrik şifreler*: RC4, 3DES, AES, IDEA, DES, Camellia
- *Şifreleme ile ilgili hash fonksiyonları için*: TLS için HMAC-MD5, HMAC-SHA, SSL için ise MD5, SHA

2.2.1. Çalışması

TLS istemcisi ve sunucusu el sıkışma prosedürünü kullanarak bağlantı şeklini müzakere ederler. Bu el sıkışma prosedürü sırasında istemci ve sunucu

bağlantının güvenliğini sağlamakta kullanılacak çeşitli parametreleri kararlaştırırlar. El sıkışma prosedürü şu şekilde çalışır (Wikipedia, 2009b):

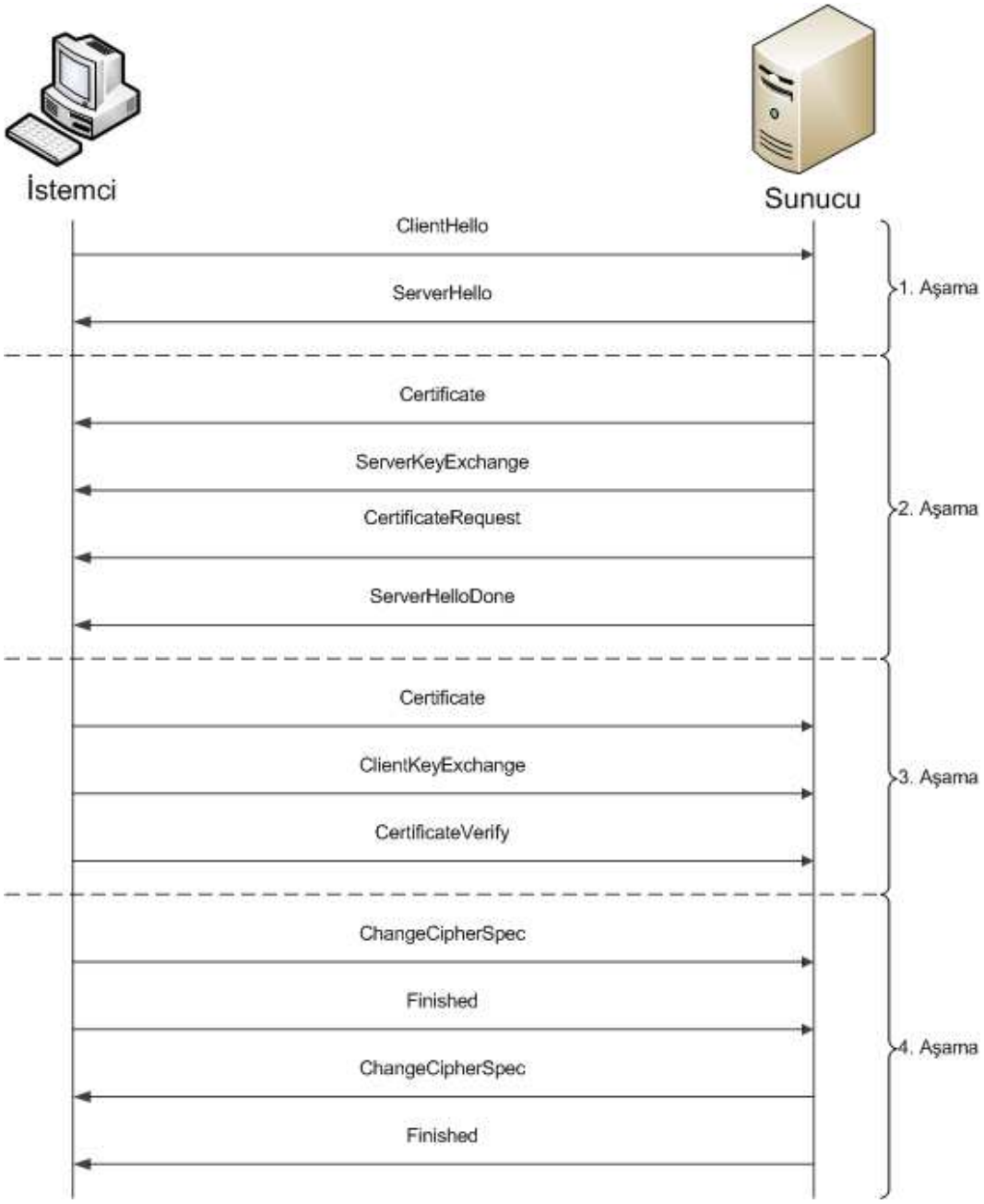
1. El sıkışma, güvenli bağlantı isteyen ve desteklediği şifre ve hash fonksiyonlarını sunucuya sunan istemcinin TLS sunucusuna bağlanması ile başlar.
2. Sunucu, istemcinin gönderdiği listeden kendisinin desteklediği ve en güçlü şifre ve hash fonksiyonunu seçer ve seçimi istemciye bildirir.
3. Sunucu bilgilerini sayısal sertifika şeklinde istemciye gönderir. Genel olarak sertifika sunucu adını, güvendiği sertifika otoritesini ve sunucunun genel şifreleme anahtarını içerir. İstemci devam etmeden önce sertifika otoritesi ile iletişime geçerek sertifikanın doğruluğunu kontrol edebilir.
4. Güvenli bağlantıda kullanılan oturum şifrelerini üretmek için istemci, sunucunun genel anahtarını (public key - PbK) kullanarak ürettiği rastgele bir sayıyı (Random Number - RN) şifreler ve bu sonucu sunucuya gönderir. Anahtarlar gizli olduğu için, gönderilen sonucu kendi özel anahtarı (private key - PvK) ile sadece sunucu deşifre edebilir. Şekil 2.5 tipik bir genel anahtar sertifikasının içeriğini göstermektedir.



Şekil 2.5 Bir Genel Anahtar Sertifikasının İçeriği (Thomas, 2000)

5. Her iki taraf rastgele numaradan şifre ve deşifre işlemleri için gerekli anahtar materyalleri üretir.

Şekil 2.6'da istemci ve sunucu arasındaki el sıkışma süreci gösterilmektedir.



Şekil 2.6 SSL ile İstemci ve Sunucu Arası El Sıkışma (Thomas, 2000)

Yukarıda adımlar doğrultusunda el sıkışma işlemi bitirilir ve bağlantı kopana kadar anahtar materyaller ile şifrelenip deşifre edilen güvenli bağlantı sağlanır. Adımlardan herhangi birinin başarısız olması durumunda TLS el sıkışma işlemi başarısızlığa uğramış sayılır ve bağlantı kurulamaz.

2.3. Dağıtık Sistemler

Dağıtık sistem, farklı bilgisayarlardaki donanım ve yazılım bileşenleri arasında haberleşme ve koordinasyonunun, sadece mesajlaşma yoluyla sağlanabildiği ağ olarak tanımlanır.

Dağıtık sistem, birtakım protokolleri işleterek tüm bileşenlerin beraberce tek bir görevi yerine getirebilmesi için bir ağdaki birden çok sürecin hareketlerini düzenleyen uygulamadır.

Dağıtık bir sistem birbirinden bağımsız bilgisayarların bir bilgisayar ağı ile iletişiminden oluşur (Tanenbaum and Steen, 2002). Bilgisayarlar ortak bir amaca istinaden birbirleri ile etkileşirler. Fakat bu sistemin kullanıcıları sistemi tek bir bilgisayarmış gibi algılar. Örneğin kullanıcı bir program çalıştırdığında, en iyi işlemciyi seçmek, giriş dosyalarını bu işlemciye aktarmak ve dönen sonuçları uygun yere koymak, sistemin yürüttüğü işlerdir. Donanımsal olarak bilgisayarlar birbirinden bağımsız olmasına karşın yazılımsal olarak aslında birdirler. Yazılımla, ağdaki bileşenler arası işlevsel uyumluluk ve kullanıcı açısından saydamlık sağlanır.

Dağıtık sistemlerde çeşitli tipteki bilgisayarların iletişimindeki farklılıklar kullanıcılardan soyutlanmıştır. Ayrıca kullanıcılar ve uygulamalar etkileşimin nerede ve ne zaman olduğuna bakmaksızın dağıtık sistem ile tutarlı olarak etkileşime girebilirler.

Ayrıca dağıtık sistemlerin genişletilmesi ve ölçeklendirilmesi kolay olmalıdır. Bazı birimlerinin geçici olarak servis dışı olmasına rağmen dağıtık sistemden beklenen, sürekli olarak çalışır durumda olmasıdır. Kullanıcı ve uygulamalar ilgili birimlerin değiştirildiğini, çalışır duruma getirildiğini veya eklenen yeni birimlerin daha fazla kullanıcıya ya da uygulamaya hizmet edeceğini fark etmemelidir.

Dağıtık sistemler tek bir sistem görüntüsü verirken farklı bilgisayar ve ağ yapılarını desteklemesi için bir yazılım katmanı halinde düzenlenmişlerdir.

2.3.1. Dağıtık sistemlerin özellikleri

Dağıtık sistemlerin özellikleri aşağıdaki gibi sıralanabilir.

- *Birden fazla süreç*: Sistem çok sayıda ardışık süreçten oluşur. Bu süreçler sistem veya kullanıcı süreçleri olabilir. Fakat her süreç bağımsız bir iş parçacığı tarafından kontrol edilmelidir.
- *Süreçler arası iletişim*: Süreçler aralarında, birinden diğerine sınırlı zamanda iletilebilen mesajlar ile iletişim kurarlar.
- *Ayrık adres alanları*: Süreçlerin ayrık adres alanları vardır. Yani ortak bellek kullanan çok işlemcili bir sistem, gerçek bir dağıtık sistem değildir.
- *Ortak hedef*: Süreçler birbirleri ile ortak bir hedefe ulaşabilmek için etkileşime geçmelidir.

2.3.2. Dağıtık sistemlerin hedefleri

Dağıtık sistemlerin birçok soruna çözüm olması ve oldukça esnek bir yapıya sahip olması, her zaman kullanılması gerektiği anlamına gelmemektedir. Dağıtık sistemlerin geliştirilme amaçları (Tanenbaum and Steen, 2002):

2.3.2.1. Kullanıcı ve kaynakların birbirine bağlanması

Dağıtık sistemlerin en önemli hedefi kullanıcıların kolayca uzak kaynaklara erişim sağlayabilmesi ve kaynaklarını diğer kullanıcılar ile kontrollü bir şekilde paylaşabilmesidir. Kaynak olarak en basit şekilde bir yazıcı, veri, dosya veya İnternet sayfası düşünülebilir. Kaynakların paylaşılmak istenmesinin nedeni olarak ilk akla gelen maliyettir. Örneğin her kullanıcı için bir yazıcı almak yerine tek bir yazıcı birçok kullanıcı ile paylaşılabilir. Ayrıca kullanıcıları ve kaynakların birbirine bağlanması bilginin karşılıklı bir şekilde kolayca paylaşılmasını sağlar. Bu paylaşıma en güzel örnek tabi ki İnternettir.

Fakat bağlanılabilirlik ve paylaşım arttıkça güvenlik daha fazla önem kazanmaktadır. İletişim kanallarının dinlenmesini önlemek için çok az çaba sarf edilmektedir. Şifreler ve diğer hassas bilgiler şifrelenmeden transfer edilmekte veya güvenilir olduğunu umduğumuz sunucularda saklanmaktadır.

2.3.2.2. Şeffaflık

Dağıtık sistemlerin diğer bir hedefi süreçlerinin ve kaynaklarının fiziksel olarak farklı bilgisayarlara dağıtıldığını gizlemesidir. Kullanıcılarına ve uygulamalarına tek bir bilgisayarmış izlenimi veren dağıtık sistem şeffaf olarak tanımlanır. Dağıtık sistemlerin şeffaf olması her ne kadar iyi bir hedefse de performans gibi şeffaflığın götürülerini de hesaba katarak şeffaflığın sağlanması çok önemlidir.

2.3.2.3. Açıklık

Açıklık her bileşenin, diğer bileşenlerle devamlı olarak etkileşime açık olması gerekliliğini ifade etmektedir. Açık bir dağıtık sistem, sözdizimi ve anlamsallığı standart kurallar tarafından tanımlanan servisler sunarlar. Örneğin bilgisayar ağları alanında standart kurallar, alınan ve gönderilen mesajların biçimini, içeriğini ve anlamını tanımlarlar. Bu kurallar protokoller adı altında toparlanır. Dağıtık sistemlerde ise servisler genellikle Arayüz Tanımlama Dili (Interface Definition Language - IDL) içerisinde yer alan arayüzlerde tanımlanır. IDL'de yer alan arayüz tanımlamaları, neredeyse sadece servislerin sözdizimlerini içerirler. Diğer bir deyişle kullanılabilir durumda olan fonksiyonların isimleri ile birlikte; parametrelerin tiplerini, geri dönüş değerlerini, fırlatılabilecek istisnai durumları (exception) tanımlarlar.

Arayüz tanımlaması arayüzlerin birbirinden tamamen farklı gerçekleştirimlerinin iki bağımsız birim tarafından derlenmesini ve aynı şekilde çalışan iki farklı dağıtık sistemin oluşabilmesini de sağlar. Düzgün tanımlamalar, tam ve tarafsızdırlar. Buradaki tanımlamaların tam olması, gerçekleştirimin yapılabilmesi için gerekli olan her şeyin belirtilmesini ifade eder.

Açık bir dağıtık sistem esnek olmalıdır. Yani farklı geliştiriciler tarafından geliştirilen farklı bileşenlerden oluşan sistemin, kolayca yapılandırılabilmesi gerekir. Ayrıca diğer bileşenleri etkilemeden, yeni bileşenlerin eklenmesi veya var olanların değiştirilmesi kolay olmalıdır. Diğer bir değişle dağıtık sistem genişletilebilir olmalıdır. Örneğin esnek bir sistemde farklı işletim sistemleri ile çalışan yeni birimlerin sisteme kolayca eklenebilmesi veya farklı bir dosya sisteminin kullanılabilmesi gerekir.

2.3.2.4. Ölçeklendirilebilirlik

İnternet üzerinden bağlanabilirlik gün geçtikçe yaygınlaşmaktadır. Bu bağlamda ölçeklendirilebilirlik dağıtık sistem geliştiricilerinin en önemli hedefleri arasında yer almaktadır.

Bir sistemin ölçeklendirilebilirliği en az üç farklı boyutta ölçülebilir. İlki; bir sistemin büyüklüğünün ölçeklendirilebilirliği, yani sisteme daha fazla kullanıcı ve kaynağın kolayca eklenebilmesidir. İkincisi; kullanıcıların ve kaynakların birbirinden çok uzaklarda olduğu coğrafi olarak ölçeklendirilebilir bir sistem olması, üçüncüsü ise birbirinden bağımsız kurumlara yayılmış olmasına rağmen yönetimi yine de kolay olan yönetsel olarak ölçeklendirilebilir bir sistem olmasıdır.

Kısacası ölçeklendirilebilir bir dağıtık sistem kullanıcıların, kaynakların ve işletim birimlerinin sayısındaki değişimlere kendini uyarlayabilmelidir.

2.3.2.5. Güvenilirlik

Dağıtık bir sistemin kullanılabilirliği, güvenilirliği ile doğru orantılıdır. Eş zamanlı olarak çalışan bileşenlerin etkileşimlerindeki karmaşıklık düşünüldüğünde güvenilirlik dağıtık bir sistem için öngörülen en zor hedeftir.

Dağıtık bir sistemin güvenilir olması için şu özellikleri barındırması gerekir:

- *Hataya dayanıklılık*: Yanlış işlemler yapmadan bileşenlerdeki hataları giderebilmeli
- *Ulaşılabilirlik*: Bazı bileşenler bozulduğunda bile servislerini sunmaya devam edebilmeli
- *Kurtarılabilirlik*: Bozulan bileşenler hatalar giderildikten sonra kendilerini sıfırlayıp sisteme tekrar dahil olabilmeli
- *Tutarlılık*: Sistem işlemleri farklı bileşenlere koordineli bir şekilde yaptırabilmeli
- *Ölçeklendirilebilirlik*: Sistem genişletildiğinde doğru bir şekilde çalışmaya devam edebilmeli
- *Performans*: İsteklere tam zamanında ve istendiğinde yanıt verebilmeli
- *Güvenlik*: Veriye ve servislere erişimi denetleyip, doğrulayabilmelidir.

2.4. Teleradyoloji

Teleradyoloji, bilgisayar ağları vasıtasıyla uzak radyoloji servisleri sunar. Uygun ağ altyapısı ve PACS yazılımlarının yapılandırılması ile DICOM görüntüleri kolaylıkla ağ dışındaki kullanıcılar ile paylaşılabilir. Bir ülkede çekilen görüntüler farklı bir ülkede incelenerek görüntülerin raporları yazdırılabilir. Bu sayede maliyetler düşürülebilir ve görüntü inceleme işlemi hızlandırılabilir. Yapılan çalışmalar ile Amerika'daki radyoloji tetkiklerinin %90'ı ülke dışında inceletilebilir olduğu belirlenmiştir (Pattynama, 2006).

Teleradyoloji uygulamalarının doktorların ve standartların öngördüğü bazı gereksinimleri yerine getirmelidir. Teleradyoloji uygulamaları (Pianykh, 2008):

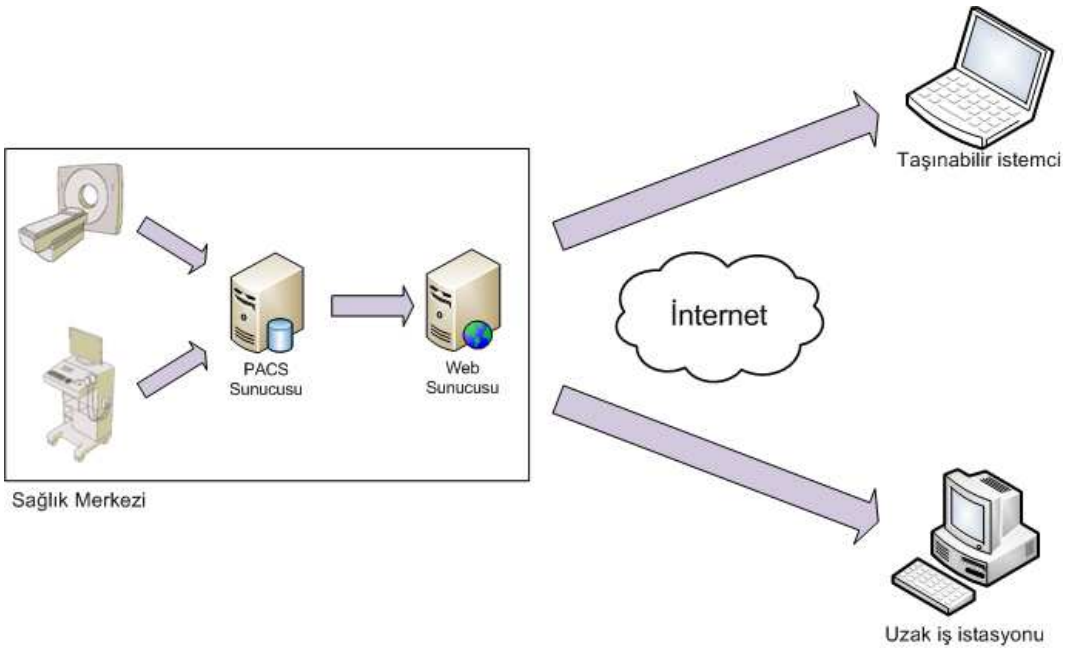
1. Tanısal kaliteyi ve tıbbi verilerin içeriğini korumalıdır.
2. Kusursuz bir entegrasyon sağlamalı ve DICOM, PACS desteği sunmalıdır.
3. Sınırlı bant genişliğine sahip ağlarda çalışabilmeli ve uzaklıktan etkilenmemelidir.
4. Birçok yazılım, donanımla ve sistemle uyumlu olmalıdır.
5. Teşhislerin doğruluğu için güvenli bir sıkıştırma yöntemi ve güvenlik sağlamalıdır.

Günümüzde bu gereksinimleri yerine getirmeye çalışan iki tip teleradyoloji uygulaması bulunmaktadır (Pianykh, 2008):

- Ağır teleradyoloji uygulamaları: Bölgesel bir PACS ağı oluşturmada kullanılabilir. İş istasyonları ve PACS arşivleri WAN (Wide Area Network) aracılığı ile birbirine bağlıdır. Bölgesel PACS ağları çoğunlukla hastaneler arası projelerde kullanılabilir.
- Hafif teleradyoloji uygulamaları: PACS arşivlerine, Web üzerinden veya mobil cihazlar üzerinden bağlantı kurulan uygulamalar bu kategorinin içine girmektedir. Web teknolojilerinden yararlanan uygulamalarda harici bir yazılıma gerek olmadığı için çoğunlukla bu yöntem tercih edilir.

Ulusal ve uluslararası radyoloji ağlarında kullanılan Bölgesel PACS uygulamaları gitgide daha yaygın hale gelmektedir (Pianykh, 2008). Coğrafi olarak çeşitli yerlere yayılmış bir hastanede bulunan PACS arşivlerinin birleştirilmesinde bu uygulamalar kullanılabilmektedir.

Hafif teleradyoloji uygulamaları, taşınabilir ve modüler çözümler sunmakta ve yaygın olarak kullanılmaktadır. İnternet tabanlı bir teleradyoloji uygulaması ile doktor, sağlık merkezi dışında olsa dahi İnternet'e bağlı bir bilgisayar kullanarak PACS arşivine bağlanabilmektedir. Çoğunlukla bağlantı bir İnternet tarayıcısı (Internet Explorer, Firefox, Opera gibi) ile sağlanmaktadır. Şekil 2.7'de bu tip teleradyoloji uygulamaları tasvir edilmiştir.



Şekil 2.7 İnternet Tarayıcısı Tabanlı Bir Teleradyoloji Uygulaması

Teleradyoloji uygulamaları yaygınlaşmaya başladığı sırada PACS geliştiren birçok yazılım firması, iş istasyonu uygulamalarını tarayıcı üzerinde çalışabilir hale getirmeye çalıştılar (Pianykh, 2008). Fakat bu uygulamalar tarayıcılar üzerinde çalışabilmek için birçok ek yazılıma (plug-in) ihtiyaç duymakta ve yavaş çalışmaktaydı. Ayrıca normal bir iş istasyonunda olduğu gibi erişim yapılandırmalarının (Uygulama birimi ismi – AE Title, IP adresi, bağlantı noktası) yapılması gerekmekteydi. Hala benzer sistemlerin olduğu günümüzde, son kullanıcılar rahatlıkla bu uygulamaları kullanamamakta ve birçok yapılandırma ve kurulumun içerisinde kaybolmaktadırlar. Fakat tarayıcı tabanlı bir teleradyoloji uygulamasının bu ve diğer bazı sorunları çözmesi gerekmektedir. Bu yüzden gerçek bir taşınabilir radyoloji sistemi kullanılan işletim sistemi, donanım, ağ ve yapılandırma ayarlarındaki farkları gözetmeksizin her ortamda çalışabilmelidir.

Kısacası İnternet tabanlı teleradyoloji uygulamaları geleneksel bir PACS uygulamasından farklı şekilde düşünülerek tasarlanmalıdır. Bu uygulamalar

İnternet teknolojileri ve PACS arşivlerinin sağladıkları tüm olanaklardan faydalanmalı ve bunlar arasında bir harmoni oluşturulmalıdır.

Ayrıca teleradyoloji uygulamalarının son kullanıcıların bu uygulamalardan bir fayda sağlayabilmeleri için sunması gereken özellikler bulunmaktadır. Bunlar (Pianykh, 2008):

- Gelişmiş görüntü sıkıştırma algoritmaları kullanmaları gereklidir. JPEG2000, JPEG-LS gibi sıkıştırma yöntemleri ile sitem düşük bant genişliğine sahip ağlarda kolayca çalışabilir.
- Daha esnek ve sağlam bir DICOM gerçekleştirimi sunmalıdırlar. Bir teleradyoloji uygulaması birçok PACS arşivi ile çalışabilmelidir. Bunu sağlamanın yolu DICOM gerçekleştiriminin standarda tam uyumlu olmasıdır.
- Verilerin güvenliği sağlanmalıdır. VPN gibi harici güvenlik yöntemleri her sağlık merkezinde bulunamayacağı için teleradyoloji uygulaması verileri şifreleyebilmelidir.
- İstemci ve sunucu arasındaki fark bilinmelidir. İstemci her bilgisayarda çalışabilmeli ve küçük boyutlu olmalıdır. Sunucu ise verileri tutan ve işlem yapan taraf olmalıdır.
- İstemci görüntü işleme özellikleri sunmalıdır. HTML (Hyper Text Markup Language)'in yanı sıra Ajax, Java ve ActiveX gibi teknolojilerde kullanılarak gerekli görüntü işleme özelliklerini (küçültme, büyütme gibi) bünyesinde barındırmalıdır.

Teleradyoloji uygulamaları 7/24 bir uzman bulunmayan küçük hastaneler, görüntüleme merkezleri gibi birçok sağlık merkezinde son derece hayati bir yere sahiptir. Ayrıca çok sayıda personele sahip büyük hastanelerde, her bir iş istasyonuna PACS uygulaması satın alınamayacağı için İnternet tarayıcısında çalışan bir iş istasyonu, birçok doktorun işini görebilecektir.

3. DICOM

DICOM, sayısal tıbbi görüntülerin depolanması, iletilmesi, görüntülenmesi gibi işlemlerin tanımlandığı standarttır. İçerisinde dosya formatı ve ağ iletişim protokolü tanımlanmıştır. İletişim protokolü TCP/IP protokolünün uygulama katmanı üzerine tasarlanmıştır.

DICOM, görüntüleme cihazı, görüntüleyici iş istasyonu, yazıcı, görüntü arşivi gibi PACS bileşenlerinin birbirleri ile nasıl iletişim kuracağını tanımlar. ACR (American College of Radiology) ve NEMA (National Electrical Manufacturers Association) tarafından geliştirilmeye devam edilen DICOM standardı şu an 3. sürümündedir.

DICOM ilk olarak 1983 yılında ACR ve NEMA tarafından kurulan ortak bir komite tarafından tasarlandı. Ortak komiteye ACR-NEMA sayısal görüntüleme ve iletişim standartları komitesi ismi verildi. Komite ilk olarak o zamana kadar oluşturulan standartları incelemeye başladı. Komitenin o zamanki en önemli hedefi sayısal tıbbi görüntüleme sistemini cihaz üreticilerinden bağımsız hale getirmekti.

Standardın ilk sürümü olan ACR-NEMA 1.0 1985 yılında yayınlandı ve RSNA'nın (Radiological Society of North America) yıllık toplantısında duyuruldu. Daha çok bir kılavuz olarak yayınlanan bu ilk sürümün kullanılması zorunlu değildi. İçerisinde birçok hata ve sorun barındıran bu ilk sürümün iyileştirilmesi gerekliliği çok geçmeden anlaşıldı ve ACR-NEMA standardı geliştirmesi için çalışma grubu (Working Group – WG) olarak isimlendirdiği alt komiteler oluşturdu. Bu grupların çalışmaları sonucunda 1988 yılında ACR-NEMA 2.0 isimli standardın ikinci sürümü ortaya çıktı. Hatalardan arındırılan ikinci sürümü ile standart birçok üretici tarafından kullanılmaya başlandı.

1980'lerin sonlarına doğru bilgisayar ağlarının yaygınlaşmasıyla tıbbi görüntülerin bilgisayarlar arasında aktarımı gündeme geldi. Başta bilgisayar ağlarının yaygınlaşması sebebiyle üçüncü sürümün geliştirilmesine başlandı. Bu sürümün prototipi 1992 yılında RSNA'da tanıtıldı. Ardından standardın ilk 9 bölümü hazırlanarak 1993 yılının Eylül ayında RSNA'da sunuldu. ACR-NEMA tarafından iyileştirilen ve güncel bir hale getirilen standarda DICOM 3.0 ismi verildi.

Günümüzde hala DICOM 3.0 veya DICOM olarak isimlendirilen standart DICOM çalışma grupları tarafından halen geliştirilmeye devam edilmektedir. Standart üzerinde yapılan değişiklikler PS3.X-YYYY şeklinde isimlendirilmektedir. Standart ismi içerisinde geçen "X" standardın alt bölümlerini "YYYY" ise yayınlanan yılı temsil etmektedir. Standardın şu anki en güncel sürümü 2008 yılında yayınlanmıştır.

Bölümün devamında DICOM standardının temel özellikleri ve iletişim protokolü tanıtılmıştır.

3.1. DICOM Dilbilgisi

Tıbbi veriler çok farklı şekillerde tutulabilir. Mesafe milimetre cinsinde, zaman saniye cinsinde ölçülebilir. Hasta isimleri genellikle alfabetik karakterlerle yazılır. Standardın PS3.5 bölümünde veri gösterimi (Value Representation - VR) olarak tanımlanan ve tüm olası tıbbi veri tiplerini kapsaması için tasarlanan 27 tane temel veri tipi tanımlamıştır. DICOM nesnesinin içerisinde varolan değerler, bu 27 ayrı veri tipinden biri ile tanımlanmalıdır. Her VR, 2 harften oluşan kısaltılmış bir isme, ne ifade ettiğine dair bir tanıma, veri uzunluğuna ve verinin içerebileceği karakterlerin neler olabileceğine dair bir açıklamaya sahiptir. DICOM'da oluşan birçok sorunun ve karışıklığın nedeni genel olarak VR temellidir.

Çizelge 3.1'de, DICOM VR'ları veri tiplerine ve karmaşıklığına göre sıralanmıştır.

Çizelge 3.1 DICOM Değer Gösterimleri (NEMA, 2008)

VR İsmi	**Tanım**	**Uzunluk**
Metin		
CS Code String	Karakter dizisi	En fazla 16
SH Short String	Kısa karakter dizisi	En fazla 16
LO Long String	Uzun karakter dizisi	En fazla 64
ST Short Text	Birden fazla paragraf içerebilen karakter dizisi	En fazla 1024
LT Long Text	ST'den daha uzun paragraflar tutabilen karakter dizisi	En fazla 10.240
UT Unlimited Text	Sınırsız denebilecek uzunluğa sahip karakter dizini	En fazla (2^{32} - 2)

Özel		
AE Application Entity	Bir cihaz ismini taşıyan karakter dizisi	En fazla 16
PN Person Name	Kişi ismi. İsim ayracı olarak "^" karakteri kullanılır	En fazla 64
UI Unique Identifier	Birçok şeyi tekil olarak birbirinden ayırmaya yarayan karakter dizisi	En fazla 64
Tarih ve Zaman		
DA Date	YYYYMMDD olarak biçimlendirimiş tarihi içeren karakter dizisi	8
TM Time	HHMMSS.FRAC biçiminde zamanı içeren karakter dizisi	En fazla 16
DT Date Time	Tarih ve zamanı tutan karakter dizisi Biçim : YYYYMMDDHHMMSS.FFFFFF	En fazla 26
AS Age String	nnnD, nnnW, nnnM, nnnY şeklinde yaşı tutan karakter dizisi (018M gibi)	4
Metin Biçimindeki Sayılar		
IS Integer String	Bir tamsayıyı tutan karakter dizisi	En fazla 12
DS Decimal String	Sabit ve kayan ondalıklı sayıları tutan karakter dizisi	En fazla 16
İkili Biçimdeki Sayılar		
SS Signed Short	16 bit uzunluğunda işaretli tamsayı	2
US Unsigned Short	16 bit uzunluğunda işaretsiz tamsayı	2
SL Signed Long	İşaretli tamsayı	4
UL Unsigned Long	32 bit uzunluğunda işaretsiz tamsayı	4
AT Attribute Tag	Sıralı 2, 2 byte uzunluğunda işaretsiz tamsayı tutan VR	1
FL Floating Point Single	Tek hassasiyetli kayan noktalı sayı	4
FD Floating Point Double	Çift hassasiyetli kayan noktalı sayı	8
OB Other Byte String	Her elemanı byte tutan karakter dizisi	
OW Other Word String	Her elemanı 2-byte'tan oluşan karakter dizisi	
OF Other Float String	32 bitlik kayan noktalı sayı içeren karakterler dizisi	
Diğer		
SQ Sequence of Items	Sıralı elemanlar	
UN Unknown	Kodlama yöntemi bilinmeyen byte'lar içeren karakter dizisi	

3.1.1. VR uzunluğu

VR'lar DICOM veri tiplerini tanımlar ve veri uzunluğu bu tanımın en önemli bölümüdür. DICOM tüm verilerin boyutlarını iki şekilde takip eder. İlk olarak DICOM her zaman veri boyutunu verinin değeri ile beraber saklar ve bu şekilde DICOM bir veri elemanının nereden başladığını ve nerede sonlandığını

bilir. İkincisi, bazı VR'lar için veri uzunlukları Çizelge 3.1'in son sütununda görüldüğü gibi sabit ya da sınırlıdır.

Farklı bilgisayar sistemleri, *Big Endian* ve *Little Endian* olarak temel nümerik veri tipleri (tamsayı, kayan noktalı sayılar vb.) için farklı byte sıralamaları kullanır. Bu sıralama biçimleri DICOM verilerini bilgisayar donanım ve yazılımındaki farklılıklardan korumaktadır.

Sabit uzunluğa sahip olsun ya da olmasın tüm DICOM veri elamanları çift sayıda uzunluğa sahip olmalıdırlar. Bu yüzden çift sayıda karakter (eğer eleman bir metin ise) ya da byte (eğer eleman ikili sayı ise) içerirler. Bunu sağlamak için, DICOM uzunluğu tek sayı olan metin dizisinin sonuna boş bir karakter ya da tek sayıda rakamdan oluşan ikili sayı dizisinin sonuna boş bir bayt ekleyerek dizilerin uzunluklarını çift sayı haline getirir. Örneğin, "Mustafa^Aydın" ismi DICOM'da tutulmak istendiğinde sonuna boş bir karakter eklenerek "Mustafa^Aydın " olarak saklanacak ve uzunluğu 14 olacaktır.

DICOM'da verilerin uzunluklarının çift olması gereksinimi, geçerliliklerini kontrol etmek için parite kontrol biti olarak ve tek sayıda uzunluğa sahip herhangi bir veriyi bozuk olarak belirtmek için kullanılabilir.

3.1.2. Özel karakterler

DICOM yerelleştirme amaçlı farklı dillerdeki özel karakterlerin kullanılmasına olanak sağlar. Dil seçimi ayrıca VR veri tiplerinde izin verilecek karakter ve harf seçimini de etkiler. DICOM bu karakter seçimini "karakter repertuvarı" olarak adlandırır.

DICOM standardını uyarlayan yazılımlar genellikle Latin alfabesini kullanır. Kullanılan "ISO (International Organization for Standardization) IR-6" karakter repertuvarı birçok dil için yeterlidir.

DICOM'da bazı karakterlere özel anlamlar yüklenmiştir. Yıldız işareti (*) "herhangi bir karakter dizisi" anlamına gelir; soru işareti (?) ise "herhangi tek bir karakteri" ifade eder; taksim işareti (\) "ya da" anlamını taşır. Örneğin, BT ya da MR çalışmaları aranmak istenirse, "BT ya da MR" anlamına gelen "BT\MR" karakter dizisi kullanılabilir. Bu özel karakterlerin birçok avantajı vardır. Örneğin eğer hastanın tam ismi bilinmiyorsa, tüm benzer eşleşmeleri getirmek için ilk

birkaç harften sonra yıldız işareti (*) kullanılabilir (“Ay*” sorgusu Aydın, Aysel, Aylin, Aynur gibi eşleşmeleri döndürür).

3.1.3. Metin dizileri

Metin VR’ları (CS, SH, LO, ST, LT ve UT sırasıyla kod dizisi, kısa dizi, uzun dizi, kısa metin, uzun metin ve sınırsız metin) metin dizileri saklamak için kullanılırlar. Metin VR’ları için boyut sınırlandırmaları çok önemlidir. Örneğin, CS için 16 karakterden sonra gelen tek bir karakter küçük bir yazılım hatasına ya da büyük bir PACS problemine sebep olabilir.

3.1.4. Tarih ve zaman verileri

DA (Tarih), TM (Zaman) ve DT (Tarih-Zaman) tipleri, tarih ve zaman verilerini tutmak için kullanılır. Bu tip veriler için en önemli şey, tarih ve zamanın doğru sırada yazılması gerekliliğidir.

3.1.5. Metin biçimindeki sayılar

IS (tam sayı dizisi) ve DS (ondalık sayı dizisi) veri tipleri, sayısal değerleri (tam sayı ve kayan ondalık sayı) metin dizisi olarak tutar. Her ne kadar ikili sayı sistemi ile bilgisayarda saklansa da, DICOM birçok yerde sayıları metin biçiminde tutar. Metin biçimli sayılar, sistemler arası Little Endian/Big Endian farklılıklarına bağlı değildir. Ayrıca yazılımsal olarak okunmaları daha basittir.

3.1.6. Sayılar

SS, US, SL, UL, FL ve FD (işaretli kısa, işaretsiz kısa, işaretli uzun, işaretsiz uzun, tek kayan noktalı, çift kayan noktalı) tek bir sayıyı göstermek için kullanılır.

OB, OW ve OF (diğer byte dizisi, diğer kelime dizisi, diğer kayan noktalı dizi) ise uzun sayısal diziler için kullanılır. Örneğin, bir dijital görüntüyü tanımlayan piksel dizisinin bu veri tiplerinde saklanması gerektiğinde, dizideki her sayı aynı byte uzunluğuna sahip olacak (1,2 ve 4 byte) ve görüntüyü temsil eden bir ikili sayı dizisini oluşturacaktır. Sadece OB tipi, 1 bayt uzunluğundaki sayıları tutar. Diğerleri 1 bayttan daha uzundurlar ve bu yüzden Little Endian/Big Endian byte sıralamasından etkilenirler.

Son olarak, AT (özellik etiketi) bir çift 2 byte'lık sayıyı depolar. AT tipi DICOM veri özelliklerini numaralandırmak için kullanılır.

3.1.7. İsimler

Kişi ismi (Personel Name - PN) VR'ı kişi ismini tutar. DICOM bu değeri tutmak için adı soyadı olarak iki farklı alan yerine tek bir alan kullanır. "Mustafa Aydın" ismi "Mustafa Aydın", "Aydın^Mustafa" ya da "Aydın,Mustafa" olarak yazıldığında yazılımda ve klinik iş akışında sıklıkla karışıklığa sebep olur. Bu belirsizliği ortadan kaldırmak için, DICOM aşağıdaki isim sırasını tavsiye eder:

Soyadı^Adı^Göbek Adı^İsim Ön Ek^İsim Son Ek

Hepsi (^) ile ayrılmıştır. Çalışma ortamında, bu sıra sürekli olarak kayıp bilgiler ya da yanlış tanımlanan hastalar yüzünden sıklıkla değiştirilir.

Bu problemin iki tane çözüm yolu vardır:

1. Hastayı belirlemek için her zaman hasta ismi yerine hasta kimlik numarasının kullanılması.
2. Hasta ismini ararken herhangi bir metin anlamına gelen yıldız işareti (*) gibi özel sembollerin kullanılması. "*Aydın*" sorgusu ile isminin herhangi bir yerinde "Aydın" bulunan hastalar bulunabilir. Böylece, hasta bilgileri kaybolmayacak ya da isim sırasından etkilenilmeyecektir. Birçok yazılım arama kriterinin sonuna otomatik olarak yıldız işaretini eklemektedir.

3.1.8. Uygulama birimleri

Uygulama birimi (Application Entity - AE) VR'ı bir DICOM uygulama birimini gösterir. AE esasen bir DICOM cihazını ya da programını tanımlayan tekil bir isimdir. Yani aynı PACS ağında aynı AE'ye sahip iki uygulama bulunamaz. AE'ler herhangi bir DICOM ağı ya da PACS için en önemli VR'dır.

AE'ler sıklıkla, sayılarla ve sadece büyük harfli karakterle etiketlenir. Boşluk, noktalama işaretleri ya da diğer karakterler kullanılmaz. Büyük ve küçük harfe duyarlı isimlerin kullanılmasından kaçınılmalıdır. Çoğu DICOM uygulaması büyük-küçük harf duyarlı olmasa da bazı uygulamalar duyarlı olabilmektedir.

3.1.9. Tekil tanımlayıcılar

UID VR'ları DICOM nesnelerini tanımlamak için kullanılan tekil tanımlayıcılardır. AE'lerin yerel olarak benzersiz olması beklenirken (örneğin kendi ağının içerisinde), DICOM UID'leri evrensel olarak benzersiz olmalıdır. Örneğin, bir DICOM biriminden diğerine bir çalışma kopyalanırken, ikinci birim aynı çalışmanın örneği ile iş yaptığını belirtmek için çalışmanın UID özelliğinin değerini değiştirmelidir. Bu şekilde iki çalışmanın birbirinden farklı olduğu belirtilerek, ikinci çalışma örneği üzerindeki herhangi bir değişikliğin (netlik, parlaklık ayarı gibi) asıl çalışmada hiçbir değişikliğe sebep olmaması sağlanır.

3.1.10. Sıralı veri kümesi

SQ VR, her bir verinin birden çok veri tipine sahip olabildiği bir veri kümesi içerir. Böylece bir VR'ın diğer bir VR'ın içerisine konmasına (iç içe VR) izin verilerek çok karmaşık DICOM veri yapıları oluşturulmasına olanak sağlanır. Benzer elemanların tek bir veri bloğu içerisinde gruplanmasında da kullanılır.

DICOM standardında iç içe konulan veriler büyüktür işareti (>) ile gösterilmiştir. Örneğin "Referenced Series Sequence" isimli özelliğin içerisinde "Referenced Instance Sequence" özelliği, bu özelliğin içerisinde ise "Referenced SOP Class UID" özelliği bulunmaktadır. Standardın içerisinde bu yapı şu şekilde gösterilir:

Referenced Instance Sequence
>...
>Referenced Instance Sequence
>>Referenced SOP Class UID
>>...

3.1.11. Bilinmeyen değerlerin gösterilmesi

UN VR'ı diğer 26 VR'a uymayan yani bilinmeyen değerleri tanımlamak için kullanılır.

Sonuç olarak, VR'lar DICOM'u dış dünyaya bağlayarak DICOM veri yapısında en önemli rolü oynarlar. VR'lar DICOM'un anlayabildiği ve

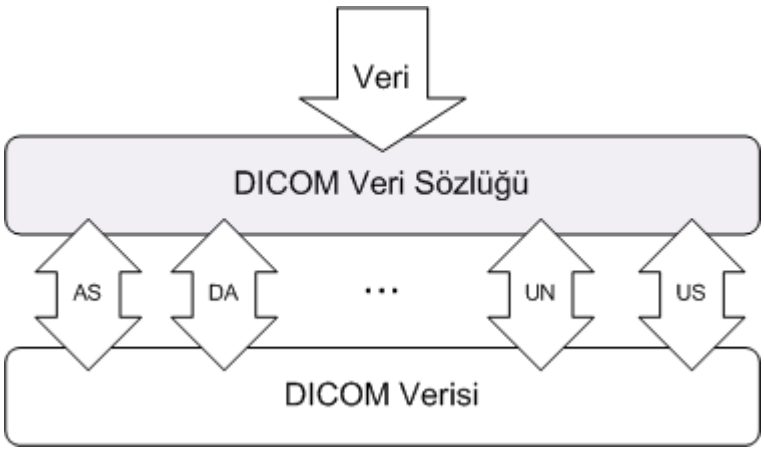
konuşabildiği kelime şekilleridir. Gerçek verileri VR'a çevirmenin yolu ise DICOM veri sözlüğüdür.

3.2. DICOM Veri Sözlüğü

DICOM standardının PS3.6 bölümü tüm standart DICOM özelliklerini içeren DICOM veri sözlüğünü içerir. Standart sözlüğe ek olarak, DICOM uygulama üreticileri kendi yarattığı özellikleri içeren sözlüklerini üretebilirler. Her iki durumda da, sözlükler aynı kurallara uymak zorundadırlar. Bu bölümde bu kurallar üzerinde durulacaktır.

3.2.1. Standart DICOM veri sözlüğü

DICOM veri sözlüğü sayısal sağlık biliminde kullanılan standart özellikler için bir kayıt defteridir. Bu özellikler 27 tane VR'a göre biçimlendirilmelidirler. Şekil 3.1'de verilerin DICOM ortamına nasıl çevirildiği gösterilmektedir



Şekil 3.1 Verilerin DICOM Biçimine Çevrilmesi (Pianykh, 2008)

Her eleman etiket (tag) olarak isimlendirilen "(grup, eleman)" çifti ile numaralandırılır. Etiketlenmiş elemanlar, özellik (attribute) ya da DICOM veri elemanı (data element) olarak isimlendirilir.

Gruplar ve elemanlar onaltılık sayı sistemi ile numaralandırılır. Çizelge 3.2'de DICOM veri sözlüğünden alınan bir kesit görülebilir. İlk sütun onaltılık "(grup,eleman)" etiketini içerir. Özellik ismi sütunu, elemanda hangi verinin saklanacağını belirtir. Yani "(grup,eleman)" etiketleri özellik isimlerine karşılık gelir.

Çizelge 3.2 DICOM Veri Sözlüğünden Bir Kesit

Etiket	**Özellik İsmi**	**VR**
(0008, 0001)	Length to End	
(0008, 0005)	Specific Character Set	CS
...		
(0010, 0010)	Patient Name	PN
(0010, 0020)	Patient ID	LO
(0010, 0021)	Issuer of Patient ID	LO
(0010, 0030)	Patient's Birth Date	DA
(0010, 0032)	Patient's Birth Time	TM
(0010, 0040)	Patient's Sex	CS
...		
(0010, 1000)	Other Patient IDs	LO
(0010, 1001)	Other Patient Names	PN
...		
(FFFE, E00D)	Item Delimitation Item	
(FFFE, E0DD)	Sequence Delimitation Item	

Sözlükteki VR sütunu her bir veri elemanının biçimini belirler. Örneğin, "Patient's Birth Date" özelliği (0010, 1001) DA biçiminde olmalıdır (Yani 8 haneli, YYYYAAGG veri dizisi).

Değer çokluğu (Value Multiplicity - VM) sütunu kendi tipinde sadece bir ya da birkaç değer içerebileceğini tanımlar. Örneğin, "Other Patient Names" elemanı birden fazla isim içerebilir. Bu yüzden elemanın çokluğu "1-n" olarak işaretlenir. Metin değerleri genellikle sabit uzunluğa sahip değildir. Bu yüzden ayraç olarak taksim işareti (\) kullanılarak birleştirilir. Örneğin, Mustafa Aydın isimli hasta "Dr. Mustafa" ve "Doç.Dr.Aydın" gibi isimleri kullanmayı tercih etseydi, "Other Patient Names" özelliği çokluk değeri 2 olan "Dr.^Mustafa\Doç.Dr.^Aydın" içerebilirdi. DICOM'da taksim işareti (\) birden fazla değeri olabilen özelliklerde "veya" (or) anlamına gelen, çok özel bir anlama sahiptir. Bu sebepten dolayı, başka amaçlar için (dosya isimleri, tarih gibi) kullanılmamalıdır.

DICOM veri sözlüğündeki "RET" etiketli özellikler tedavülden kalkmış özelikleri gösterir. Bu özellikler, DICOM veri sözlüğünde italik olarak yazılmıştır.

Çizelge 3.2'de verilen VR ve DICOM veri sözlüğü ile ufak bir DICOM nesnesi örneği verilebilir. Örneğin:

"Mustafa Aydın isimli Erkek hasta 6 Ağustos 1954 yılında doğmuştur"

Bu cümlede üç tane veri elemanı bulunmaktadır: “Patient Name” (010, 0010), “Patient’s Sex” (0010, 0040) ve “Patient Birth Date” (0010, 0030). Bu elemanlar sırasıyla PN, CS ve DA tipindedirler. Bu nedenle, isimler etiketlerle yer değiştirilip, VR biçimlendirme uygulanırsa:

(0010,0010)Aydın^Mustafa(0010,0030)19540806 (0010,0040)M

sonucu elde edilir.

3.2.2. Özel DICOM veri sözlükleri

Standart DICOM veri sözlüğü tıbbi görüntüleme endüstrisinden dikkatlice derlenmiş 2000’den fazla özellik içermektedir.

Çeşitli DICOM ve PACS üreticileri kodlanmış DICOM verisinin içerisine kendilerine ait DICOM özelliklerini eklerler. DICOM veri sözlüğündeki tüm çift sayılı grup numaraları genel kullanım, tek sayılı grup numaraları ise özel kullanım için ayrılır. Örneğin 0009 grup numarası yazılımı geliştiren kişi tarafından özel elemanlar için ayrılabilir

3.2.3. Standart DICOM komut sözlüğü

PRINT, STORE, MOVE, GET gibi DICOM komutları tıbbi görüntüleme iş akışında kullanılır, ama veri sözlüğünün hiçbir yerinde bulunmamaktadırlar.

DICOM ayrılmış 0000 grubu içerisinde komutları veri elemanları ile aynı biçimde tanımlamıştır. Örneğin, “(0000, 0100)” elemanı çoğunlukla komut tipi (Command Type) özelliğini ve “(0000, 0110)” komut mesaj numarası (Command Message ID) özelliğini göstermek için kullanılır. DICOM standardı, komut mesaj nesnelerinin tanıtıldığı PS3.7 bölümünde, detaylı bir şekilde, farklı komut mesajlarının içeriklerini ve kullanımlarını açıklar.

DICOM komutları, veriler gibi özelleştirilebilir grupları desteklememektedir. DICOM komut kümesi sınırlı olan ve genellikle yerel ağda kullanılan PACS mimarisi için tasarlanmıştır. Fakat teleradyoloji gibi modern dijital görüntüleme projeleri, standartların sunduğundan daha esnek bir DICOM komut yapısı gerektirmektedir.

3.3. DICOM Nesneleri

DICOM nesneleri standardın en önemli parçasıdır. DICOM nesneleri aslında DICOM bilgilerini ve komutlarını saklayan, taşıyan ve kodlayan gerçek kelime ve cümlelerdir.

Tüm DICOM verileri (tıbbi görüntüler, komutlar ve raporlar) DICOM veri biçimine dönüştürülür. Veriler, bu biçimde bir DICOM ağı üzerinde çeşitli DICOM cihazları arasında yol alır ve böylece DICOM dosyalarının iletimi sağlanır.

Bir DICOM nesnesi aslında bir veri elemanı yığınıdır. Örneğin bir tıbbi görüntü genişlik, uzunluk, renkler ve görüntünün elde edildiği tarih gibi özelliklere sahip olur. Tüm bu özellikler etiket ve değerleri ile birlikte DICOM nesnesine çevrilir. Yani görüntüye ait tüm bilgileri içeren bu elemanlar dizisi aslında bir DICOM nesnesidir.

DICOM nesneleri, daha önce bahsedildiği gibi, basit elemanların yanı sıra SQ VR gibi karmaşık elemanları da içerebilir. Diğer bir değişle DICOM nesneleri başka DICOM nesnelerini de içerebilir. Bu öz yineleme ya da iç içe geçmiş DICOM nesneleri, eğer DICOM nesneleri bir veri ağacı olarak düşünülürse, çok karmaşık bir ağaç yapısı yaratır. Bu veri ağacında, dallar DICOM nesnelerini yapraklar ise veri elemanlarını gösterir.

3.3.1. Veri elemanlarını kodlamak

DICOM’da kodlamanın anlamı, özellik verilerini DICOM’a özgü biçimde yazmak ve karmaşık DICOM özellik değerlerini bir bayt dizisine dönüştürmektir. DICOM standardının PS3.5 bölümünde iki tane önemli kodlama tipi tanımlanır: kapalı (implicit) VR kodlaması ve açık (explicit) VR kodlaması. Kapalı kodlama basittir ve DICOM standardında varsayılan VR kodlama tipi olarak kullanılır. Uygulamalar DICOM nesneleri üzerinde kullanılacak kodlama tipini müzakere edebilirler.

3.3.2. Veri gruplarını kodlama

DICOM veri sözlüğündeki elemanlar, gruplara bölünür ve “(grup, eleman)” çifti ile etiketlenir. Örneğin, veri sözlüğündeki 0010 grubu hasta ile ilgili tüm elemanları (isim, ID, ağırlık, beden, yaş vb.) bir arada toplar.

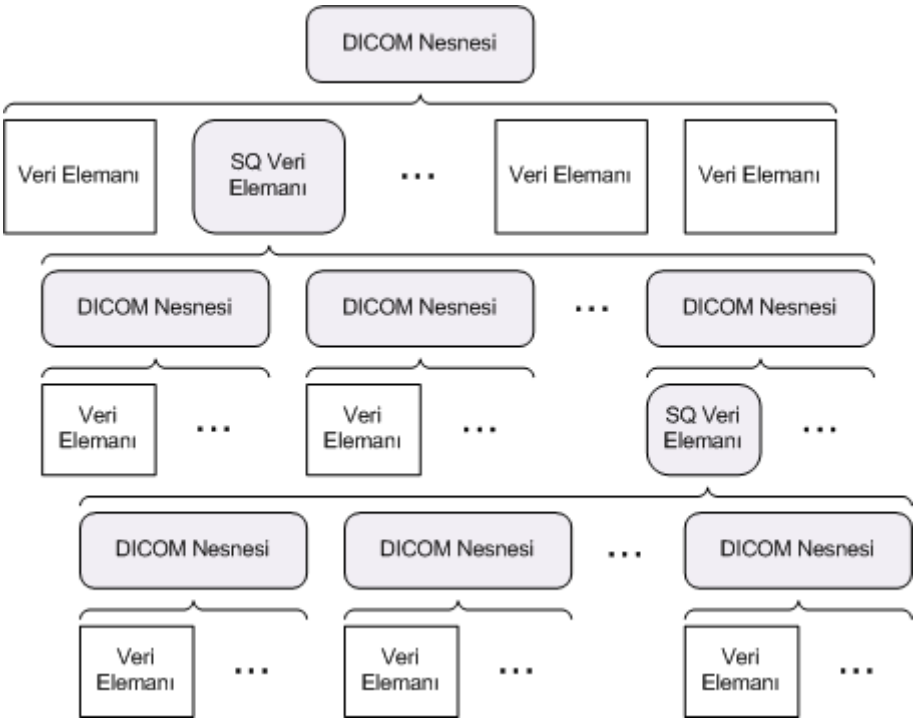
DICOM veri elemanları, DICOM nesneleri olarak kodlanırken “(grup, eleman)” etiketlerinin sırasıyla en küçükten en büyüğe doğru yazılır. Böylece, elemanlar, her eleman grubunun içerisinde yükselen sırada sıralanır. Ayrıca gruplar da kendi aralarında yükselen sırada sıralanırlar. Örneğin, “(0008, 0012)” elemanı “(0008, 0014)” elemanından önce gelir. “(0010, 0010)” elemanı ise daha yüksek grup numarasına sahip olduğu için diğer ikisinde sonra gelir.

Her bir DICOM “gggg” grubu için, “(gggg, 0000)” ilk elemanı verilen gggg grubundaki veri elemanlarının uzunluğu olan L’yi tutar. Bir DICOM nesnesi yazıldığında, sonraki grubun başlangıcına kadar olan gggg grubunun tüm elemanlarının uzunluğunu kaydetmek için, gggg grubunun başlangıcında UL VR biçiminde “(gggg, 0000)” elemanı yazılır.

3.3.3. DICOM nesne dizilerini kodlamak

SQ VR, daha önce anlatıldığı gibi, bir DICOM nesnesini içerisinde tutabilir. SQ elemanı bir DICOM nesne dizisi içerir. SQ elemanının içerisinde bulunan DICOM nesnesi, başka bir SQ elemanını yani bir DICOM nesnesini de içerebilir. Böylece iç içe DICOM nesneleri tek bir nesnenin içerisinde yer alabilir. Şekil 3.2’de SQ elemanları ile bir DICOM nesnesinin nasıl oluşabileceği verilmiştir.

İçerisinde SQ elemanı olmayan bir DICOM nesnesi, veri elemanlarının bir listesidir. Bu yüzden bir liste olarak kodlanabilir. SQ elemanları için, DICOM tek bir VR içerisinde DICOM nesne dizisini kapsayan özel bir SQ kodlama şeması sağlar. SQ elemanlarının nesne içerisine nasıl kodlandığı DICOM standardının PS3.5 bölümünden detaylı olarak incelenebilir.



Şekil 3.2 SQ Elemanları ile İç İçe DICOM Nesneleri (Pianykh, 2008)

3.3.4. Görüntü verisinin saklanması

Görüntüler, diğer elemanlar için geçerli veri kodlama kurallarını takip ederler. Bu yüzden kolayca DICOM nesnelerinin içerisine koyulabilirler. Her bir tıbbi görüntü, standart DICOM veri sözlüğündeki özelliklere benzeyen birkaç elemana sahiptir. Bunlardan birkaçı:

1. Görüntü Uzunluğu : (0028, 0010)
2. Görüntü Genişliği: (0028, 0011)
3. Piksel Veri: (07E0, 0010)

Piksel veri elemanı görüntünün içeriğini saklar ve çoğu durumda bir DICOM nesnesinin boyutunun yaklaşık %95'ini oluşturur.

Ayrıca, sayısal videoyu (anjio gibi) oluşturan görüntü kareleri dizisi tek bir DICOM nesnesine yerleştirilebilir.

3.3.5. Tekil tanımlayıcılar

DICOM nesnelerini birbirlerinden ayırt edebilmek için tekil tanımlayıcılar (Unique Identifier – UID) kullanılır.

Görüntü örnekleri başka arşivlerde birçok diğer görüntü ile saklanması için PACS arşivlerinin dışına gönderilebilir. DICOM UID'leri, insan DNA'sı gibidir. Bir nesneyi tanımlayabilmek için kullanılır. Benzer sebeplerden dolayı UID'ler DICOM'da sadece görüntüleri etiketlemek için kullanılmaz. Görüntü dizileri, çalışmalar, cihazlar, söz dizimleri ve diğer birçok yerde de kullanılırlar.

DICOM UID'leri 1.2.840.10008.1.2 gibi noktalarla ayrılmış sayısal değerlerden oluşurlar ve DICOM nesneleri içerisinde UI VR tipi ile tutulmaktadırlar. UI dizileri birçok ülke, yer, satıcı ve donanım arasındaki ayrımı sağlayabilmek için global olarak tekil olmalıdır. Bu yüzden, DICOM aşağıdaki UID kodlamasını kullanır:

UID = <org root>.<suffix>

Burada, UID'in "<org root>" kısmı tekil olarak bir organizasyonu (örneğin, üreticiler, araştırma örgütü, NEMA vb.) tanımlar. Ayrıca, tüm DICOM işlem UID'leri için genel "<org root>" olarak "1.2.840.10008" dizisi ayrılmıştır ve başka bir yerde kullanılamaz.

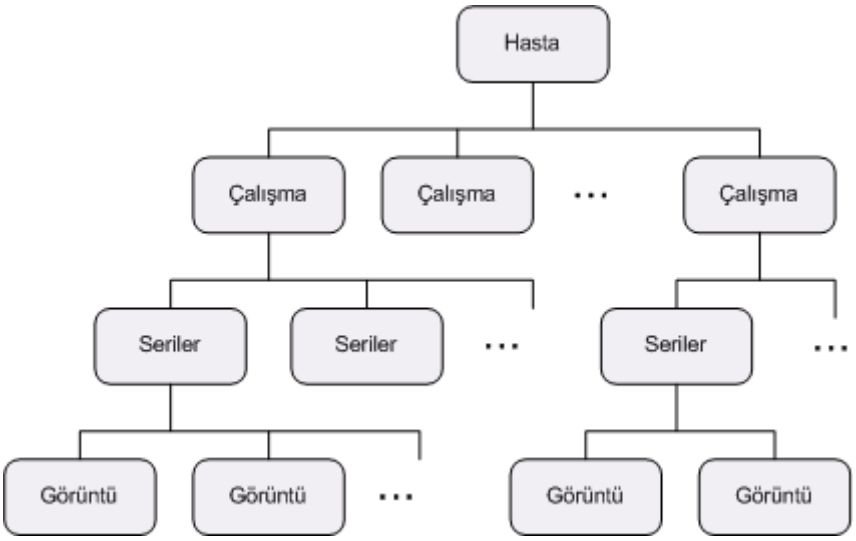
UID'nin "<suffix>" kısmı birkaç nümerik bileşenden oluşur ve "<org root>" içerisinde tekil olmalıdır. Kökler (roots) oldukça kısayken, sonekler (suffix) oldukça uzun olabilir. Herhangi iki görüntünün aynı UID'e sahip olmaması için UID'ler 64 karakterden oluşabilir. Aynı sebepten ötürü birçok uygulama, UID'leri DICOM dosya isimleri olarakta kullanılır.

3.4. DICOM Bilgi Hiyerarşisi

DICOM'da bilgiler Hasta-Çalışma-Seriler-Görüntü (Patient-Study-Series-Image) hiyerarşisi ile tutulmaktadır. Bu hiyerarşi içerisinde:

1. Bir hasta birçok çalışmaya sahip olabilir.
2. Her çalışma bir ya da daha fazla görüntü serisine sahip olabilir.
3. Her bir seri bir ya da daha fazla görüntüye sahip olabilir.

DICOM'daki bilgi hiyerarşisi Şekil 3.3'de görülebilir. Bu hiyerarşi hastaların görüntülerinin tıbbi ortamda nasıl tutulduğuna da ışık tutmaktadır.



Şekil 3.3 DICOM Bilgi Hiyerarşisi

Bu hiyerarşiyi gerçekleştirmek için, DICOM her hiyerarşi seviyesine bir anahtar ID'si atar. Hasta seviyesi için, bu "Patient ID" elemanıdır. "Patient ID" elemanı "(010, 0020)" etiketi ile saklanır ve herhangi bir görüntü için bu etiketin kullanılması zorunludur. Aynı ilke Hasta-Çalışma-Seriler-Görüntü hiyerarşinin diğer üç seviyesine de uygulanır: inceleme seviyesinde, her bir inceleme tekil "Study Instance UID"ine (0020, 000D) sahiptir. Gruplar seviyesinde her bir grup tekil "Series Instance UID"ine (0020, 000E) sahiptir. Görüntü seviyesinde her bir görüntü kendi tekil "SOP Instance UID"ine (0008, 0018) sahiptir. Hasta, çalışma, seriler ve görüntü UID'leri rakamlar ve noktalar ile oluşturulmuş 64 karakter uzunluğundaki tekil tanımlayıcı olarak tanımlanan, UI VR tipi ile tanımlanır. Patient ID elemanı, grafik arayüzlerde sıklıkla kullanılmasından dolayı diğer tanımlayıcılara nazaran daha az sayıda karakterden oluşur.

Tüm DICOM komutları ve çoğu DICOM veri özellikleri, her zaman dört seviyeli bilgi modeli ile sınırlıdır. Eğer iki hasta aynı hasta UID'ine sahipse, aynı kişi olmaları beklenir. Bu kural diğer seviyeler için de geçerlidir.

3.5. Modüller, IOD'lar ve Bilgi Birimleri

Veri elemanlarından DICOM nesneleri oluşturulduğuna, daha önceki bölümlerde değinilmiştir. Ama DICOM sözlüğündeki 2000 veri elemanıyla bu durum karmaşık bir hal alabilir. Normalde MR görüntüsüne, CT görüntüsüne ait birkaç etiketi eklenip DICOM nesnesi oluşturulamaz. Oluşturulsa dahi çoğu DICOM cihazı tarafından bu nesne reddedilir.

Veri elemanları daha küçük bir parçaya ayrılamayan, en küçük yapı öbekleridir. Çok sayıda bulunan bu veri elemanları daha geniş yapı öbekleri şeklinde gruplanmalıdır. Böylece daha anlamlı bir hale gelirler.

DICOM bu grupları bilgi modülleri, bilgi elemanları (Information Entities - IEs) ve bilgi nesne tanımı (Information Object Definition – IOD) olarak tanımlar. DICOM'daki çoğu şey gibi onlar da hiyerarşik olarak ilişkilidir. Modüller, IE'leri; IE'ler ise IOD'ları oluşturur.

3.5.1. Makroların özellikleri

Makrolar sıklıkla, farklı modüllerdeki benzer özellikleri tek bir kaynak göstererek kısaltmak için gruplar içerisinde birleştirilirler. Makrolar diğer hiçbir özel nesneye ya da veri öbeğine benzemez; DICOM standardı içerisinde tekrar eden elemanlardan kaçınılmasına yardım eder.

3.5.2. Bilgi modülleri

Modüller, veri eleman organizasyonunun en temel ve ilk seviyesini tanımlar. Örneğin, Çizelge 3.3'te gösterilen hasta tanımlama modülü (Patient Identification Module) tüm hasta tanımlama bilgilerini gruplar.

Çizelge 3.3 Hasta Tanımlama Modülü (NEMA, 2008)

Özellik	**Etiket**	**Tanım**
Patient Name	(0010,0010)	Hastanın ismi
Patient ID	(0010,0020)	Hasta tekil tanımlayıcısı
Other Patient Ids	(0010,1000)	Hastayı tanımlamak için kullanılan diğer tanımlayıcı numaralar
Other Patient Names	(0010,1001)	Hastayı tanımlayabilmek için kullanılan diğer isimler
Patient's Birth Name	(0010,1005)	Hastanın kızlık soyadı
Patient's Mother's Birth Name	(0010,1060)	Hastanın annesinin kızlık soyadı
Medical Record Locator	(0010,1090)	Hastanın tıbbi kaydına erişebilmek için kullanılan tanımlayıcı

Modüllerin asıl amacı, yapılı ve tutarlı bir şekilde ilişkili veri özelliklerini bir arada toplamaktır.

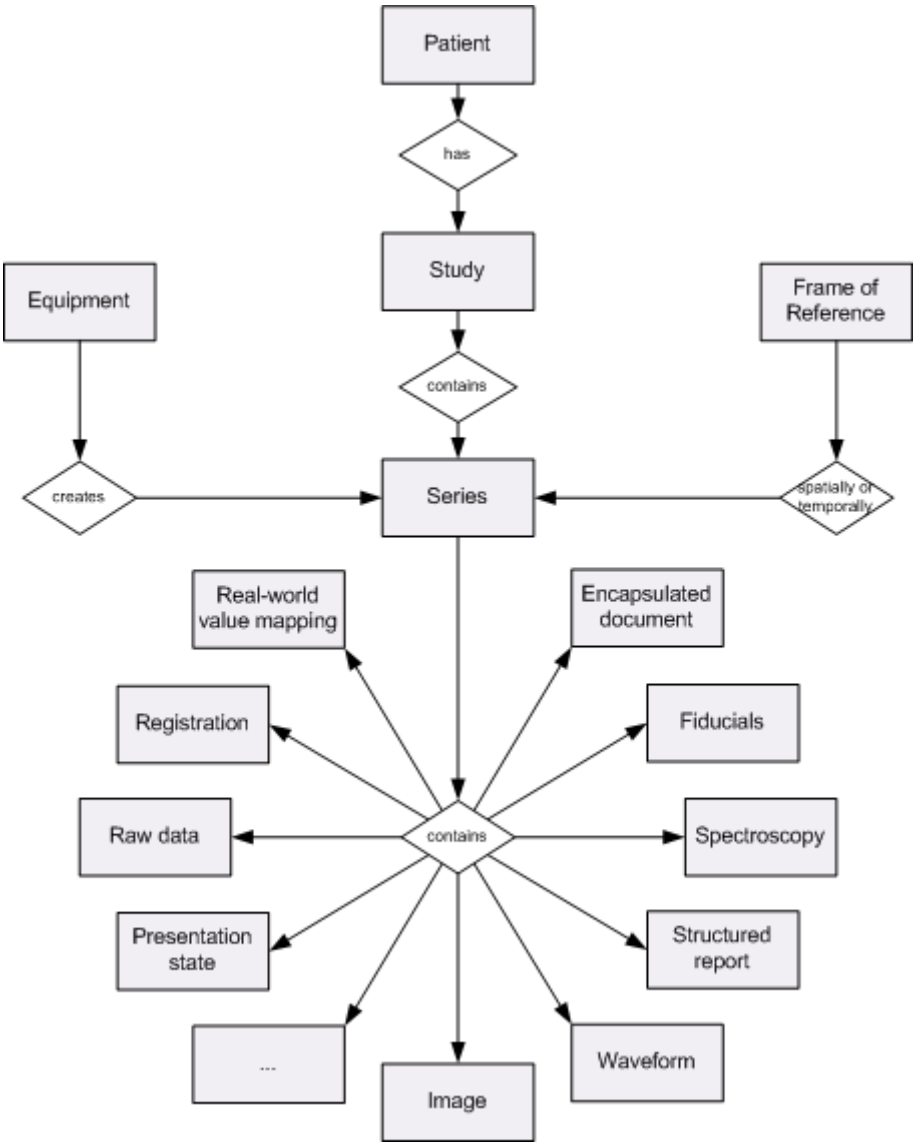
DICOM'da neredeyse her şey gibi, bilgi modülleri de zorunlu (standartta M harfi ile gösterilir), isteğe bağlı (C harfi ile gösterilir) ve kullanıcı tanımlı (DICOM'da U harfi ile gösterilir) olabilir. Uygun modül seçimi DICOM nesnesinde saklanan veri tipine bağlıdır.

Ayrıca DICOM modülleri normalize edilmemiştir. Yani, farklı modüller aynı özellikleri paylaşabilir. Örneğin, "Patient Name" gibi önemli özellikler birkaç DICOM modülünde birden bulunabilir. DICOM standardının 3. bölümü tüm standart modüllerin tanımlamalarını detaylı bir şekilde içermektedir.

3.5.3. Bilgi birimleri

DICOM bilgi birimleri (IE), DICOM bilgi modüllerinden oluşturulmuştur. DICOM; her bir IE için, o IE içerisinde kullanılması gereken birkaç modülü belirtir. Örneğin, ortak hasta IE'si (Common Patient IE) hasta modülünü (Patient Module), numune tanımlama modülü (Specimen Identification Module) ve klinik araştırma konu modülünü (Clinical Trial Subject Module) içerir.

IE'ler, DICOM bilgi modelindeki karmaşıklığın bir sonraki seviyesini gösterir. Modüllerin ilişkili veri elemanlarını birleştirdiği düşünülürse, IE'ler tıbbi görüntüleme iş akışındaki birimleri tanımlamak üzere tasarlanmışlardır. Şekil 3.4'deki kutular IE'leri temsil etmektedir.



Şekil 3.4 DICOM Bilgi Modeli (Pianykh, 2008)

DICOM standardında toplam 20 IE bulunmaktadır. Standardın 3. bölümünde IE'ler detaylı olarak anlatılmaktadır.

3.5.4. DICOM bilgi nesneleri

IE'ler anlamlı bir şekilde birleştirildiklerinde bilgi nesne tanımlarını (IOD) oluşturur. IOD'lar gerçek dünya nesnelerinin soyut gösterimleridir.

IOD'ler, DICOM bilgi hiyerarşinin en üstünde yer alır. Yani DICOM tarafından kullanılan nesneleri tanımlarlar. DICOM'daki veri işlemeleri IOD'lar bazında gerçekleştirilir. Çizelge 3.4'de örnek bir BT bilgi nesnesinin tanımlaması verilmiştir.

Çizelge 3.4 IE ve Bilgi Modülleri Tarafından Oluşturulan BT IOD (NEMA, 2008)

IE	Bilgi Modülü	Kullanım
Patient	Patient	M
	Clinical Trial Subject	U
Study	General Study	M
	Patient Study	U
	Clinical Trial Study	U
Series	General Series	M
	Clinical Trial Series	U
Frame of Reference	Frame of Reference	M
Equipment	General Equipment	M
Image	General Image	M
	Image Plane	M
	Image Pixel	M
	Contrast/Bolus	C
	Device	U
	CT Image	M
	Overlay Plane	U
	VOI LUT	U
	SOP Common	M

Hasta, MR görüntüsü, DICOM yazıcısı gibi gerçek dünyada var olan birimlerin IOD'ları, o birimin DICOM içerisinde kullanılacak sınıfını tanımlayan bir dizi özellikten ibarettir.

Ayrıca DICOM standardı, IOD'ları normalleştirilmiş ve birleşik olarak ikiye ayırır. Normalleştirilmiş bir IOD, Hasta IOD'u gibi tek bir gerçek dünya birimini ifade eder. Birleşik IOD ise; CT IOD'u gibi farklı farklı özellikleri bünyesinde barındıran IOD'ları tanımlar.

3.6. DICOM Hizmetleri

DICOM AE'leri, birbirlerine hizmet sağlarlar. Bir DICOM birimi bir diğer DICOM biriminden hizmet talep edebilir ve ona hizmet sunabilir. DICOM dilinde hizmet talebinde bulunan uygulama birimleri (Application Entity – AE) hizmet sınıfı kullanıcısı (Service Class User – SCU) ve hizmet sağlayan AE'ler ise hizmet sınıfı sağlayıcı (Service Class Provider – SCP) olarak tanımlanır. Başka bir deyişle AE'ler birbirleriyle iletişim kurmak için SCP ve SCU uygulama rollerini üstlenirler.

SCU ve SCP isimlerinden de anlaşılacağı gibi, tüm DICOM hizmetleri DICOM hizmet sınıfları seviyesinde gerçekleştirilir. DICOM hizmet sınıfları, veri

işleme fonksiyonlarıyla DICOM verilerini birbirine bağlar. Daha kesin bir deyişle, DICOM hizmet sınıfı bir veya daha fazla DICOM IOD'u bir veya daha fazla komut ile birbirine bağlar. Örnek olarak, DICOM Yazdırma Yönetimi Hizmet Sınıfı değişik resimleri (BT veya MR resimleri gibi IOD'ları) basmakla (komut) sorumludur. Sonuç olarak herhangi bir DICOM yazıcısı bu hizmeti sağlayabilir. Yani Yazdırma Yönetimi SCP'si (Print Management SCP) gibi davranabilir. Yazıcıya resim gönderen herhangi bir DICOM aygıtı bu hizmeti talep eder. Yani, Yazdırma Yönetimi SCU'su gibi davranır. Bölümün devamında bu ve benzeri işlemler anlatılarak DICOM hizmetleri anlatılacaktır.

3.6.1. DIMSE hizmetleri

DICOM AE'leri birbirlerinin sunduğu hizmetleri kullanabilmek için hizmet bilgisi sağlayan ya da talep eden hizmet mesajları gönderirler. Bu nedenle tüm DICOM Hizmet Komutları, DICOM mesaj hizmet elemanları (DICOM Message Service Elements – DIMSE) olarak bilinirler. DIMSE protokolü DICOM hizmet alışverişi için kurallar belirler. DICOM ağının bel kemiği niteliğindedir. Her DIMSE hizmetinin istek ve yanıt ileti bileşenleri vardır. İstekler SCU AE'leri (örneğin Arşive yeni bir resim eklemeye çalışan BT aygıtı) tarafından gönderilirler ve yanıtlar SCP AE'leri (örneğin bir BT arşivi) tarafından sağlanır.

Hizmet özellikleriyle veri özelliklerini birbirinden ayırmak için DICOM veri sözlüğü tüm hizmet elemanları için 0000 grubunu ayırmıştır ve hizmet nesnelerini DICOM komut nesneleri olarak adlandırır.

Her hizmet nesnesinin sonunda bir veri kümesi tipi özelliği "(0000, 0800)" bulunur. Eğer bu özellik değeri 0101 ise, bu hizmetin DICOM verisi transfer etmediği belirtilmiş olur. Buna karşılık eğer bu eleman 0101'den farklı bir değer ise, hizmet komut nesnesinden sonra veri nesnesi gönderilecektir.

Birleşik verilerle ilgili DIMSE hizmetleri DIMSE-C hizmetleri olarak adlandırılırlar. Normalleştirilmiş verilerle ilgili DIMSE hizmetleri ise DIMSE-N olarak adlandırılırlar. C ve N harfleri genellikle hizmet isminin önüne gelirler, örneğin C-Store hizmeti birleşik yapıdaki DICOM görüntüsünü depolar. Benzer şekilde hizmet talep eden DIMSE mesajları Rq (Request) son ekiyle etiketlenirler. Örneğin bir BT tarayıcısı, arşive gönderdiği görüntünün depolanması talebinde bulunmak için C-Store-Rq mesajını kullanır. Hizmet yanıtları ise Rsp (Response)

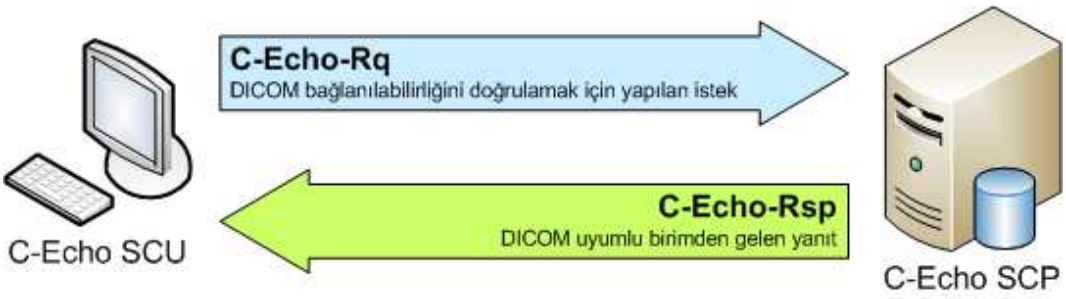
son ekiyle etiketlenir. Örneğin, arşiv tarafından tarayıcıya cevap göndermek amacıyla C-Store-Rsp mesajı kullanılır.

3.6.2. C-Echo

C-Echo en basit DIMSE hizmetidir. Bir DICOM AE'nin diğerine bağlandığını doğrulamak için kullanılır. Bu kolaylık C-Echo'yu en temel ve sık kullanılan bir hizmet haline getirir. Çünkü:

1. İki DICOM aygıtının birbirlerine bir ağ kablosuyla fiziksel olarak bağlı olduklarını bilmek yeterli değildir.
2. Bir aygıtın diğerine TCP/IP protokolünü kullanarak bağlanabilmesi de yeterli değildir. Çünkü bu bağlantı onların DICOM bağlantısı sağlayıp sağlamayacaklarına dair bir bilgi vermez..

İki AE'nin birbirlerine bağlanmaları için, uygun şekilde yapılandırıldığını doğrulamanın tek yolu C-Echo'dur. İstek yapan AE, C-Echo-Rq mesajı gönderdiğinde karşı taraftaki AE geçerli bir C-Echo-Rsp mesajı gönderirse bu iki AE'nin uygun biçimde DICOM bağlantılı oldukları söylenebilir. Şekil 3.5'te iki taraf arasında gerçekleştirilen C-Echo iletişimi verilmiştir.



Şekil 3.5 C-Echo Protokolü (Pianykh, 2008)

DIMSE tarafından oluşturulan C-Echo-Rq ve C-Echo-Rsp mesajlarının içeriği ise Çizelge 3.5 ve 3.6'da gösterilmiştir.

Çizelge 3.5 C-Echo-Rq Mesajı (NEMA, 2008)

Alan	**Etiket**	**VR**	**Değer/Tanım**
Group Length	(0000,0000)	UL	Byte cinsinden mesajın uzunluğunu
Affected Service Class UID	(0000,0002)	UI	1.2.840.10008.1.1

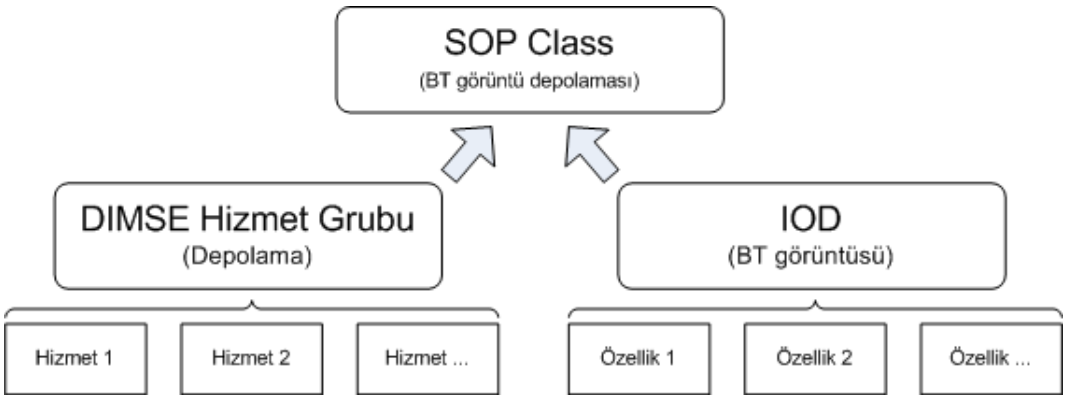
Command Field	(0000,0100)	US	0030
Message ID	(0000,0110)	US	Mesajın tekil numarası
Data Set Type	(0000,0800)	US	0101

Çizelge 3.6 C-Echo-Rsp Mesajı (NEMA, 2008)

Alan	**Etiket**	**VR**	**Değer/Tanım**
Group Length	(0000,0000)	UL	Byte cinsinden mesajın uzunluğunu
Affected Service Class UID	(0000,0002)	UI	1.2.840.10008.1.1
Command Field	(0000,0100)	US	8030
Message ID Being Responded To	(0000,0110)	US	C-Echo-Rq mesajının içeriğinde bulunan Message ID alanındaki değeri
Data Set Type	(0000,0800)	US	0101
Status	(0000,0900)	US	0000

3.6.3. Hizmet-nesne çiftleri

DIMSE hizmetlerinin uygun IOD nesneleriyle eşlenmesi ile hizmet nesne çiftleri (Service-Object Pair – SOP) oluşur. Yani DICOM veri nesneleri, verinin nasıl işlenmesi gerektiğine uygun yönergelerle birleştirilir. Şekil 3.6, DICOM SOP sınıfının genel yapısını göstermektedir.



Şekil 3.6 SOP Sınıf Yapısı (Pianykh, 2008)

3.6.4. Doğrulama SOP'u

SOP doğrulaması iki AE arasındaki DICOM bağlanılabilirliğini doğrular. Doğrulama SOP Class UID'i (1.2.840.10008.1.1), C-Echo mesajı içerisinde bir SOP parametresi olarak kullanılmaktadır. Bağlanılabilirlik doğrulaması, veri işleme gereksinimi doğurmadığı için doğrulama SOP'unun IOD'u bulunmamaktadır.

Doğrulama SOP'u C-Echo ile tam olarak aynı şey değildir. Doğrulama SCU'su bağlantıya geçtiği karşı tarafa, DICOM standartında haberleşme olanakları olup olmadığını sorar ve Evet yanıtını bekler. Doğrulama SOP'unun başarısız olması şu nedenlerden dolayı olabilir:

1. Genel bir ağ bağlantı hatası oluşmuş olabilir. Bağlanmamış bir ağ kablosu, yanlış yapılandırılmış IP adresi, güvenlik duvarı gibi bir bağlantı engelleyici gibi.
2. Ağ bağlantısı sorunsuz fakat DICOM yapılandırılması yanlış olabilir. IP, port ya da AE ismi gibi bilgiler yanlış olabilir.

Doğrulama SOP Sınıfı, DICOM ağı içerisinde en temel rolü oynar. Bu nedenle, kullanıcıların DICOM bağlantısı kurup kuramadıklarını sınamak amacıyla, pek çok DICOM uygulamasında bağlantı doğrulama özelliği bulundurulur.

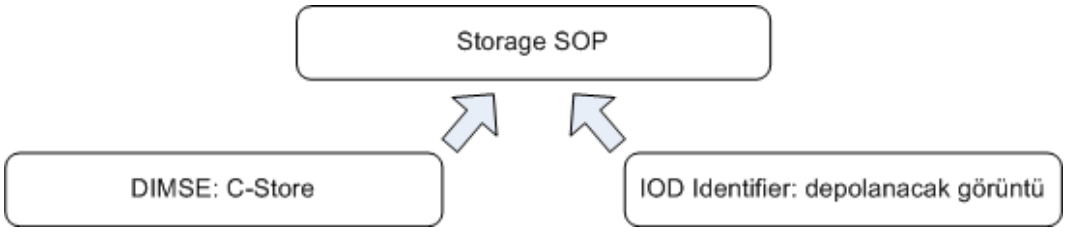
3.6.5. Depolama

Depolama SOP'u olan C-Store, AE'ler arasında DICOM görüntülerin ve diğer veri tiplerinin taşınmasından sorumlu olan temel işlemdir. C-Store DICOM veri nesnelerini bir DICOM ağı üzerinde, bir AE'den diğerine taşır. DICOM her modaliteyi (görüntüleme cihazını) ya da veri tipini, ayrı bir SOP sınıfı ile eşler. Bu nedenle DICOM depolama, Çizelge Ek 1.1'de verilen veriye özgü depolama SOP'ları ile tanımlanır. Şekil 3.7'de ise DICOM'da depolama SOP'unun nasıl oluştuğu görülebilir.

3.6.5.1. C-Store IOD

C-Store IOD bileşeni depolanacak olan veriyi içerir. Bu verinin tipi (MR, BT, ultrason gibi) Çizelge 3.7'de gösterilen C-Store SOP Class UID'ini belirtir.

Eğer birden fazla görüntü depolanmak istenirse, her görüntü ayrı bir C-Store isteği ile iletilir.



Şekil 3.7 Storage SOP (Pianykh, 2008)

3.6.5.2. C-Store DIMSE

C-Store, görüntüyü kabul eden AE'yi yapılması gereken konusunda bilgilendirir. C-Echo nun yaptığı gibi C-Store'un da C-Store SCU tarafından gönderilen bir istek parçası ve C-Store SCP tarafından yanıtlanan bir yanıt parçacığı vardır. Çizelge 3.7'de C-Store Rq mesajının içeriği verilmiştir. Görüldüğü gibi C-Store-Rq mesajında C-Echo'dakinden farklı parametreler bulunmaktadır. Bunlar:

1. *Priority*: Öncelik düşük, normal ve yüksek olarak sınıflandırılabilir. Pek çok DIMSE mesajında öncelik alanı, varsayılan olarak ayarlanmıştır.
2. *Affected SOP instance UID*: Gönderilen görüntünün tekil tanımlayıcısı.
3. *Move originator title and message ID fields*: C-Store isteği C-Move ya da C-Get DIMSE komutları ile çalıştırılmış olabilir. Böyle bir durum söz konusu olduğunda, isteği yapan AE'nin ismi ve istek mesajına ait mesaj numarası, ilgili parametreler ile gönderilmelidir.
4. *Data set type*: C-Store-Rq mesajını izleyen bir görüntü olması gerektiği için genellikle 0101 dışında bir değer olarak belirlenir. Bu değer için çoğunlukla 0000 kullanılır.

Çizelge 3.7 C-Store-Rq Mesajının İçeriği (NEMA, 2008)

Alan	Etiket	VR	Değer/Tanım
Group Length	(0000,0000)	UL	Byte cinsinden mesajın uzunluğunu
Affected Service Class UID	(0000,0002)	UI	Çizelge Ek 1.1'deki SOP Class UID'lerden biri
Command Field	(0000,0100)	US	0001

Message ID	(0000,0110)	US	Mesajın tekil numarası
Priority	(0000,0700)	US	0002 (Düşük), 0000 (Orta), 0001 (Yüksek)
Data Set Type	(0000,0800)	US	0101
Affected SOP Instance UID	(0000,1000)	UI	Depolanacak görüntünün UID'i
Move Originator AE Title	(0000,1030)	AE	C-MOVE işlemini başlatan uygulamanın AE ismi
Move Originator Message ID	(0000,1031)	US	C-Move-Rq mesajı ile gelen Message ID alanındaki değer

3.6.6. Sorgulama

DICOM sorgularını gerçekleştirmek amacıyla standartta 3 tane C-Find SOP sınıfı tanımlanmıştır. Çizelge 3.8'de bu SOP sınıfları ve sınıfların tekil tanımlayıcıları görülebilir.

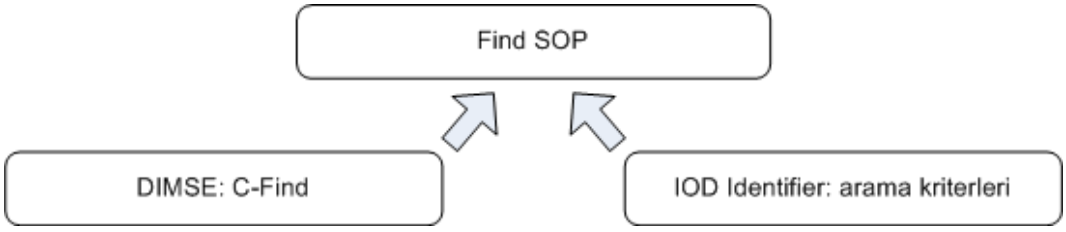
Çizelge 3.8 Sorgulama SOP'u (NEMA, 2008)

SOP Sınıfı	**SOP Class UID**
Patient Root Query/Retrive Find	1.2.840.10008.5.1.4.1.2.1.1
Study Root Query/Retrive Find	1.2.840.10008.5.1.4.1.2.2.1
Patient-Study Root Query/Retrive Find	1.2.840.10008.5.1.4.1.2.3.1

Modaliteler (görüntü cihazları), arşivlerdeki görüntü verilerini sorgulamaz. Bu yüzden C-Find, modalite tabanlı SOP sınıflarını içermez. Bu yönüyle C-Find, C-Echo ile benzeşir.

DICOM tüm C-Find veri aramalarını 3 veri düzeyine böler: hasta, çalışma ve hastaya ait çalışma. Bu düzeyler kök olarak adlandırılırlar ve bu kökler üzerinde arama yapabilmek için ayrı ayrı SOP'lar bulunur.

DICOM sorgu köklerine olan gereksinim, DICOM standardında tanımlı Hasta-Çalışma-Seriler-Görüntü hiyerarşisinden doğmuştur. DICOM veriyi dört hiyeraarşik seviyeye bölmüştür ve veri araması bu düzeylerle sınırlandırıldığında en iyi sonucu vermektedir. Sorgulama SOP'u da diğer SOP'lar gibi, DIMSE + IOD ikilisi ile oluşturulur. Diğerlerinden farklı olarak ise, IOD içerisinde arama kriterleri tutulur. Şekil 3.8'de sorgulama SOP'unun bileşenleri görülebilir.



Şekil 3.8 Sorgulama SOP'u (Pianykh, 2008)

Şekil 3.9'da ise sorgulama işleminin AE'ler arasında nasıl gerçekleştiği görülebilmektedir.

Şekil 3.9 C-Find Protokolü ve Örneği (Pianykh, 2008)

3.6.6.1. DICOM'da Sorgulama Şekilleri

C-Find SCU'su bir sorgulama isteği gönderdiğinde, bu istek eşleştirilmesi gereken bir takım özellikler içerir. Bu özellikler arama alanları olarak çalışırlar ve hedef AE üzerinde bulunan verilerle karşılaştırılırlar. DICOM aramaların detaylandırılması için bir kaç yol önermektedir (Pianykh, 2008):

1. *Sembol Eşleme*: Örneğin "Ay" ile başlayan bir isim ararken "Ay*" şeklinde bir arama ölçütü oluşturduğumuz da burdaki "*" sembolü yerine gelebilecek bir karakter dizisinin yerine kullanılmış olur. (Aylin, Aynur, Aysel, Aydın gibi) Başka bir DICOM sembolü ise "?" dir. Bilinmeyen tek bir karakter yerine kullanıldığı için çok da kullanışlı değildir. Örneğin "Ayd?n " şeklinde yapılacak bir sorgu ile Aydan, Aydın gibi isimler bulunabilir.
2. *Liste Eşleme* : "\" sembolünü kullanır. Bu sembol "veya" anlamına gelir. Sembol eşleme yöntemiyle de birleştirilerek kullanılabilir. "Aydın\Ahmet" sorgusu ile aranan alan içerisinde "Aydın" veya "Ahmet" geçen tüm sonuçlar bulunabilir.

3. *Genel Eşleme* : En genel eşleme tipidir. Özetle, eğer bir özellik “0” uzunluğundaysa her şeyle eşleşecektir. Örneğin; Hasta Adı alanının boş bırakıldığı bir C-Find isteği, bütün hasta isimlerinin listelenmesini sağlayacaktır.
4. *Aralık Eşleme* : Bu tip eşleme tarih ve zaman gibi aralıklarla ifade edilebilecek özellikler için kullanılır. Özelliğin başlangıç ve bitiş değerleri belirtilerek bir eşleme aranır. Örneğin çekim tarihi aramasında “20090801-20090831” sorgusu 2009 yılının Ağustos ayında yapılan tüm çekimleri döndürecektir.
5. *Dizi Eşleme*: Belki de en karmaşık DICOM özellik eşleme şeklidir. Burada birden fazla özelliğin birlikte araştırılması durumu söz konusudur ve bu özelliklerin arasında mantıksal bir “ve” bağlacı varmış gibi eşleme sağlanmalıdır.
6. Tek değer Eşleme: Özelliğin kesin değerinin sorgulanması esasına dayanır. Semboller ya da aralık içermez. Genellikle tekil tanımlayıcıları sorgulama işleminde kullanılır.

3.6.6.2. C-Find IOD

C-Find IOD’ları C-Find hizmet sağlayıcıları üzerinde (örneğin bir görüntü arşivi) eşlenmesi gereken arama parametrelerini içerir. C-Find sorguları istisnasız daha önce belirtilen Hasta-Çalışma-Seriler-Görüntü hiyerarşisine göre çalışır. Sorgu, bu 4 düzeyden birisine ait olmalıdır. Örnek bir C-Find IOD’u Çizelge 3.9 üzerinde incelenebilir.

Çizelge 3.9 Sorgu parametreleri (NEMA, 2008)

Etiket	**İsim**	**Örnek**	**Eşleştirme**
(0008,0052)	Query Level	STUDY	PATIENT, STUDY, SERIES ve IMAGE’dan biri olabilir
(0010,0010)	Patient’s name	Aydın*\Aydan*	Sembol eşleme
(0010,0020)	Patient ID	12345	Tek değer eşleme
(0008,0020)	Study date	20090801-20090831	Aralık eşleme
(0008,0030)	Study time	090000-180000	Aralık eşleme
(0008,0050)	Accession number	Abc123	Tek değer eşleme

(0020,0010)	Study ID	1.234.567	Çoğunlukla tek değer eşleme
(0008,1030)	Study description	*kırık\çatlak*	Çoğunlukla sembol eşleme
(0008,0090)	Referring physician name	*Mustafa*	Sembol eşleme
(0008,1060)	Reading physician name	*Nuri*	Sembol eşleme
(0008,0061)	Modalities in study	MR\CT	Tek değer ya da liste eşleme
(0010,0030)	Patient's birth date	19830101-19831231	Aralık eşleme

3.6.6.3. C-Find DIMSE

C-Store gibi, C-Find DIMSE de C-Find IOD'larını değişik AE'ler arasında taşıyarak, bir ağ taşımacılığı işi yürütür. Bu nedenle yapısı C-Store'dan çok da farklı değildir. Bir C-Find-Rq mesajı bir C-Find IOD tarafından izlenir. Bu DIMSE + IOD çifti ağ üzerinde C-Find SCU uygulama biriminden C-Find SCP uygulama birimine doğru seyahat eder. Yanıt verecek olan uygulama birimi önce C-Find DIMSE'yi alır ve inceler. Daha sonra C-Find IOD'u alır ve eşleşme için veritabanını tarar. Eşleşme bulunduğunda, C-Find SCP bulunan her bir sonuç için C-Find IOD'u üreterek C-Find-Rsp mesajıyla yanıt verir. Genelde birden fazla eşleşmenin bulunmasını beklenir. Son eşleşme hariç, her eşleşme için C-Find SCP bir DIMSE,IOD çiftiyle yanıt verir ve IOD mevcut eşleşmeyi iletir. Son eşleşme işlemin başarıyla sonuçlandığı anlamını taşıyan 0000 durum mesajıyla birlikte döner. Eğer hiç bir eşleşme bulunamamışsa C-Find tek bir C-Find-Rsp mesajıyla veri seti tipini 0101 (null) değerine ayarlar. Bazı durumlarda da hatalarla karşılaşılabilir. C-Find böyle durumlarda bir hata değeri döndürür. Yaygın olarak görülen hatalardan bazıları aşağıda belirtilmiştir (Pianykh, 2008).

1. *Yanlış yapılandırma*: Örneğin C-Find SCU'nun AE'si, C-Find SCP'nin yapılandırılmasında girilmemişse bu türden bir hata oluşur. En yaygın DICOM bağlantı hatasıdır. İki AE'den birisinin diğerini tanımadığı durumlarda oluşur.
2. *Yanlış biçimlendirilimiş C-Find isteği*: Hiyerarşik sorgu kurallarının sağlanmaması durumunda ortaya çıkar. Örneğin görüntünün çalışma

anahtar özelliklerini bilmeden sorgulama yapılmaya çalışılınca ortaya çıkar.

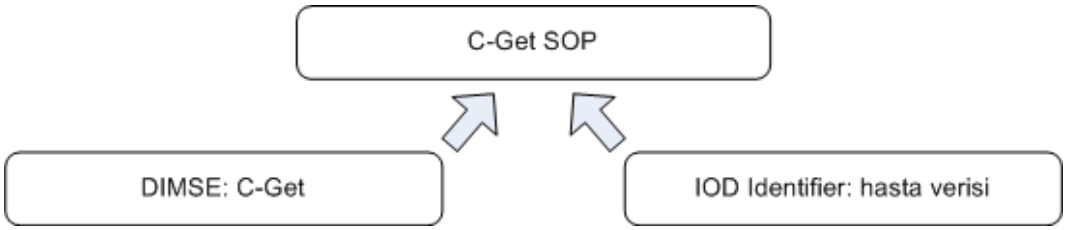
3. *Uyumsuz Sorgu SOP'u*: Eğer bir uygulama çalışma kökünde bir sorgu göndermiş ve uygulama sadece hasta kökünde sorguları kabul ediyorsa bu türden bir hata oluşur.
4. *Desteklenmeyen C-Find SCP*: Tüm DICOM modalitelerinin C-Find SCP desteği sağlaması beklenmemelidir. Modalitelerin pek çoğu uzak arşive görüntü eklemek üzere programlanmışlardır.

3.6.6.4. C-Cancel

Çok büyük bir PACS arşivi sorgulamak istendiğinde eğer sorgulama sınırlandırılmadıysa çok fazla miktarda sonuç geri dönebilir. Böyle bir durumda C-Find SCU sonuçları yeterli görüp C-Cancel mesajını SCP'ye göndererek işlemi yarıda kesebilir. Kısacası C-Cancel mesajı çalışan bir C-Find sorgusunu sonlandırmaya yardımcı olur.

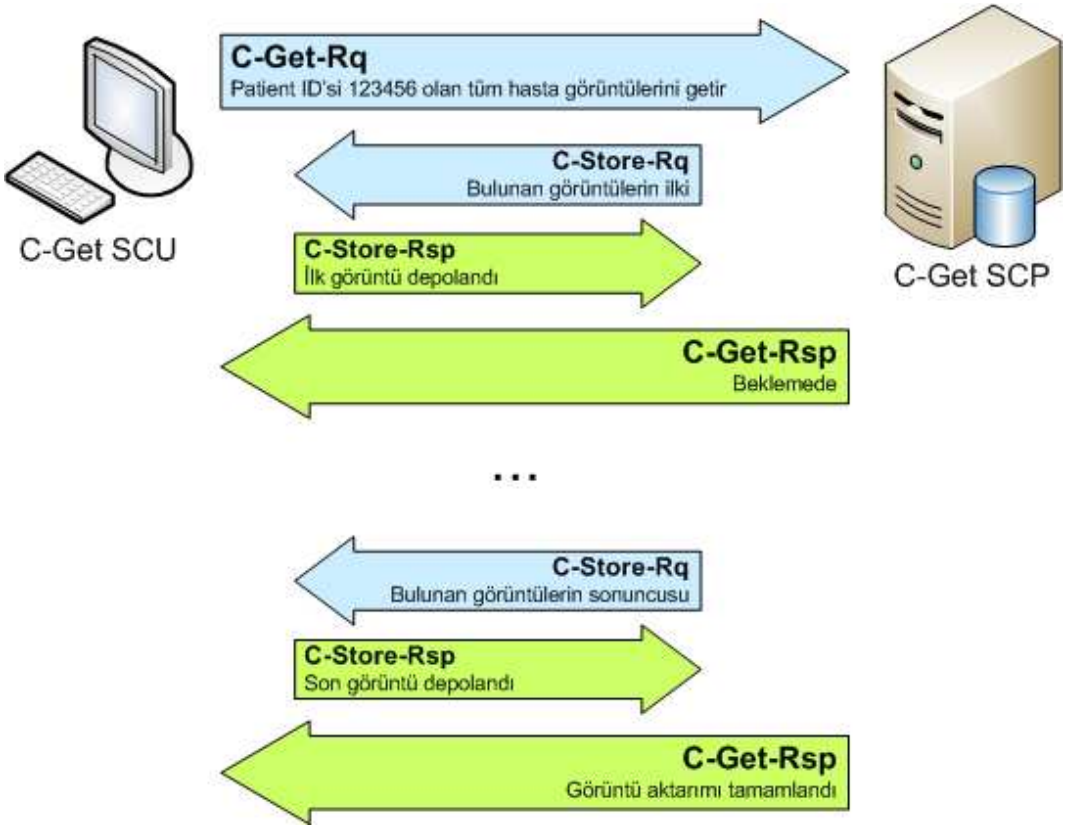
3.6.7. C-Get

Temel DICOM görüntü erişimi, C-Get SOP'u aracılığıyla gerçekleştirilir. Kavramsal olarak C-Get, C-Find ve C-Store komutlarını tek bir serviste birleştirerek gereksinim duyulan görüntüyü bir C-Find benzeri sorguyla bulup C-Store ile görüntüye erişimi sağlar. C-Find'da olduğu gibi, C-Get isteğinde de görüntü arama ölçütüne bir tane IOD belirleyici nesnesi iliştirilir. C-Get SOP'unun C-Get DIMSE ve C-Get IOD'dan oluştuğu Şekil 3.10'da gösterilmiştir.



Şekil 3.10 C-Get SOP (Pianykh, 2008)

İstek C-Get SCP'ye gönderildiğinde, SCP öncelikli olarak görüntüyü bulmak için arama parametrelerini kullanır ve daha sonra görüntüyü C-Get SCU'suna göndermek için C-Store işlemine başvurur. C-Get'in çalışma şekli Şekil 3.11'de detaylı bir şekilde gösterilmektedir.



Şekil 3.11 C-Get Protokolü (Pianykh, 2008)

Çizelge 3.10'da görülebildiği gibi C-Get, C-Find'da kullanılan üç sorgu/erişim kökünü de içerir. En çok kullanılanı ise, çalışma köküdür (Study Root).

Çizelge 3.10 C-Get SOP'u (NEMA, 2008)

SOP Sınıfı	SOP Class UID
Patient Root Query/Retrive Get	1.2.840.10008.5.1.4.1.2.1.3
Study Root Query/Retrive Get	1.2.840.10008.5.1.4.1.2.2.3
Patient-Study Root Query/Retrive Get	1.2.840.10008.5.1.4.1.2.3.3

3.6.7.1. C-Get IOD

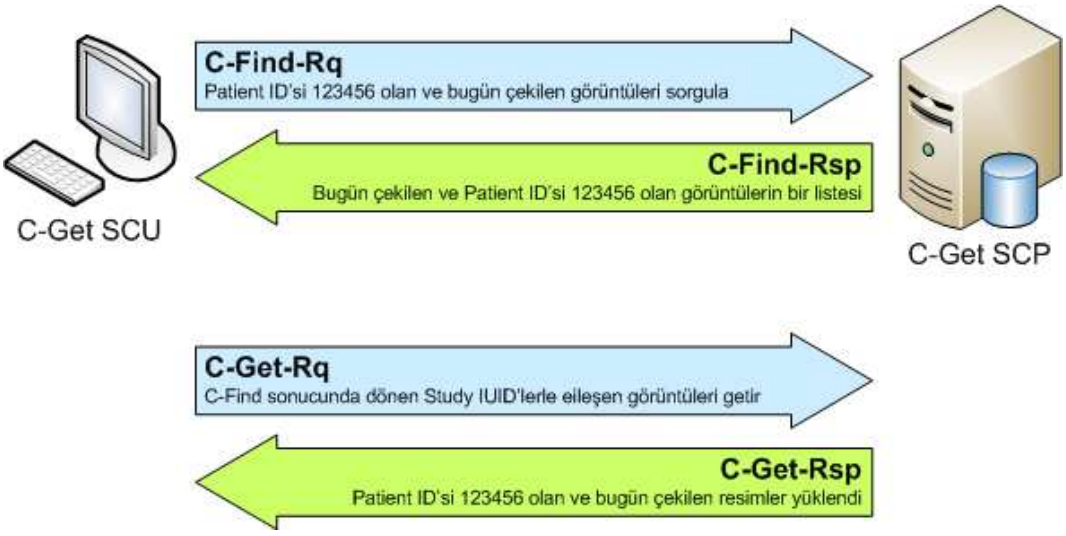
C-Get-IOD erişilecek olan görüntüler için arama kriterlerinin iletimini sağlar. Bu özelliğiyle C-Find IOD'a benzemektedir. Sürecin C-Get SCP tarafında ise, eşlenmesi beklenen özellikler sağlanır. Bu özelliklerle uyuşan görüntüler geri döndürülür. C-Find'da olduğu gibi, C-Get için de arama özellikleri Hasta-Çalışma-Seriler-Görüntü şeklindeki hiyerarşik arama mantığına uygun olmalıdır.

C-Find ile C-Get IOD'ları arasında tek bir fark bulunmaktadır. C-Get IOD'lar içerisindeki arama işleminde sadece tekil tanımlayıcılar ("Patient ID", "Study Instance UID", "Series Instance UID" veya "SOP Instance UID") kullanılır. C-Get IOD tekil tanımlayıcılar haricindeki arama anahtarları kullanılmaz. Örnek bir C-Get IOD'u Çizelge 3.11'de görülebilir.

Çizelge 3.11 Örnek bir C-Get IOD (NEMA, 2008)

Etiket	**İsim**	**Örnek**	**Eşleştirme**
(0008,0052)	Retrieve Level	STUDY	PATIENT, STUDY, SERIES ve IMAGE'dan biri olabilir
(0010,0020)	Patient ID	123456	Tek değer eşleme
(0020,000D)	Study Instance UID	1.2.840.1234567	Tek değer ya da liste eşleme

Arşivlerden görüntü getirme işlemlerinde, kullanıcılar alt seviyedeki tekil tanımlayıcıları bilmedikleri için, C-Get doğrudan kullanılmaz. C-Get işlemi öncesinde C-Find işlemi gerçekleştirilmelidir. C-Find işlemi ile öğrenilen tekil tanımlayıcılar ile C-Get işlemi gerçekleştirilir. Şekil 3.12'de bu işlem gösterilmiştir.



Şekil 3.12 Görüntüleri getirmek için ardarda kullanılan C-Find ve C-Get işlemi (Pianykh, 2008)

3.6.7.2. C-Get DIMSE

C-Get DIMSE, C-Find ve C-Store ile büyük ölçüde benzerlikler taşır. Daha önceki bölümlerde de anlatıldığı gibi, DIMSE + IOD çiftleri ağ üzerinde C-Get SCU AE'sinden C-Get SCP AE'sine doğru seyahat ederler. Tanılama işleminden sonra, C-Get IOD üzerindeki verilerle veritabanındaki eşleşmeler araştırılır. Eşleşme bulunduğunda C-Get SCP, C-Get-Rsp mesajıyla bulunan her C-Get IOD'u için yanıt döndürür.

C-Get, görüntülerin aktarımı için C-Store işlemini kullanır. C-Get-Rsp mesajının içeriğinin yer aldığı Çizelge 3.12'deki son 4 özellik, C-Store işleminin çalışmasına ilişkin güncel istatistikleri döndürmek için kullanılır. Bu istatistikler; kaç tane görüntünün iletildiği, kaç tanesinin iletiminde başarısızlık olduğu gibi bilgileri, uyarılarla birlikte gösterir. C-Get SCP, C-Store işleminin çalışması sırasında "bekliyor" şeklinde C-Get-Rsp mesajı döndürüyorsa; mesajın içerisinde bulunan istatistiksel numaralar gönderim sürecinin özetini alıcı tarafa sunar.

Çizelge 3.12 C-Get-Rsp Mesajının İçeriği (NEMA, 2008)

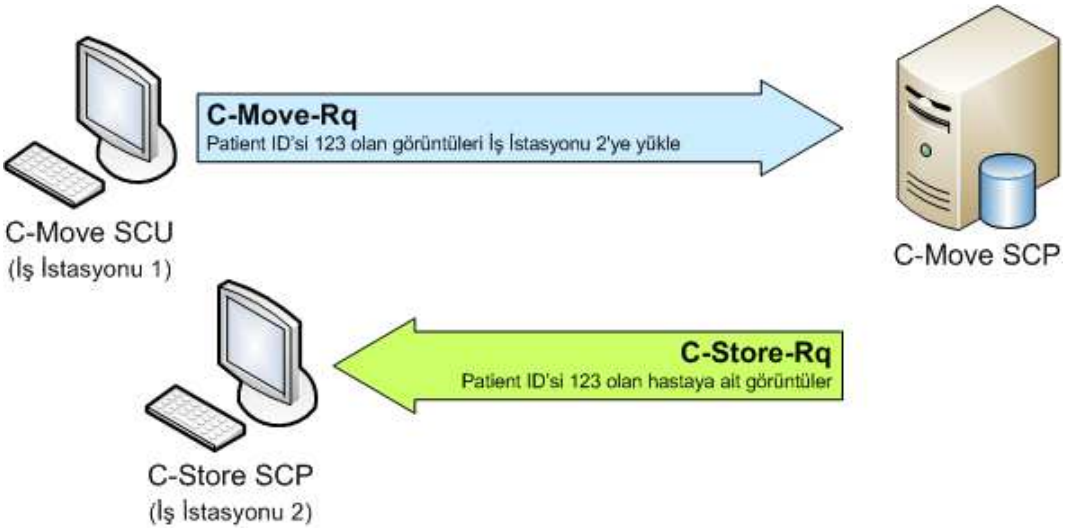
Alan	Etiket	VR	Değer/Tanım
Group Length	(0000,0000)	UL	Byte cinsinden mesajın uzunluğunu
Affected Service Class UID	(0000,0002)	UI	Çizelge 3.10'daki SOP Class UID'lerden biri
Command Field	(0000,0100)	US	8010
Message ID Being Responded To	(0000,0110)	US	C-Get-Rq mesajında gönderilen Message ID değeri ile eşit olmalıdır
Data Set Type	(0000,0800)	US	0101
Status	(0000,0900)	US	0000 (Başarılı), FF00(Beklemede)
Number of Remaining Suboperations	(0000,1020)	US	Çalıştırılması gereken toplam C-Store alt işlemi
Number of Completed Suboperations	(0000,1021)	US	Çalıştırılmış C-Store alt işlemlerinin toplamı
Number of Failed Suboperations	(0000,1022)	US	Çalıştırılan fakat sonucu hatalı olan C-Store alt işlemlerinin toplamı
Number of Warning Suboperations	(0000,1023)	US	Çalıştırılan fakat sonucunda uyarı oluşan C-Store alt işlemlerinin toplamı

Tüm görüntü gönderimleri bittikten sonra C-Get-SCP tarafı, son bir C-Get-Rsp mesajı ile işlemin bittiğini SCU tarafına iletir.

C-Get işlemi de C-Find gibi, C-Cancel mesajı ile istenildiği zaman yarıda kesilebilir.

3.6.8. C-Move

C-Move pratikte, C-Get ile aynı işleve sahiptir. Tek farkı, C-Move ile C-Move SCU, C-Move SCP'den görüntüleri başka bir AE'ye göndermesini talep edebilir. Yani C-Get işlemi görüntüleri istemede kullanırken; C-Move buna ek olarak farklı bir AE'ye bu görüntüleri göndertebilir. Şekil 3.13'de bu işlem gösterilmektedir.



Şekil 3.13 C-Move İşlemi ile Diğer Bir İstasyona Görüntülerin Gönderilmesi (Pianykh, 2008)

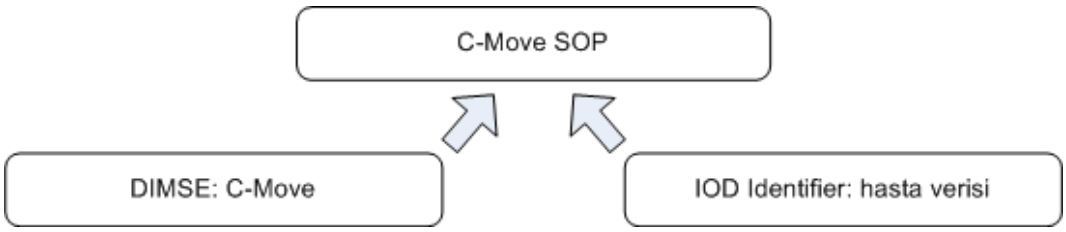
En basit senaryoda bir AE, C-Move mesajını kendisine görüntü çekmek için kullanabilir. Bu durumda C-Move, C-Get ile aynı işlemi yapmış olur. Aslında bu işlem sık kullanılan bir durumdur. Çoğu PACS'ta, C-Move basit C-Get işlemi yerine kullanılır.

C-Get'ten diğer bir farkı C-Move'da SCP'nin görüntüyü göndereceği AE'yi tanıması gerektiğidir. Yani C-Get tek bir ilişki (association) üzerinde çalışırken C-Move görüntüyü göndereceği AE ile bir ilişki (association) kurmalıdır. Bu ilişkiyi kurmak için C-Move SCP alıcı tarafın AE ismini, IP adresini (ya da alan adını) ve bağlantı noktasını bilmelidir.

Ayrıca C-Move'un sorgulamada kullandığı kökler, Çizelge 3.13'den de anlaşılacağı gibi, C-Get ile benzerdir. SOP yapısı ise Şekil 3.14'de görülebileceği gibi aynıdır.

Çizelge 3.13 C-Move SOP'u (NEMA, 2008)

SOP Sınıfı	SOP Class UID
Patient Root Query/Retrive Move	1.2.840.10008.5.1.4.1.2.1.2
Study Root Query/Retrive Move	1.2.840.10008.5.1.4.1.2.2.2
Patient-Study Root Query/Retrive Move	1.2.840.10008.5.1.4.1.2.3.2



Şekil 3.14 C-Move SOP'u (Pianykh, 2008)

3.6.8.1. C-Move IOD

C-Move IOD, C-Get'te olduğu gibi seviyeler esas alınarak getirilecek görüntüler için arama özelliklerini taşır. Çizelge 3.14'de C-Move IOD'a bir örnek verilmiştir.

Çizelge 3.14 Örnek bir C-Move IOD

Etiket	İsim	Örnek	Eşleştirme
(0008,0052)	Retrieve Level	STUDY	PATIENT, STUDY, SERIES ve IMAGE'dan biri olabilir
(0010,0020)	Patient ID	123456	Tek değer eşleme
(0020,000D)	Study Instance UID	1.2.840.1234567	Tek değer ya da liste eşleme

C-Get bölümünde de bahsedildiği gibi, C-Move ve C-Get'in içerisinde yer alan sorgulama işlemi görüntüyü bulmak için yeterli olmadığından dolayı, bir

görüntünün getrilebilmesi için C-Move ve C-Get işleminden önce bir C-Find işlemi ile sorgulamanın yapılmış olması beklenir.

3.6.8.2. C-Move DIMSE

C-Move DIMSE C-Get'in özellikleri haricinde görüntünün gönderileceği AE'yi içeren bir özellik (Move Destination) barındırır.

"(0000, 0600)" etiketiyle tutulan bu ek bilgi, görüntünün C-Move-SCP tarafından, C-Store alt-işlemi ile nereye gönderileceğini tutar. C-Move isteğini kabul eden C-Move-SCP ise şu işlemleri gerçekleştirir (Pianykh, 2008):

1. IOD'a eklenmiş tekil tanımlayıcılar ile kendi veritabanında eşleşen görüntüleri bulur.
2. Her görüntü için farklı bir C-Store isteğini hedef AE'ye gönderir. Çizelge 3.15'de görülebileceği gibi C-Move-Rq mesajında sadece hedef AE'nin ismi yer alır. C-Move-SCP'nin bu AE ile ilişki kurabilmesi için IP adresi ve bağlantı noktası gibi erişim parametrelerini daha önceden bilmesi gerekir.
3. Görüntüler hedef AE'ye C-Store ile gönderilirken C-Move-SCP isteği yapan AE'ye (C-Move SCU) "beklemede" değerini içeren bir C-Move-Rsp mesajı gönderir. C-Move Rsp mesajının içeriği Çizelge 3.16'dan incelenebilir.
4. Görüntü aktarımı bittikten sonra C-Move-SCP "başarılı" değerini içeren bir C-Move-Rsp mesajını C-Move SCU'ya iletir ve işlem sonlanır.

C-Cancel işlemi bu adımların herhangi bir aşamasında işlemi yarıda kesebilmek için kullanılabilir.

Çizelge 3.15 C-Move-Rq Mesajının İçeriği (NEMA, 2008)

Alan	**Etiket**	**VR**	**Değer/Tanım**
Group Length	(0000,0000)	UL	Byte cinsinden mesajın uzunluğunu
Affected Service Class UID	(0000,0002)	UI	Çizelge 3.13'teki SOP Class UID'lerden biri
Command Field	(0000,0100)	US	0021
Message ID	(0000,0110)	US	Mesajın tekil numarası

Priority	(0000,0700)	US	0002 (Düşük), 0000 (Orta), 0001 (Yüksek)
Data Set Type	(0000,0800)	US	0101
Move Destination	(0000,0600)	AE	Görüntülerin gönderilmesi istenen AE

Çizelge 3.16 C-Move-Rsp Mesajının İçeriği (NEMA, 2008)

Alan	**Etiket**	**VR**	**Değer/Tanım**
Group Length	(0000,0000)	UL	Byte cinsinden mesajın uzunluğunu
Affected SOP Class UID	(0000,0002)	UI	Çizelge 3.13'teki SOP Class UID'lerden biri
Command Field	(0000,0100)	US	8021
Message ID Being Responded To	(0000,0110)	US	C-Move-Rq mesajında gönderilen Message ID değeri ile eşit olmalıdır
Data Set Type	(0000,0800)	US	0101
Status	(0000,0900)	US	0000 (Başarılı), FF00(Beklemede)
Number of Remaining Suboperations	(0000,1020)	US	Çalıştırılması gereken toplam C-Store alt işlemi
Number of Completed Suboperations	(0000,1021)	US	Çalıştırılmış C-Store alt işlemlerinin toplamı
Number of Failed Suboperations	(0000,1022)	US	Çalıştırılan fakat sonucu hatalı olan C-Store alt işlemlerinin toplamı
Number of Warning Suboperations	(0000,1023)	US	Çalıştırılan fakat sonucunda uyarı oluşan C-Store alt işlemlerinin toplamı

C-Move ile C-Get bazı durumlarda aynı görevi yerine getirse de ikisi arasında bazı farklar bulunmaktadır. Bu farklar (Pianykh, 2008):

- C-Move işlemini gerçekleştirebilmek için görüntüyü gönderen ve alan tarafın birbirlerini tanıma gereksinimi, güvenlik açısından faydalıdır. Böylelikle tanınmayan AE'lere görüntü gönderilmesi engellenmiş olur.
- C-Get servisi çoğunlukla, birden fazla ilişki kuramayan sistemlerde kullanılmaktadır. Küçük gömülü sistemler veya düşük kapasiteli bilgisayarlar, birden fazla bağlantı açmaya izin vermeyebilir. Bu gibi durumlarda C-Get kullanılabilir.

- C-Get ile DICOM sunucularına İnternet üzerinden herhangi bir yerden bağlanılabilir. Böylece, mobil cihaz'ın IP adresi değişse dahi sunucuya bağlantı kurabilir. Ayrıca C-Get işleminin tek ilişki tabanlı modeli güvenlik duvarları ile dahi çalışabilmektedir Bu yüzden teleradyoloji uygulamalarında sıklıkla C-Get kullanılmaktadır. C-Move ise tarafların birbirlerini tanıma gereksinimlerinden dolayı daha durağan bir yapılandırmada kullanılabilir.

Yani C-Move daha çok yerel ağlardaki PACS uygulamalarında kullanılmaktadır. C-Get ise; PACS arşivlerine yerel ağın dışından yani İnternet üzerinen kolayca bağlanılabilmeyi sağlar.

3.7. DICOM İlişkileri

DICOM ilişki (association) kuralları, DICOM ağ bağlantıları için alt seviye protokolleri tanımlar. Daha önce bahsedilen üst seviye DIMSE mesajlaşmaları, bu kurallar üzerine kuruludur. DICOM ilişki protokollerinin asıl hedefi, iletişim kuran iki DICOM uygulamasının uyumlu olmasını sağlamanın yanı sıra belirlenen bir biçimde ve sırada verileri transfer etmelerini sağlamaktır. Bu hedefi sağlamak için ilk adım, iki taraf arasında DICOM ilişkisi (association) kurmaktır.

DICOM ilişki kuralları TCP/IP'nin üzerine kurulmuştur. Bu sayede TCP/IP'yi, DICOM nesneleri ve komutlarını iletebilecek düzeye getirir. Yani TCP/IP'yi, DICOM'a özgü gereksinimler ile genişletir. DICOM ilişki mekanizması, DICOM Upper Layer (UL) olarak adlandırılır ve iki AE'yi birbirine bağlar.

DICOM ilişki kurma işlemi ağ iletişiminin en başında gerçekleşir ve DICOM el sıkışması (handshake) olarak adlandırılır. El sıkışma esnasında her iki AE birbirinin işlevleri hakkındaki bilgileri öğrenir ve kullanılacak iletişim parametrelerini kararlaştırır.

3.7.1. İlişki kurma işleminin temelleri

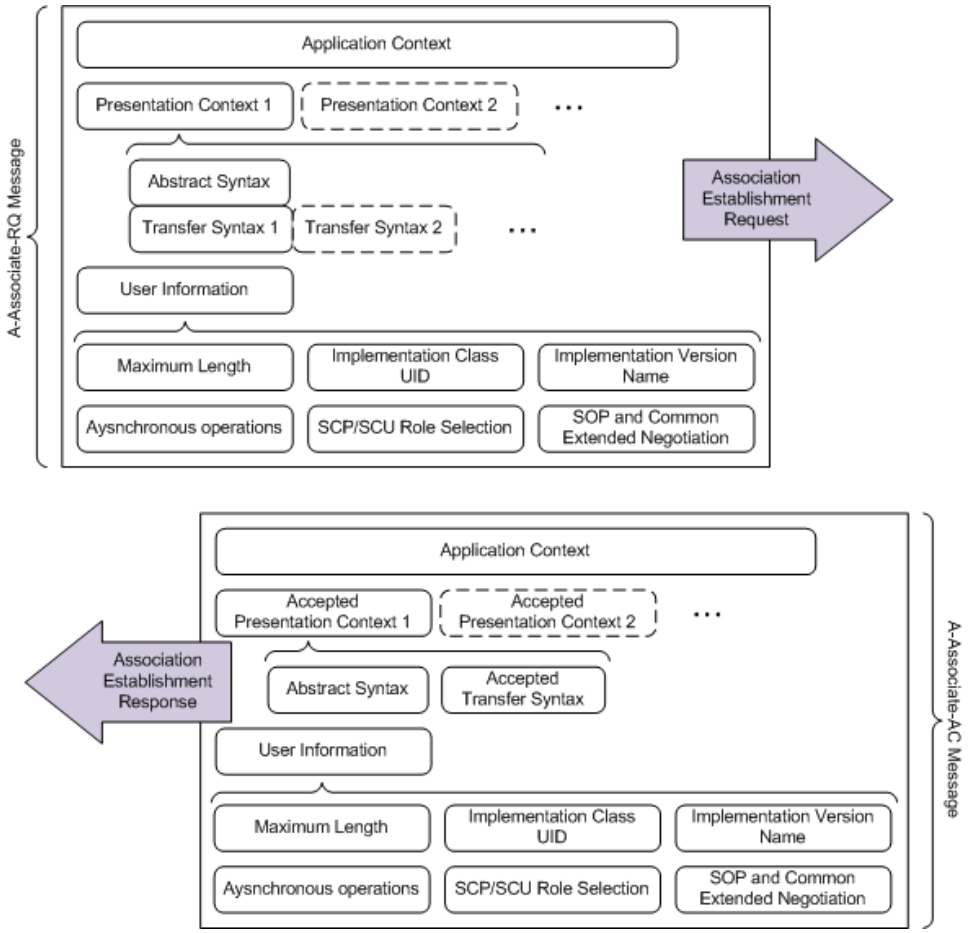
DICOM ilişki kurma işleminin temeli, sunum içeriğidir (Presentation Context). Bir AE ağ iletişimine başlamak istediğinde, kendisi hakkındaki tüm bilgileri sunum içeriği mesajına koyar ve alıcı AE'ye gönderir. Sunum içeriğini alan alıcı AE, ya içeriği kabul eder ya da bağlantıyı reddeder.

Sunum içeriğinin bir bölümü müzakere edilebilirdir. DICOM soyut sözdizimi (Abstract Syntax) olarak bilinen bölüm değiştirilemezken, DICOM transfer sözdizimi (Transfer Syntax) bölümü değiştirilebilirdir. Transfer sözdizimi daha esnektir ve bağlantı kuracak iki tarafta kabul edecekleri bir sözdiziminde karar kılarlar.

3.7.2. İlişki kurma

İlişki kurmak isteyen AE, alıcının ilişkiyi başlatmasını isteyen A-Associate-RQ mesajını oluşturur ve alıcıya gönderir. Bu mesaj, çeşitli sunum içerikleri ve ilişkiyi başlatan AE'nin yeteneklerini belirten kullanıcı bilgi verisinden oluşur.

Alıcı AE, teklif edilen tüm iletişim parametrelerine bakar ve en uygununu seçer. Ardından ilişkiyi kabul ederse, A-Associate-AC mesajı ile cevap verir. Alıcı AE'nin profiline önerilen parametrelerden hiçbiri uymuyorsa ilişkiyi, A-Associate-RJ mesajı ile reddeder. Şekil 3.15'te ilişki kurma istek ve cevabının içerisinde bulunan yapılar verilmiştir.



Şekil 3.15 İlişki Kurma İsteği ve Cevabının İçeriği (Pianykh, 2008)

3.7.3. Soyut söz dizimi

Soyut söz dizimleri (Abstract Syntax), DICOM uygulamalarının sunduğu servisleri belirtirler. Yani iletişim kuran AE'ler arasında desteklenen SOP'ları içerirler. Bazı önemli soyut söz dizimlerinin listesi Çizelge Ek 1.2'de görülebilir.

3.7.4. Transfer söz dizimi

Transfer söz dizimi iletişimin biçimini tanımlar. DICOM standardının PS3.5 bölümünde transfer söz dizimi şu şekilde tanımlanır:

"Bir transfer söz dizimi soyut söz dizimleri tarafından tanımlanan veri elemanlarını belirleyebilen kuralları kodlar"

Yani transfer söz dizimleri, gönderilen veri ve mesajların nasıl kodlanacağını ifade eder. Her DICOM uygulaması, en azından varsayılan olarak kullanılan Little Endian transfer söz dizimini desteklemelidir. Transfer söz dizimleri Çizelge Ek 1.3'ten incelenebilir.

Gönderilen görüntü piksel verisi sıkıştırılarak gönderilmiş ise, gönderen AE gönderilen görüntülerin sıkıştırılmış olduğunu transfer söz dizimi kullanarak bildirir. Alıcı AE teklif edilen sıkıştırma yöntemini desteklemiyor ise, her iki AE varsayılan söz dizimi olan 1.2.840.10008.1.2 transfer söz dizimini kullanmalıdır.

Tüm görüntü sıkıştırma algoritmaları, bilindiği üzere, çeşitli sıkıştırma parametrelerine bağlıdırlar. Fakat transfer söz dizimi, algoritmanın ayarları ile alakalı bilgileri değil sadece sıkıştırma algoritmasının ismini kodlar. Bu yüzden sıkıştırma parametrelerinin değerleri transfer söz dizimi aracılığı ile müzakere edilemez. Alıcı taraf gönderen tarafın seçtiği sıkıştırma parametrelerini doğrudan kullanmak zorundadır.

3.7.5. Uygulama içeriği

Uygulama içeriği metin dizisi, ilişkiyi isteyen uygulamanın içeriğini belirtir. İlişki isteğine cevap veren AE içeriği desteklemediği takdirde ilişkiyi yarıda kesebilir.

NEMA her bir DICOM uygulama geliştiricisine tekil uygulama içeriği isimleri vermekle sorumludur. Ayrıca uygulama geliştiricileri kendilerine ait uygulama içeriklerini tanımlayıp geliştirdikleri uygulamalarda kullanabilirler. Bu sayede; aynı firmanın uygulamaları, uygulama içeriği ile birbirlerini tanıyarak kendilerine has veri transferi gerçekleştirebilirler.

DICOM ayrıca varsayılan olarak bir uygulama içeriği ismi (1.2.840.10008.3.1.1.1) tanımlamıştır. Varsayılan uygulama içeriği birçok uygulama tarafından kullanılmaktadır.

3.7.6. Sunum içeriği

Sunum içeriği, önerilen işlevsellik ile biçimi birleştirir ve soyut söz dizimine ek olarak müzakere edilebilen transfer söz dizimi listesinden oluşur. Çizelge 3.17'de örnek bir sunum içeriği incelenebilir.

Çizelge 3.17 Örnek Bir Sunum İçeriği (NEMA, 2008)

MR Görüntü Depolama	1.2.840.10008.5.1.4.1.1.4	Implicit VR Little Endian	1.2.840.10008.1.2	SCU
		DICOM JPEG-LS kayıpsız sıkıştırma	1.2.840.10008.1.2.4.80	
		DICOM JPEG2000 kayıplı sıkıştırma	1.2.840.10008.1.2.4.91	

Sunum içerikleri DICOM bağlantısının, isteyen (A-Associate-RQ) ve kabul eden (A-Associate-AC) olmak üzere iki farklı biçimi vardır. Karşı taraf A-Associate-RQ mesajından sunum içeriğini aldıktan sonra:

1. Önerilen soyut söz dizimini kendisinin sunup sunmadığını kontrol eder
2. Transfer söz dizimini seçer

Bu işlemleri yaptıktan sonra eğer karşı taraf ilişkiyi kabul ederse, seçilen söz dizimlerini içeren kısa bir sunum içeriğini A-Associate-AC mesajı ile geri gönderir. İsteği gönderen taraf gelen cevabı aldıktan sonra:

1. A-Associate-RQ mesajı ile gönderilen sunum içeriği içerisindeki sunum içerik numaralarını, gelen sunum içerik numarası ile karşılaştırır. Bu numaraların ilişki boyunca tekil olması beklenir.

2. A-Associate-AC mesajını alan taraf, sunum içeriğinin içerisinden bağlantının durumunu (kabul edildi, reddedildi gibi) öğrenir.

3.7.7. Kullanıcı bilgisi

Kullanıcı bilgisinde geçen kullanıcı aslında el sıkışma isteğinde bulunan AE'yi ifade eder. Kullanıcı bilgisi, özel iletişim parametrelerini içermektedir. Bir kullanıcı bilgisinin içerdiği elemanlar (Pianykh, 2008):

1. *Maksimum uzunluk*: Ağ üzerinden gönderilecek verinin byte cinsinden maksimum uzunluğunu belirler.
2. *Gerçekleştirim tanımlaması*: DICOM yazılımının sürümü hakkında bilgi veren sınıf tekil numaraları (Class UID) ve seçimlik olarak sürüm ismi.
3. *Eş zamansız işlem müzakeresi*: Tüm komutların eş zamanlı ya da eş zamansız olarak çalıştırılacağını ifade eder.
4. *SCP/SCU rol seçim müzakeresi*: Her iki AE'nin, SCU (isteği yapan) veya SCP (isteklere cevap veren) rollerinden birini seçmesini sağlar.
5. *Uzatılmış özellik müzakeresi*: DICOM standardı tarafından tanımlanmayan özellikler bu eleman ile müzakere edilir.

3.7.8. Protokol veri birimi

DICOM komut nesneleri gibi ilişki komutlarını taşımak için tasarlanan ilişki yönetim yapıları vardır. Bu yapılar DICOM veri alışverişi protokolünün temeli oldukları için, protokol veri birimi (Protocol Data Unit – PDU) olarak adlandırılır. DICOM içerisinde tanımlı PDU'lar (NEMA, 2008):

- A-Associate-RQ PDU
- A-Associate-AC PDU
- A-Associate-RJ PDU
- P-Data-TF PDU
- A-Release-RQ PDU
- A-Release-RP PDU
- A-Abort PDU

AE'ler arası ilişki A-Associate-RQ ile açılır, veriler P-Data-TF yığınları ile gönderilir ve A-Release-RQ mesajı ile ilişki sonlandırılır.

Bir DICOM bağlantısını başlatabilmek için ilişkiyi kurmak isteyen AE, belirli bağlantı parametrelerini içeren bir A-Associate-RQ mesajını gönderir. Alıcı parametreleri kabul ederse, A-Associate-AC mesajı ile bağlantıyı kabul eder. Eğer alıcı kendisine sunulan parametreleri kabul etmezse, bağlantıyı A-Associate-RJ mesajı ile reddeder.

3.7.8.1. A-Associate-RQ

A-Associate-RQ, bir DICOM ilişkisi kurabilmek için bir AE'nin diğerine gönderdiği bağlantı açma teklifidir. Gönderici ve alıcının AE isimleri boş olmamalıdır ve DICOM ağı içerisinde o AE'ler var olmalıdır. Birçok görüntüleyici iş istasyonu ve PACS arşivi, kendisine gelen bağlantı isteklerini gönderen AE isimlerinin, ilk olarak kendi bünyesinde tuttuğu listenin içerisinde olup olmadığını kontrol eder. Listede istek yapan AE bulunmuyorsa, ilişki A-Associate-RJ mesajı ile reddedilir.

3.7.8.2. A-Associate-AC

A-Associate-RQ mesajı ile başlatılmak istenen ilişki kabul edilirse, cevaben A-Associate-AC mesajı alınır. Daha önce bahsedildiği gibi A-Associate-RQ mesajında birçok transfer söz dizimi olmasına karşın, A-Associate-AC mesajı içerisinde sadece tek bir transfer söz dizimi bulunur.

3.7.8.3. A-Associate-RJ

A-Associate-RJ mesajı, teklif edilen ilişki istek mesajını (A-Associate-RQ) reddetmek için kullanılır. Desteklenmeyen transfer protokolleri, geçersiz biçim gibi sebeplerden ötürü teklif reddedilebilir.

3.7.8.4. A-Abort

A-Abort mesajı, A-Associate-RJ mesajı ile temelde aynı işleve sahiptir. A-Associate-RJ mesajı, sadece ilişki kurulma aşamasında gönderilebilirken; A-Abort mesajı, ilişkinin herhangi bir safhasında gönderilebilir. İlişkinin anormal bir şekilde sonlandırıldığını ifade eder.

3.7.8.5. A-Release-RQ ve A-Release-RP

A-Release-RQ ve A-Release-RP mesajları, herhangi bir hata oluşmadan devam eden bir ilişkiyi başarılı bir şekilde sonlandırmaya olanak sağlar. Veri aktarımı bittiğinde ilişkinin sonlandırılması için veriyi gönderen AE, A-Release-RQ mesajını karşı tarafa iletir. Bu mesajı alan taraf A-Release-RP mesajı ile cevap vererek iki AE arasındaki ilişki sonlandırılmış olur.

3.7.8.6. P-Data-TF

P-Data-TF, diğer PDU tiplerinden oldukça farklıdır. Verinin taşınması görevini üstlenir. DICOM nesnelerini protokol veri birimi (Protocol Data Value – PDV) ismi verilen küçük parçalar halinde karşı tarafa iletir.

4. KULLANILAN ARAÇ VE TEKNOLOJİLER

4.1. dcm4che2

dcm4che2, DICOM standardının Java programlama dili kullanılarak açık kaynak kodlu olarak gerçekleştirimidir. Yazılımın 1.x sürümü mimari olarak iyileştirilerek daha hızlı, daha az bellek kullanan, daha basit ve daha sağlam bir gerçekleştirim haline gelmiş ve 2.x sürümü olarak adlandırılmıştır (dcm4che2, 2009). Tez projesi kapsamında geliştirilen dağıtık PACS sunucusunda DICOM iletişimini sağlamak için dcm4che2 kütüphanesi kullanılmıştır.

Dağıtık PACS sunucusunun geliştirilme aşamasında dcm4che2 kütüphanesinde bulunan bazı eksiklikler ve hatalar geliştiricilere bildirilmiş ve bu sorunları giderecek yamalar geliştirilerek projeye destek sağlanmıştır.

Dağıtık PACS uygulaması dcm4che2 2.0.20 sürümünün yamalı hali ile geliştirilmiştir. Bu yamalar bir sonraki sürüme ekleneceği için sunucu gelecek sürüm olan 2.0.21 ile tam uyumludur.

DICOM, nesneye dayalı olarak geliştirilen bir standarttır. dcm4che2 kütüphanesi de nesneye dayalı olarak geliştirilmiş ve hem standardın hem de Java'nın tüm nimetlerinden faydalanmıştır. Kısacası, dcm4che2 tam anlamıyla DICOM standardının bir gerçekleştirimidir. Bu yüzden standart ile aşina olan geliştiriciler dcm4che2'yi kolaylıkla anlayabilir.

Standartta adı geçen birçok nesne ve bu nesnelerin metod ve özellikleri doğrudan kütüphaneye uyarlanmış ve gerçekleştirilmiştir. Örneğin;

- *Device*: DICOM iletişim yeteneği olan cihaz
- *Association*: İki AE arasındaki ilişki
- *DicomObject*: Görüntü ya da komut gibi bir veriyi tutan DICOM nesnesi
- *NetworkApplicationEntity*: Uygulama birimi (AE)
- *NetworkConnection*: İki AE arasındaki ağ bağlantısı
- *Tag*: DICOM veri sözlüğünün yazılıma uyarlanmasında kullanılan "(grup, eleman)" çiftini tanımlayan etiket
- *StorageService*: C-Store hizmetini sunan sınıf
- *VerificationService*: C-Echo hizmetini sunan sınıf
- *DimseRSP*: DIMSE cevabı

Yukarıdaki sınıf ve nesneler gibi DICOM standardında tanımlı birçok bileşen kütüphane içerisinde gerçekleştirilmiştir.

dcm4che2 ile geliştirilen örnek bir C-Echo SCU uygulaması aşağıdaki gibidir:

```
public class SimpleEchoSCU {
	private Device device = new Device("ECHO-DEVICE");
	private Executor executor = new NewThreadExecutor("ECHO-
THREAD");;

	private NetworkConnection localConnection = new
NetworkConnection();

	private NetworkConnection remoteConnection = new
NetworkConnection();

	private ApplicationEntity localAE = new ApplicationEntity();
	private ApplicationEntity remoteAE = new
ApplicationEntity();

	public SimpleEchoSCU() {
		device.setNetworkApplicationEntity(localAE);
		device.setNetworkConnection(localConnection);

		localAE.setNetworkConnection(localConnection);
		localAE.setAssociationAcceptor(false);
		localAE.setAETitle("ECHOSCU");

		remoteAE.setInstalled(true);
		remoteAE.setAssociationAcceptor(true);
		remoteAE.setNetworkConnection(new NetworkConnection[]{
remoteConnection });
	}

	public void cecho(String aeTitle, String hostname, int
port){
		configureRemoteHost(aeTitle, hostname, port);

		try {
			Association assoc = localAE.connect(remoteAE,
executor);

			assoc.cecho().next();

			assoc.release(true);
```

```
            } catch (InterruptedException e) {
                  e.printStackTrace();
            } catch (IOException e) {
                  e.printStackTrace();
            } catch (ConfigurationException e) {
                  e.printStackTrace();
            }
      }

      private void configureRemoteHost(String aeTitle, String
hostname, int port){
            remoteAE.setAETitle(aeTitle);
            remoteConnection.setHostname(hostname);
            remoteConnection.setPort(port);
      }

      public static void main(String[] args) {
            SimpleEchoSCU echoSCU = new SimpleEchoSCU();

            echoSCU.cecho("ECHOSCP", "192.168.1.100", 104);
      }
}
```

SimpleEchoSCU sınıfının yapılandırıcı metodunda, nesneler arası gerekli ilişkiler kurulur ve bu nesnelerin AE ismi gibi bazı özellikleri ayarlanır. cecho() metoduna gelen uzak AE'nin erişim bilgileri, configureRemoteHost() metodu ile ilgili nesnelerin özelliklerine atanır. Ardından iki AE arasında Association sınıfının connect() metodu ile bağlantı kurulur ve cecho() metodu ile C-Echo-Rq mesajı gönderilir. Son olarak Association sınıfının release() metodu ile iki AE arasında kurulan ilişki sonlandırılır.

4.2. Uzak Metot Çağrımı

Java Uzak Metot Çağrımı (Remote Method Invocation - RMI) basit bir model sunarak programcının Java ile dağıtık nesneler yazmasına olanak sağlamaktadır. Uzaktaki bir bilgisayarda halihazırda çalışan bir nesnenin metotlarını çalıştırıp bir görevi yerine getiren tekniğe Java'da uzak metot çağrımı ismi verilmiştir (Sun Microsystems, 2009).

RMI, Java tabanlı bir teknoloji olduğu için Java'nın sunduğu üstün güvenlik ve platform bağımsızlığı kavramlarını dağıtık programlamaya taşımaktadır.

4.2.1. RMI kullanımının avantajları

RMI, Java'nın uzak prosedür çağırımı (Remote Procedure Call - RPC) mekanizmasıdır. Geleneksel RPC sistemlerine göre RMI, Java'nın nesneye yönelik yaklaşımının bir parçası olduğu için bir çok avantaj içermektedir.

RMI teknolojisinin temel avantajları şunlardır (Sun Microsystems, 2009):

- *Nesneye dayalı*: RMI sadece önceden tanımlanmış veri tiplerini değil, nesneleri de parametre ve geri dönüş değerleri olarak iletebilir. Bunun anlamı Java'da halihazırda mevcut olan Hashtable tipindeki nesnenin bile basit bir parametre olarak metotlar arası iletilebileceğidir. Mevcut RPC sistemlerinde bu işlemin yapılabilmesi için nesne, istemci tarafında ilkel veri tiplerine dönüştürülmeli bu şekilde iletilmeli ve sunucu makine de nesne, yeniden yaratılmalıdır. RMI, nesnelerin ekstra bir kod kullanılmadan doğrudan iletimini sağlamaktadır.
- *Taşınabilirlik*: RMI sınıf gerçekleştirimlerini istemciden sunucuya ve sunucudan istemciye taşıyabilmektedir. Örneğin, çalışanların masraf raporlarını inceleyip şu anki şirket politikalarına uygun davranıp davranmadıklarını anlayabilmek için bir arayüz tasarlanmıştır. Bir masraf raporu yaratıldığı an, bu arayüzü gerçekleştiren bir nesne, istemci tarafından sunucudan alınabilmektedir. Şirket kuralları değiştiği zaman, sunucu bu arayüzün yeni kuralları uygulayan farklı bir gerçekleştirimini döndürmeye başlayacaktır. Kullanıcı sistemlerine hiç bir yeni yazılım yüklenmeyecektir, sadece sunucu tarafa yeni kuralları içeren bir Java sınıfı yazılacak ve sisteme dahil edilecektir. Böylelikle maksimum esneklik sağlanmaktadır.
- *Tasarım desenleri (Design Patterns)*: Nesnelerin iletilebilmesi, dağıtık programlamada nesneye yönelik yazılım geliştirme yönteminin tüm gücünün kullanılmasına olanak sağlamaktadır. Bunlar arasında nesneye yönelik tasarım desenleri de yer almaktadır.
- *Güvenilirlik*: Sistem güvenliğini sağlamak için Java güvenlik mekanizmasını kullanmaktadır.
- *Kolaylık*: Uzak Java sunucuları ve bunlara bağlanan Java istemcilerin kodlamasını kolaylaştırmaktadır. Uzak arayüz tam anlamıyla bir Java arayüzüdür. Normal bir Java nesnesinin sunucu olmasını sağlamak oldukça basittir. Böylelikle tam dengeli, dağıtık nesne sistemleri için sunucu yazımı karmaşıklıktan kurtulmaktadır. Ayrıca prototiplerin

kurulması, yazılımın geliştirilmesi ve test edilmesi oldukça hızlıdır. RMI programlarının, yazımı kadar sürdürülebilirlikleri ve geliştirilebilirlikleri de oldukça kolaydır.

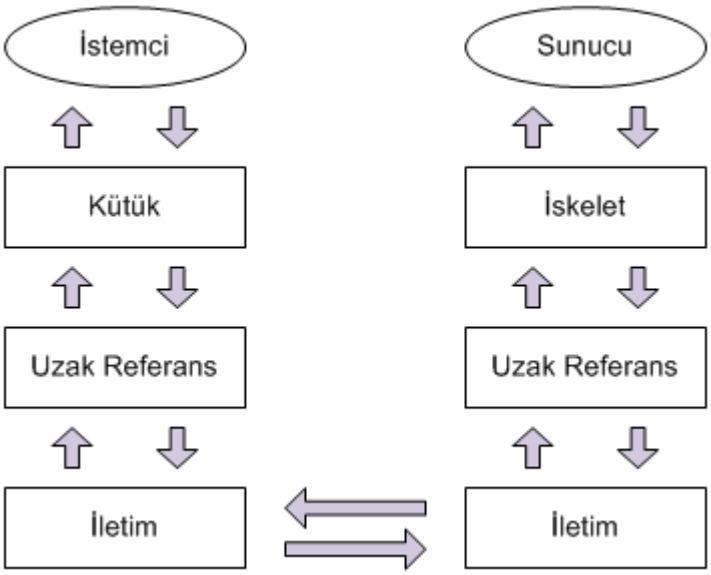
- *Kolay entegrasyon*: RMI, Java Native Interface (JNI) aracılığı ile mevcut sistemler ile iletişim kurabilmektedir. RMI ve JNI kullanılarak istemci, Java ile yazılabilmekte ve mevcut sunucu uygulaması ile iletişim kurabilmektedir. Benzer şekilde RMI, JDBC (Java Database Connectivity) aracılığı ile Java ile yazılmamış mevcut veritabanları ile konuşabilmektedir.
- *Yazılımın taşınabilirliği*: RMI, Java'nın " Bir kere yaz, her yerde çalıştır " yaklaşımına dayanmaktadır. Tüm RMI tabanlı sistemler tüm Java sanal makinelerine uyumludur.
- *Dağıtık çöp toplama (Garbage Collection)*: RMI, ağ üzerindeki hiç bir istemci içerisinde referansı tutulmayan uzak sunucu nesnelerini toplamak için dağıtık çöp toplama özelliğini kullanmaktadır.
- *Paralel işleme*: RMI, aynı anda oluşan kullanıcı isteklerine daha iyi cevap verebilmek için çoklu iş parçacıklarını desteklemekte ve işletmektedir.
- *Java dağıtık işleme çözümü*: Tüm RMI sistemleri aynı protokol üzerinden konuşurlar, yani tüm Java sistemleri protokol dönüşüm işlemleri gibi yükler yaratmadan birbirleri ile direk olarak iletişim kurabilmektedirler.

4.2.2. Java RMI mimarisi

Java RMI mimarisi üç katmandan oluşmaktadır. Bu katmanlar (Kara ve Üstündağ, 2002):

- Kütük (Stub) ve İskelet (Skeleton) Katmanı
- Uzak Referans (Remote Reference) Katmanı
- İletim (Transport) Katmanı

Şekil 4.1'de de görüldüğü gibi, uzakta bulunan bir Java nesnesinin metotlarını çağıran bir istemci ilk önce kütük ile iletişime geçer. Kütük, isteği uzak referans katmanına, o katman da iletim katmanına geçirir. Daha sonra, karşı taraftaki bilgisayarda istek önce uzak referansa, oradan da iskelete iletilerek istenen servise ulaşılır. Servisin sonuçları da, aynı biçimde istemciye iletilmektedir.



Şekil 4.1 İstemci-Sunucu Etkileşim Katmanları (Kara ve Üstündağ, 2002)

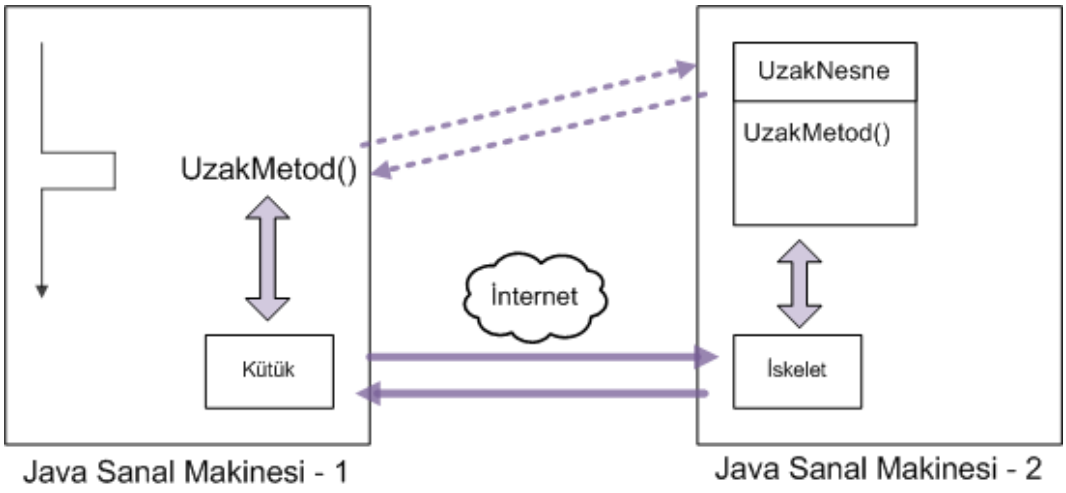
4.2.2.1. Kütük ve iskelet katmanı

İstemci, uzakta bulunan bir nesnenin metotlarını çağırmaya başladığı anda ilk olarak kütük katmanı ile iletişime geçecektir. İstemcinin uzaktaki nesne ile ilgili olarak elinde bulunan referans aslında yerel kütük koda olan referanstır. Karşı tarafta bulunan iskelet kod ise, uzak nesnenin istenen metodunun gerçekleştiriminin çağrılması ve sonuçların elde edilmesinden sorumludur.

Bir RMI çağrımında gerçekleşen olayların adımları ve bu adımlar içerisinde kütük ve iskelet kodların görevleri aşağıda verilmiştir (Kara ve Üstündağ, 2002).

- Kütük kodu; nesneleri serileştirerek isteği, parametreleri ile birlikte paketler.
- Kütük kodu, isteğin uzak nesneye gönderimini yapar. Bu istek uzak referans ve iletim katmanları yolu ile karşı tarafa aktarılır. Karşı taraftaki iletim ve uzak referans katmanları ile de iskelet koda ulaşır.
- Karşı taraftaki iskelet kodu gelen paketi açar, parametreleri çıkarır ve uzak nesnenin istenen metodunun gerçekleştirimini çağırır.
- İskelet kod sunucudaki nesneden gerçekleştirimin sonucunu alır.
- İskelet kod sonucu paketleyerek, isteğin geldiği yoldan benzer biçimde karşı tarafa gönderir.
- Sonuçlar, kütük koda gelir. Kütük kod, sonuçların bulunduğu paketi açar ve sonuçları istemciye geçirir.

Kütük ve iskelet katmanının diğer bir özelliği de, yazılan programların platform bağımsız olmasını sağlamasıdır. Bu katman kullanıcıyı, uzak nesnenin gerçekleştirim ayrıntılarını veya aktarım katmanı ayrıntılarını bilmekten kurtarır. Kütük ve iskelet kodlarının uzak nesne çağrımındaki rolü Şekil 4.2'de görülmektedir.



Şekil 4.2 Kütük ve İskelet Kod İlişkisi (Kara ve Üstündağ, 2002)

4.2.2.2. Uzak referans katmanı

Bu katman temel olarak programcının yazdığı uygulama ve bilgisayar ağı iletişimi arasında bir köprü görevi yapmaktadır.

Bu katmanın görevlerinden birisi, kütük kodundan gelen istekleri aktarım katmanı biçimine dönüştürmek veya aktarım katmanından gelen istekleri iskelet kodunun anlayacağı biçime dönüştürmektir.

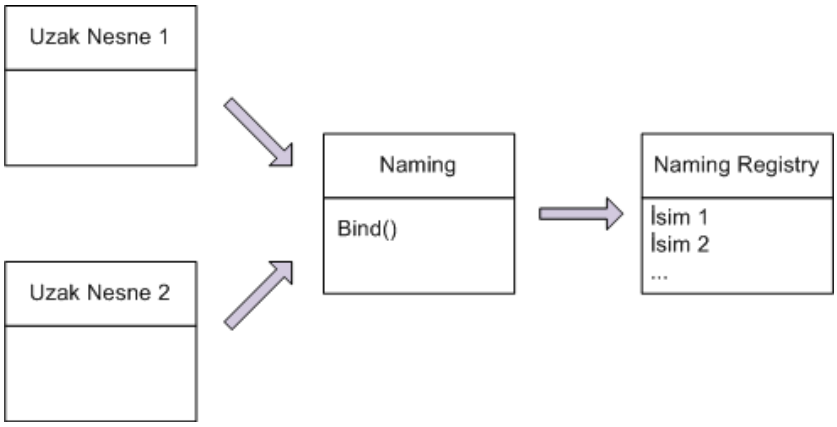
Bu katmanın diğer bir görevi, uzak referans protokollerini gerçekleştirmektir. Bu protokoller, noktadan noktaya (point to point) veya tekrarlı (replicated) nesnelere çağrım protokolleri olabilir.

4.2.2.3. İletim katmanı

Bu katman bağlantının sağlanması, bakımı, kapatılması gibi işlerden sorumludur. İletişimin sağlanmasında Java soketleri kullanılmaktadır. Aktarım protokolü olarak TCP kullanılmaktadır. RMI mimarisinde her katman birbirinden bağımsız olduğu için aktarım protokolünü değiştirmek olasıdır.

4.2.3. RMI isimlendirme servisi

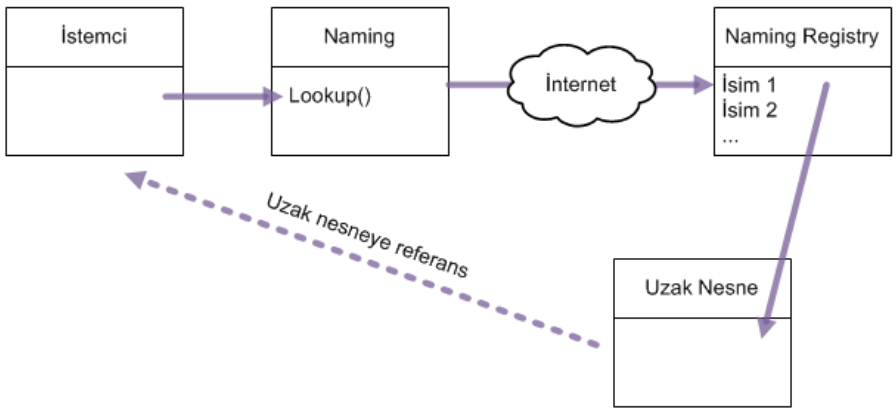
İstemcilerin uzaktaki nesnelerin metotlarını RMI ile çağrılabilmeleri için, söz konusu uzak nesnelere ilişkin referansları elde etmesi gerekmektedir. RMI modelinde bu amaca yönelik olarak bir isim sunucusu kullanılmaktadır. Uzak nesneler java.rmi.Naming sınıfının bind() metodunu kullanarak kendilerini İsimlendirme Kayıt (Naming Registry) servisine kaydettirmelidirler. Bu durum Şekil 4.3'de görülmektedir.



Şekil 4.3 Uzak Nesnelerin İsimlendirme Servisine Kayıt Edilmesi (Kara ve Üstündağ, 2002)

İsimlendirme servisi direk olarak yazılımdan veya JRE (Java Runtime Environment) ile gelen rmiregistry komutu uygulaması ile çalıştırılabilmektedir.

İstemciler de java.rmi.Naming sınıfının lookup() metodunu kullanarak uzak nesnelerin hangileri olduğuna bakıp istenen uzak nesneye bir referans elde edebilmektedir. İstemcinin uzak nesneye referans elde etmesi Şekil 4.4'de görülmektedir.



Şekil 4.4 Uzak Nesneye Referans Elde Edilmesi (Kara ve Üstündağ, 2002)

4.3. log4j

log4j, Apache bünyesinde Java programlama dili için geliştirilen açık kaynaklı bir kayıt tutma (logging) paketidir. log4j, harici yapılandırma dosyaları ile çalışma zamanında dahi yapılandırılabilmektedir (Apache, 2009).

Kayıt tutma, çoğunlukla yazılımlarda çalışma zamanında oluşabilecek hataları yakalayabilmek için yazılımın içerisindeki süreci bir dosya, veri tabanı tablosu ya da komut satırı gibi çeşitli ortamlarda kayıt altına alma işlemidir. Oluşan hataların kaynağı bu ortamlardan elde edilecek kayıtlar ile kolayca bulunabilir.

Ayrıca hata ayıklayıcıların (debugger) kullanılamadığı durumlarda da çok faydalıdır. Dağıtık uygulamalar için de çoğunlukla bu durum söz konusudur.

Tüm iyi yanlarına karşın kayıt tutma, ek kod gerektirdiği için kullanıldığı yazılımı yavaşlatabilmektedir. Özellikle Java programlama dilinde ön işlemci bulunmadığı için kayıt tutma özelliği yapılandırma dosyaları ile kapatılmış olsa dahi kodun boyutunu artırır ve uygulamanın hızını yavaşlatır (Apache, 2009).

log4j'de çeşitli kayıt seviyeleri ve mesajları bulunur. Çizelge 4.1'de bu seviyeler açıklamaları ile birlikte önem seviyesine göre sıralı olarak verilmiştir.

Çizelge 4.1 log4j'de Kayıt Seviyeleri (Wikipedia, 2009c)

Seviye	**Açıklama**
FATAL	Programın sonlandırılmasına sebep olabilecek hatalar
ERROR	Diğer çalışma zamanında oluşabilecek hatalar veya beklenmedik durumlar
WARN	API'nin artık kullanılmayan özelliklerini kullanmak, gerektiği gibi kullanmamak, hata oluşturmaya yakın şeyler, çalışma zamanında oluşabilecek yanlış olmayan fakat istenmeyen ya da beklenmeyen şeyler
INFO	Çalışma zamanında oluşan önemli olaylar. Özellikle yazılımın açılması ve kapanmasında oluşan olaylarda kullanılır
DEBUG	Sistemin akışı hakkında detaylı bilgiler
TRACE	DEBUG ile kaydedilenlerden daha detaylı bilgiler

Bu seviyeler log4j içerisinde bulunan Logger sınıfındaki şu metotlar ile kodlanır (Gülcü, 2002):

- trace()
- debug()
- info()
- warn()
- error()
- fatal()

Aşağıda log4j ile nasıl kayıt tutulabileceğini gösteren bir kod örneği verilmiştir.

```
public class LoggingSample {
      static Logger logger =
Logger.getLogger(LoggingSample.class);

      public void print(){
            logger.debug("write() method is executed");
      }

      public static void main(String[] args) {
            BasicConfigurator.configure();

            logger.info("Starting application");
            LoggingSample sample = new LoggingSample();
            sample.print();
            logger.info("Stopping application");
      }
}
```

Bu örnekte yapılandırma işlemi, BasicConfigurator sınıfı ile kod içerisinde yapılmaktadır. BasicConfigurator yapılandırması ile kayıtlar komut satırına belirli bir formatta yazılmaktadır. Örneğin komut satırına yazılan çıktısı şu şekildedir:

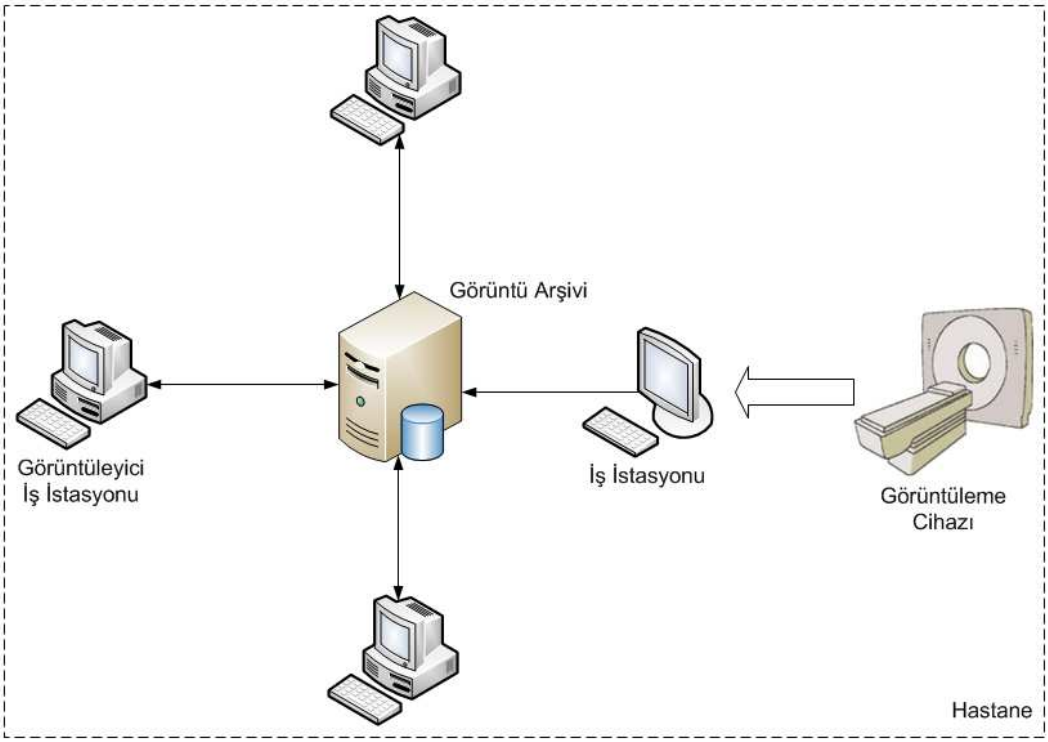
```
0 [main] INFO  LoggingSample  - Starting application
0 [main] DEBUG LoggingSample  - write() method is executed
0 [main] INFO  LoggingSample  - Stopping application
```

5. GERÇEKLEŞTİRİLEN SİSTEM

Bu bölümde ilk olarak, tez projesi kapsamında gerçekleştirilen uygulamanın yazılım ve sistem mimarisi anlatılacaktır. Ardından mimari, gerçekleştirim aşamasında kullanılan akış diyagramları ve kaynak kodlar ile detaylandırılacak ve bölüm sonunda örnek bir senaryo ile sistemin işleyişinin pekiştirilmesi sağlanacaktır.

5.1. Mimari

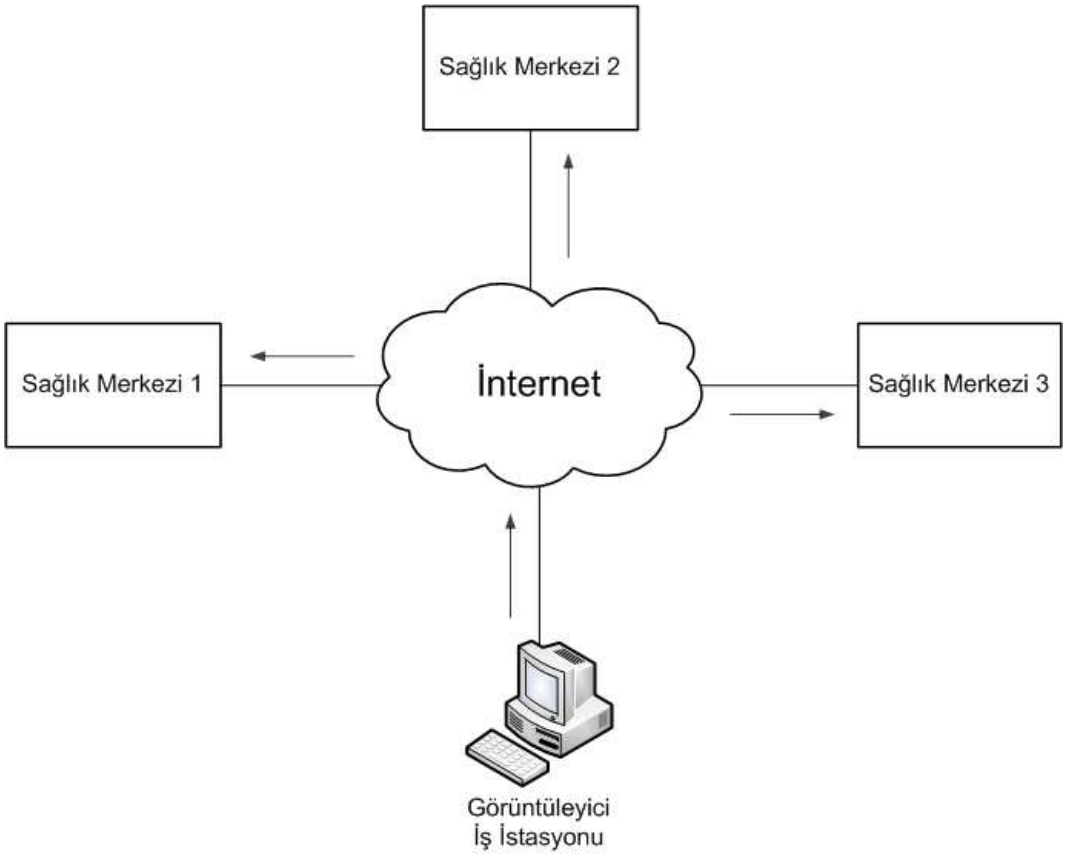
PACS bölümünde anlatıldığı üzere, görüntüleme cihazları ve iş istasyonları PACS sunucusuna ağ üzerinden bağlıdırlar. Görüntüleme cihazında çekilen görüntüler radyoloji teknisyenleri tarafından düzenlendikten (parlaklık ve netlik ayarı, görüntüyü kırpma gibi) sonra PACS sunucusuna gönderilir. Kullanıcı (doktor veya radyolog) incelemek istediği görüntüyü iş istasyonları üzerinde bulunan görüntüleyiciler vasıtasıyla sorgular. Sorgulama işlemini hasta adı, numarası, çekim tarihi gibi arama kriterlerini girerek gerçekleştirir. PACS sunucusu kullanıcıya sonuç listesi döndürür. Kullanıcı bu listeden istediği sonucu seçerek görüntüleyici yardımıyla arşivden görüntüyü çeker. Bu süreci yerine getirebilen genel bir PACS'ın görünümü Şekil 5.1'de verilmiştir.



Şekil 5.1 Genel Bir PACS Görünümü

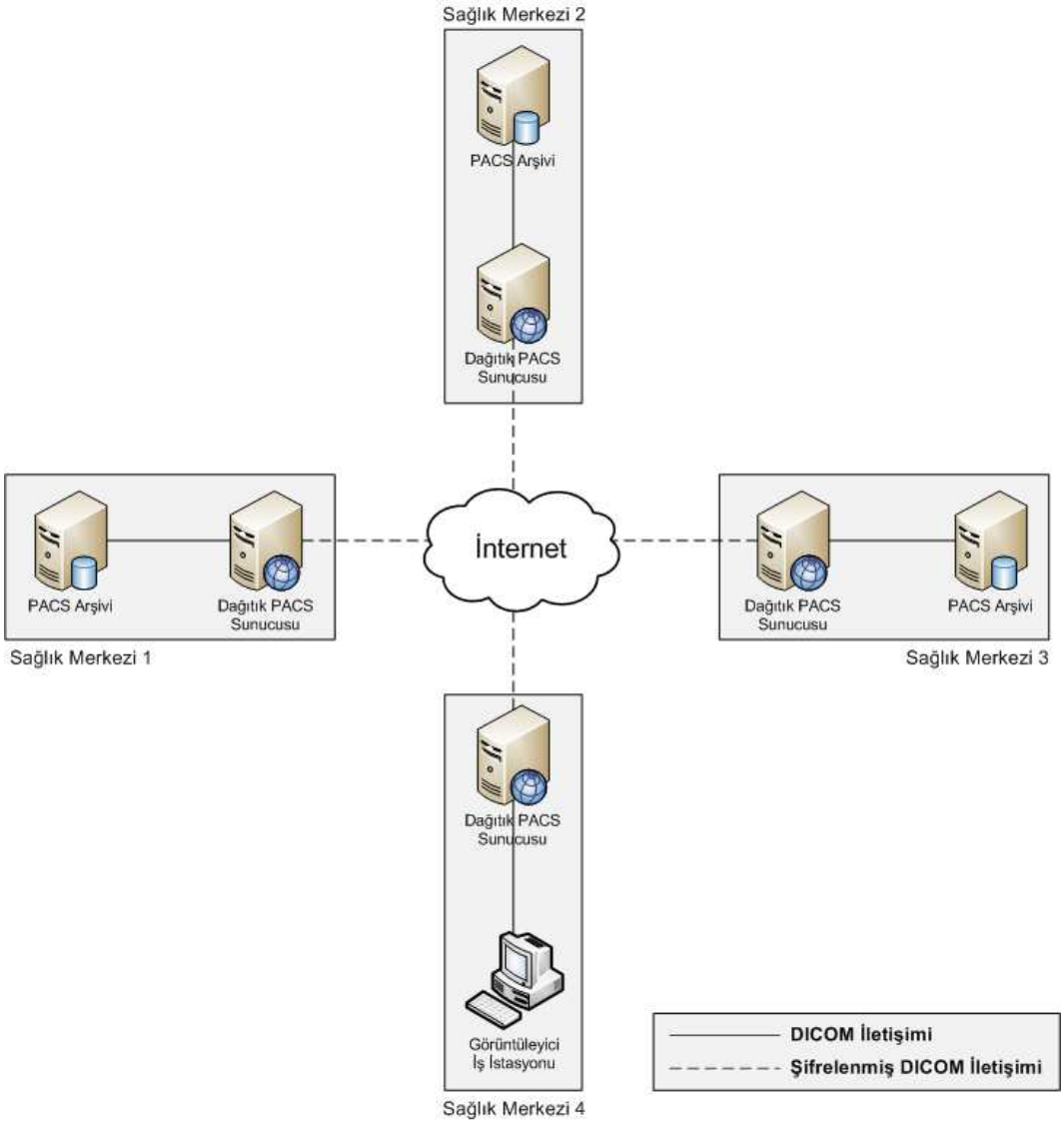
5.1.1. Sistem mimarisi

Dağıtık PACS sistemi ile coğrafi olarak birbirinden farklı yerlerde bulunan sağlık merkezlerindeki PACS sunucularını, İnternet ile birbirine bağlayarak tek bir sorgu ile tüm arşivleri sorgulayabilmek mümkün olacaktır. Örnek olarak Şekil 5.2'de görüntüleyici ve 3 sağlık merkezinin bağlı olduğu dağıtık sistem gösterilmiştir.



Şekil 5.2 Dağıtık PACS'ın Genel Görünümü

Dağıtık sistemi meydana getiren bileşenler, birbirleri arasındaki iletişimi tez çalışması sırasında geliştirilen yazılım sayesinde yaparlar. Şekil 5.3'te bileşenler arasındaki bağ daha ayrıntılı bir şekilde verilmiştir.



Şekil 5.3 Dağıtık PACS Sistemi İçerisindeki Bileşenler Arası Bağlantı

Şekilde görüldüğü gibi, sağlık merkezleri dağıtık sisteme dahil olabilmek için dağıtık PACS sunucularını kullanırlar. Sağlık merkezlerindeki görüntüleyici iş istasyonları ve PACS sunucuları, dağıtık PACS sunucuları ile DICOM standardını kullanarak iletişim kurarlar. Bu sayede sağlık merkezlerinde halihazırda var olan DICOM uyumlu PACS sunucuları ve görüntüleyici iş istasyonları sisteme kolaylıkla entegre olabilirler.

PACS sunucuları ve görüntüleyici iş istasyonları dağıtık PACS sunucusuna bir yapılandırma dosyası ile tanıtılırlar. Bu işlemden sonra her istasyon sisteme dahil edilen tüm sağlık merkezlerindeki sunuculara sorgu gönderebilir. Uygulamaya gelen sorgular ise; sağlık merkezindeki kayıtlı olan tüm PACS

sunucularına gönderilir. Bu süreç, sonraki bölümlerde detaylı olarak ele alınacaktır.

5.1.2. Yazılım mimarisi

Yazılım, 3 farklı proje olarak geliştirilmiştir. Bunlar:

- Dağıtık PACS sunucusu (dipacs)
- İsim sunucusu (dipacs-nameserver)
- Ortak bileşenler (dipacs-common)

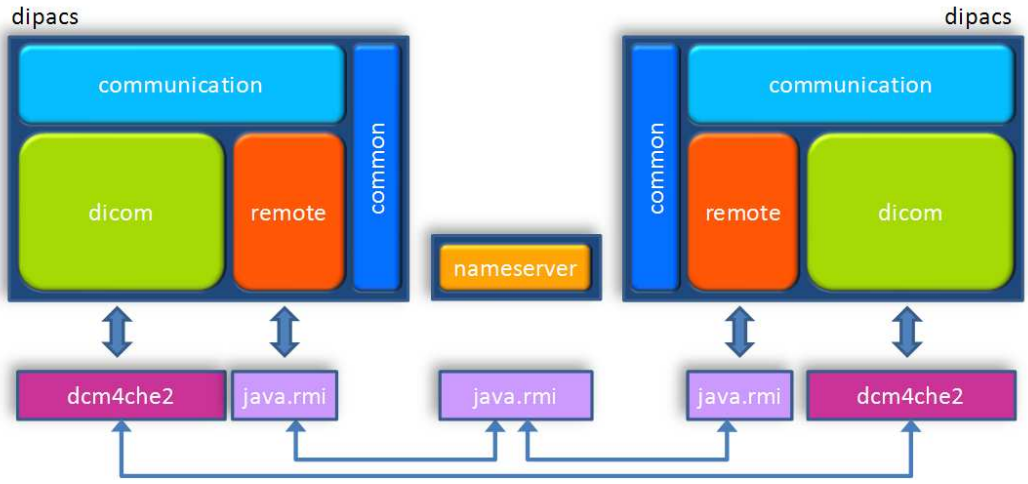
Dağıtık PACS sunucusu; görüntüleyici iş istasyonlarından gelen sorguları, diğer sağlık merkezlerinde bulunan uygulamalara gönderir. Dış uygulamalardan gelen istekleri ise, sağlık merkezi içerisinde bulunan PACS sunucularına yönlendirir.

İsim sunucusu, sisteme dahil olan tüm dağıtık PACS sunucularının erişim bilgilerini tutar. Dağıtık PACS sunucuları, isim sunucusundan diğer sunucuların adres bilgilerini öğrenirler. İsim sunucusunun en önemli özelliği, dağıtık sisteme yetkisiz girmeye izin vermemesidir. Dağıtık PACS sunucuları kendisine gelen sorguların göndericisini isim sunucusundan aldığı adres bilgileri ile karşılaştırarak güvenliği sağlar.

Ortak bileşenler, uygulama ve isim sunucusunun kullandığı ortak sınıf ve arayüzleri içerisinde barındırır.

Projelerin birbirleri ile iletişimi ve paketlerin iletişimde kullanılan kütüphanelerle ilişkisi Şekil 5.4'de verilmiştir.

dicom paketi dcm4che2 kütüphanesini kullanarak, uygulamalar ve PACS bileşenleri arasında DICOM iletişimini sağlar. Gelen DICOM mesajlarını, dcm4che2 sınıf ve metotlarını kullanarak communication paketine iletir. communication paketi, gelen mesajların içeriği doğrultusunda gerekli işlemleri yapar ve oluşturduğu mesajları dicom paketine gönderir. dicom paketi communication'dan gelen mesajları dcm4che2 kütüphanesinin yardımıyla diğer sunuculara iletir.



Şekil 5.4 Dağıtık PACS ve İsim Sunucusunun Bileşenleri ve İletişimi

Ayrıca communication paketi isim sunucusundan tüm uygulamaların erişim bilgilerini döndürmek için remote paketi ile iletişime geçer. remote paketi gelen isteği java.rmi paketini kullanarak isim sunucusuna iletir. İsim sunucusunda nameserver paketine gelen istek işlenerek isteği yapan sunucuya sonuç geri döndürülür.

5.2. Gerçekleştirim

Geliştirilen dağıtık sistem farklı bilgisayarlarda çalışacağı için, farklı platformları desteklemesi gerekmektedir. Bu nedenle sistem geliştirilirken Java programlama dilinin kullanılması uygun görülmüştür. Platformlar arası taşınabilir olmasının yanı sıra geliştirilen sistem için gerekli olan güvenlik, RMI gibi Java'nın birçok özelliğinden faydalanılmıştır.

Dağıtık sunucuda DICOM iletişimini sağlamak için DICOM standardına uyumlu ve açık kaynak kodlu dcm4che2 kütüphanesi kullanılmıştır.

Sistemin nasıl çalıştığını izlemek (tracing) ve oluşabilecek hataları yakalayabilmek (debugging) için log4j isimli loglama kütüphanesi ile birlikte SLF4J kullanılmıştır.

5.2.1. Dağıtık PACS sunucusu

Uygulamalar nesneye dayalı tasarım yöntemi ve tasarım desenleri kullanılarak geliştirilmiştir. Bu yöntemler kullanılarak gerçekleştirilen sistemin

UML (Unified Modelling Language) sınıf diyagramı Ek 1'de görülebilir. Sınıf diyagramında bulunan sınıfların etkileşimi daha sonraki bölümlerde anlatılacaktır.

5.2.1.1. İletişim kurulacak sistemlerin dağıtık sunucuya tanıtılması

Sağlık merkezi içerisinde bulunan görüntüleyici iş istasyonları ve PACS sunucuları ile birlikte, diğer dağıtık PACS sunucularının erişim bilgilerinin iletişime başlanmadan önce sisteme tanıtılması gerekmektedir. Daha önce bahsedildiği gibi, sağlık merkezi içerisinde bulunan görüntüleyiciler ve PACS sunucularının erişim bilgileri bir yapılandırma dosyası ile sisteme kaydedilir. configuration.xml isimli bu yapılandırma dosyasına aşağıdaki örnek verilebilir.

```
<Configuration>
      <Servers>
            <Server>
                  <AETitle>DCM4CHEE</AETitle>
                  <Hostname>192.168.1.100</Hostname>
                  <Port>104</Port>
            </Server>
      </Servers>
      <Workstations>
            <Workstation>
                  <AETitle>DOKTOR1</AETitle>
                  <Hostname>192.168.1.51</Hostname>
                  <Port>104</Port>
            </Workstation>
      </Workstations>
</Configuration>
```

Bu yapılandırma dosyasında bir sunucu ve bir iş istasyonu için erişim bilgileri bulunmaktadır. Erişim bilgileri arasında sistemin uygulama birimi ismi (AE title), IP adresi veya alan adı ve bağlantı noktası (port numarası) bulunmaktadır.

Yapılandırma dosyası ConfigurationParser sınıfı ile ayrıştırıldıktan sonra, ServiceController sınıfı içerisindeki lanAE nesnesinin içerisine HostInformation vektörü olarak atılır. HostInformation sınıfının detayları Ek 3'de incelenebilir. Bu sayede sağlık merkezi içerisinden gelen mesajları gönderen sistemlerin, dağıtık sunucuya tanıtılıp tanıtılmadığı kontrol edilir. Bu süreci gerçekleştiren kod parçacığı aşağıdaki gibidir.

```
ConfigurationParser parser = new
ConfigurationParser("configuration.xml");

lanAE.addHosts(parser.getApplicationEntities(AEType.WORKSTATION),
AEType.WORKSTATION);

lanAE.addHosts(parser.getApplicationEntities(AEType.SERVER),
AEType.SERVER);
```

Ayrıca diğer dağıtık sunuculardan gelen mesajların hangi sisteme yönlendirileceği de ApplicationEntity sınıfına eklenen HostInformation vektörü ile belirlenir.

Dağıtık sunucuların, diğer dağıtık sunucular ile iletişime geçebilmesi için erişim bilgilerini bilmesi gerekmektedir. Bu sunucuların erişim bilgileri, daha sonra da inceleneceği gibi, bir RMI sunucusundan okunmaktadır. RMI sunucusundan HostInformation vektörü olarak okunan dağıtık sunucuların erişim bilgileri wanAE isimli nesneye aşağıdaki gibi eklenir.

```
NameServerClient nsClient = new NameServerClient();
wanAE.addHosts(nsClient.getRemoteServers(), AEType.SERVER);
```

NameServerClient sınıfında bulunan getRemoteServers() metodunun içeriği aşağıda verilmiştir.

```
public Vector<HostInformation> getRemoteServers(){
      Vector<HostInformation> remoteServers = null;

      try {
            Registry registry =
LocateRegistry.getRegistry("nameserver.dipacs.org");

            NameServer stub = (NameServer)
registry.lookup("PacsNameServer");

            remoteServers = stub.getRemoteServers();
      } catch (Exception e) {
            e.printStackTrace();
      }

      return remoteServers;
}
```

Registry nesnesi, nameserver.dipacs.org alan adına bağlantı kurularak çağırılır. Ardından PacsNameServer isimli uzak referans çekilerek, NameServer arayüzüne çevrilir. Daha sonra elde edilen uzak nesnenin getRemoteServers() metodu çağırılarak, dağıtık PACS sunucularının erişim bilgileri elde edilmiş olur.

RMI sunucusundan okunan dağıtık sunucuların erişim bilgileri, mesajların hangi sunuculara yönlendirileceğinde kullanılır.

5.2.1.2. Sunulan Servislerin Kaydedilmesi

Dağıtık PACS sunucusunun DICOM'daki C-Echo, C-Find, C-Move ve C-Store servislerini (SCP servislerini) istemcilere sunabilmesi için sunucunun dcm4che2 içerisinde tanımlanan NetworkApplicationEntity sınıfına, bu servisleri kaydettirmesi gerekir. NetworkApplicationEntity sınıfına bazı ek özellikler kazandırılarak ApplicationEntity sınıfı türetilmiştir.

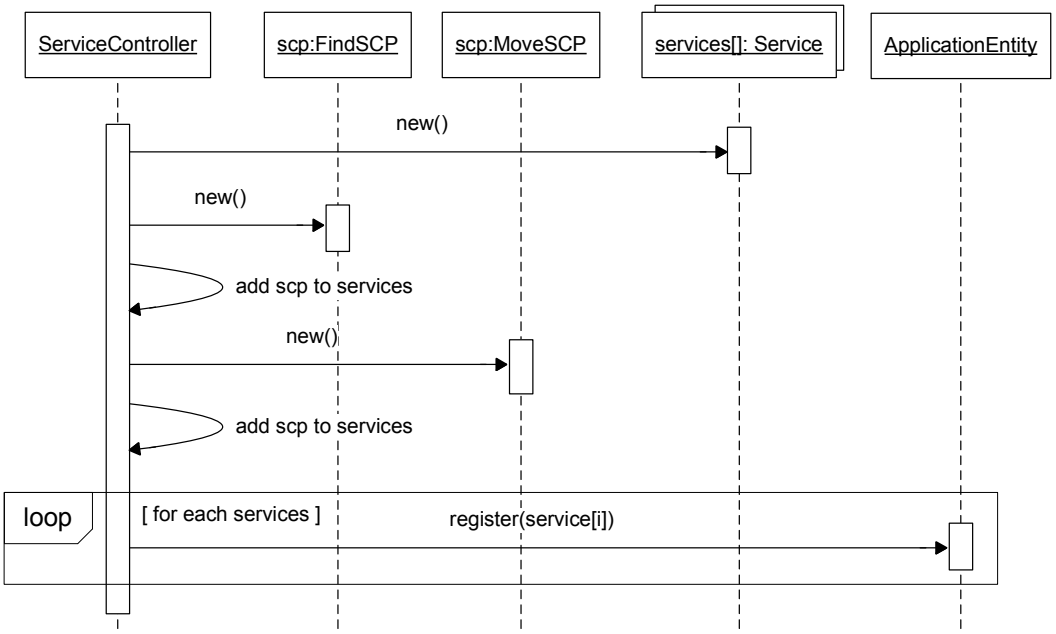
Geliştirilen sistemde, servislerin ilk yapılandırmasını ve kayıt işlemini ServiceController isimli sınıf “dicom” ve “communication” paketlerine erişim sağlayarak gerçekleştirir. Sınıf kütüphanelerine tek bir yerden erişebilmek ve bu erişimi kolaylaştırmak için kullanılan *Facade* tasarım deseni ile geliştirilen ServiceController sınıfının tamamı Ek 2'de görülebilir. Ayrıca dağıtık sunucu, sağlık merkezi içerisinde bulunan görüntüleyici ve sunucular ile iletişim kurabilmek ve diğer dağıtık sunucular ile iletişime geçebilmek için, iki ayrı bağlantı noktası kullanır. Bu bağlantıların her ikisinde de iletişim yöntemi olarak DICOM standardı kullanılır. Böylece sağlık merkezindeki istasyonlardan gelen DICOM mesajlarını, diğer dağıtık sunuculara gönderebilmek için farklı bir biçime dönüştürmeye gerek kalmaz. Bir SCP servisinden gelen mesaj, aynı tipteki SCU üzerinden alıcıya (görüntüleyici, PACS sunucusu veya bir dağıtık PACS sunucusuna) gönderilir. Kısacası servislerin (SCP'lerin) iki ayrı bağlantı noktası (port numarası) olması ve ApplicationEntity sınıfına tanıtılması gerekmektedir.

Aşağıda bağlantı noktalarını kaydetmede kullanılacak nesne ve nesne dizileri verilmiştir.

```
private static ApplicationEntity lanAE = new ApplicationEntity();
private static ApplicationEntity wanAE = new ApplicationEntity();

private static final Service[] lanServices = new Service[] { new
EchoSCP(lanAE) , new FindSCP(lanAE, wanAE), new StoreSCP(lanAE),
(Service) new MoveSCP(lanAE, wanAE)};
```

```
private static final Service[] wanServices = new Service[] { new
EchoSCP(wanAE) , new FindSCP(wanAE, lanAE), new StoreSCP(wanAE),
(Service) new MoveSCP(wanAE, lanAE)};
```

Yukarıda verilen lanAE nesnesi dağıtık sunucunun sağlık merkezi içerisinde bulunan birimlerle olan iletişimde kullanılırken, wanAE nesnesi ise diğer dağıtık sunucular ile iletişimde kullanılır. lanServices ve wanServices dizileri ise, sunulan servislerin ApplicationEntity nesnesine kaydolmasında kullanılır. Şekil 5.5'te servislerin ApplicationEntity sınıfına kaydedilmesini gösteren ardıl etkileşim diyagramı verilmiştir.



Şekil 5.5 Servislerin ApplicationEntity Sınıfına Kaydedilmesi

Servislerin kaydedilmesi görevini yapan ServiceController sınıfının registerServices() metodunun kod içeriği aşağıda görülebilir.

```
private void registerServices(ApplicationEntity ae, Service[]
services){
      Vector<TransferCapability> tcList = new
Vector<TransferCapability>();

      for (int i = 0; i < services.length; i++) {
            Service scp = services[i];
            ae.register((DicomService) scp);
```

```
            tcList.addAll(scp.getTransferCapability());
      }

      TransferCapability[] tc = new
TransferCapability[tcList.size()];
      for (int i = 0; i < tcList.size(); i++) {
            tc[i] = tcList.get(i);
      }

      ae.setTransferCapability(tc);
}
```

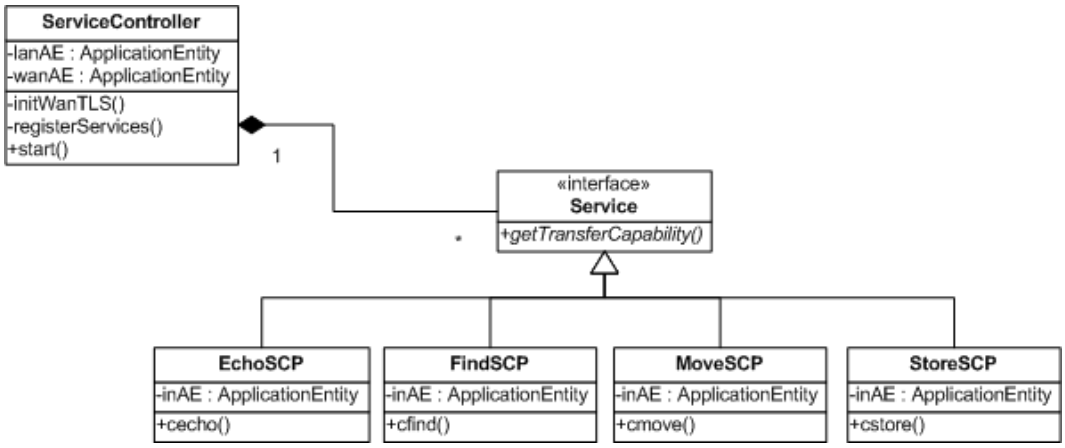
registerServices() metoduna gelen services dizisinin içerisindeki servisler ApplicationEntity sınıfına kaydettirilir ve her bir ApplicationEntity sınıfının sunduğu transfer yetenekleri (TransferCapability) bir vektöre atılır. Daha sonra bu vektör bir dizi aracılığı ile metoda gelen ApplicationEntity sınıfının setTransferCapability() metodu ile transfer yeteneklerini belirler.

Transfer yeteneği bir NetworkApplicationEntity'nin sunduğu ve kullanacağı DICOM servislerini tanımlar. Örneğin PatientRootQueryRetrieveInformationModelFIND hasta seviyesinde yapılacak olan sorgulama işlemini tanımlar. Her servisin (FindSCP, MoveSCP gibi) sunduğu, getTransferCapability() metodu içerisinde tanımlı tüm işlemleri, SCP ve SCU transfer yeteneği olarak bir vektöre ekleyerek geri döndürür. FindSCP sınıfının getTransferCapability() metodu aşağıda verilmiştir.

```
private static final String[] STUDY_LEVEL_FIND_CUID = {
        UID.StudyRootQueryRetrieveInformationModelFIND,
        UID.PatientRootQueryRetrieveInformationModelFIND,

UID.PatientStudyOnlyQueryRetrieveInformationModelFINDRetired };
...
public Vector<TransferCapability> getTransferCapability() {
      Vector<TransferCapability> tcList = new
Vector<TransferCapability>();

      for (int i = 0; i < STUDY_LEVEL_FIND_CUID.length; i++) {
            tcList.add(new
TransferCapability(STUDY_LEVEL_FIND_CUID[i], DEF_TS,
TransferCapability.SCP));
            tcList.add(new
TransferCapability(STUDY_LEVEL_FIND_CUID[i], DEF_TS,
TransferCapability.SCU));
      }
```

```
        return tcList;
}
```

Yukarıda anlatılan transfer yeteneklerinin servis sınıflarının getTransferCapability() metodu üzerinden alınıp kaydedilmesi işlemi *Strategy* tasarım deseni kullanılarak gerçekleştirilmiştir. Şekil 5.6'da verilen UML sınıf diyagramında *Strategy* deseninin kullanımı ve sınıflar arası etkileşim gösterilmiştir.



Şekil 5.6 Transfer Yeteneklerinin Kaydedilmesi İşleminde Strategy Deseninin Kullanımı

Servislerin tanımlama ve kayıt aşamasından sonra ServiceController sınıfının start() metodu ile servisler bağlantıları dinlemeye başlar.

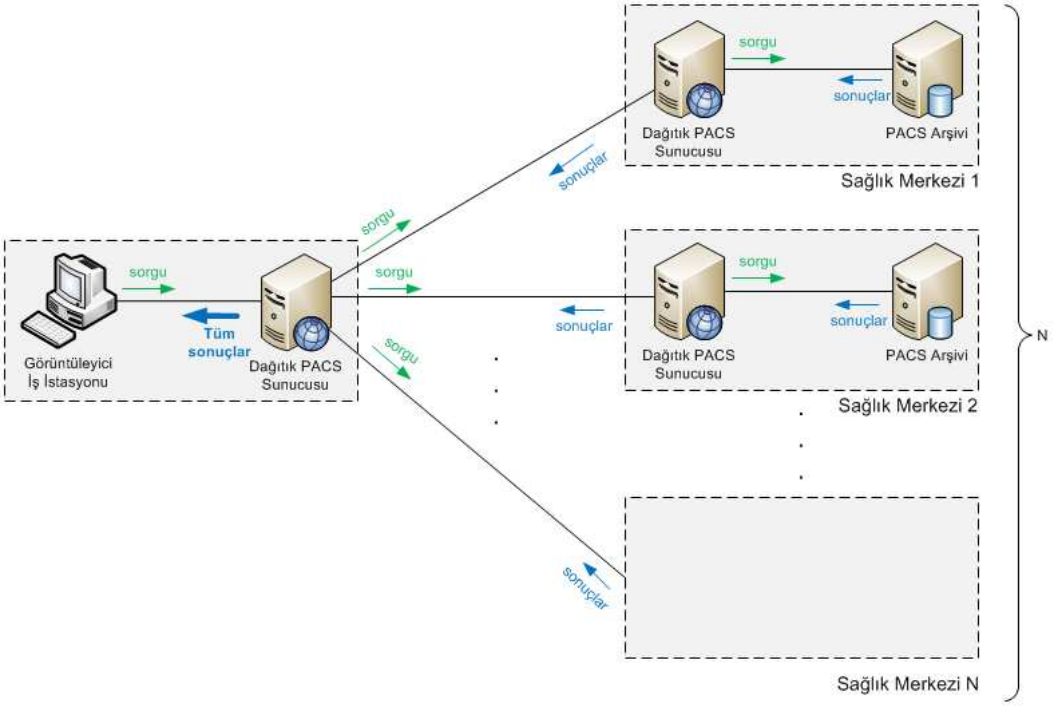
Sunulan servislerin ve transfer yeteneklerinin ApplicationEntity sınıfına nasıl tanımlanacağı belirtildikten sonra gelen isteklerin yorumlanması aşamasına geçilebilir.

Dağıtık PACS sunucusuna gelen istekler sorgulama isteği (C-Find) ve görüntü çağırma isteği (C-Move) olarak ikiye ayrılır.

5.2.1.3. Sorgulama

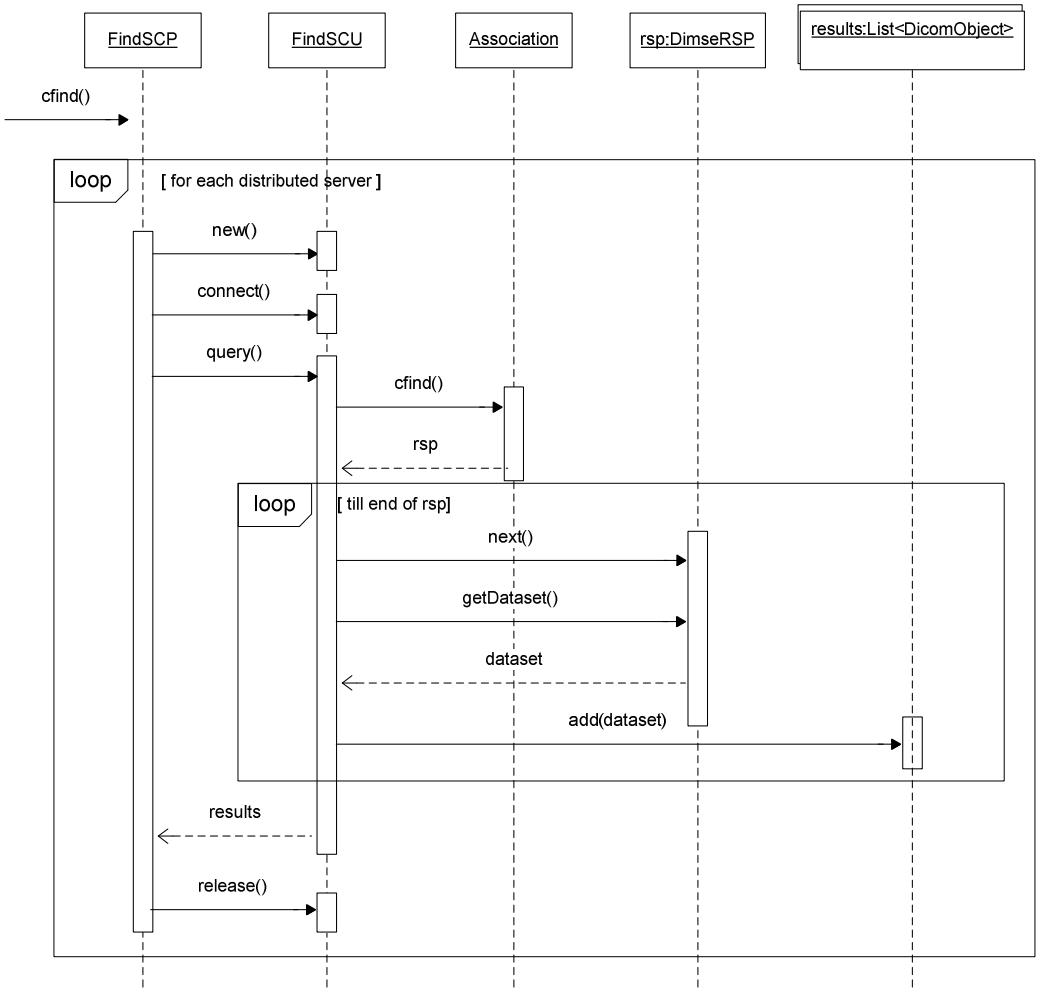
Doktor sağlık merkezi içerisinde bulunan bir görüntüleyici iş istasyonu üzerinden arama kriterlerini girerek sorgulama işlemini başlatır. Sorgulama isteği dağıtık PACS sunucusuna iletilir. Dağıtık PACS sunucusu, gelen sorguyu işleyerek tanımlı olan diğer dağıtık PACS sunucularına yönlendirir. Diğer dağıtık PACS sunucusu, gelen sorguyu içerisinde bulunduğu sağlık merkezindeki PACS sunucularına iletir. Sağlık merkezinde bulunan PACS sunucusu sorgu sonucunu

bağlı olduğu ağdaki dağıtık PACS sunucusuna gönderir. Bu sunucu ise isteği yapan ilk dağıtık PACS sunucusuna cevabı iletir. İsteği yapan dağıtık PACS sunucusu, birçok dağıtık PACS sunucusundan gelen sorgu sonuçlarını toplar ve iş istasyonuna sonuçları gönderir. Şekil 5.7 sistemler arasında yapılan bu mesaj alışverişini basit bir şekilde göstermektedir.

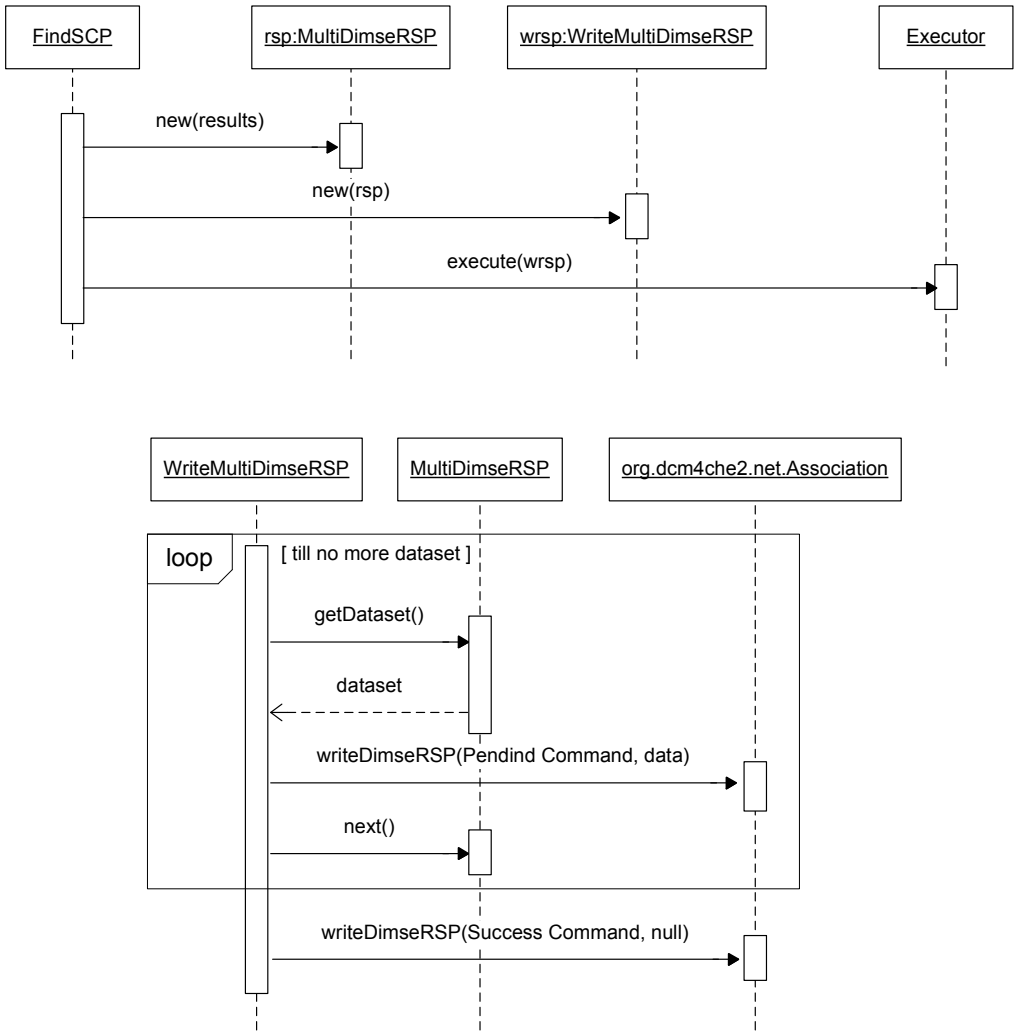


Şekil 5.7 Sorgulama İşleminde Bileşenler Arası Mesaj Alışverişi

Dağıtık PACS sunucusunun kendisine gelen sorgu isteğini hangi işlemlerden geçireceği Şekil 5.8'de sonuçları nasıl döndüreceği ise Şekil 5.9'de verilen ardıl etkileşim diyagramlarında gösterilmiştir.



Şekil 5.8 Sorgu İşlemi için Ardıl Etkileşim Diyagramı



Şekil 5.9 İşlemi için Ardıl Etkileşim Diyagramı (devamı)

Dağıtık sunucuya gelen C-Find-Rq mesajı daha önce NetworkApplicationEntity sınıfına kaydedilen FindSCP sınıfının cfind() metoduna yönlendirilir. cfind() metodu gelen sorguyu ve sorguyu gönderen sistemin AE ismini query() metoduna parametre olarak gönderir.

```
List<DicomObject> list = query(data, as.getCallingAET());
```

query() metodu içerisinde gelen sorgunun hangi sisteme ait olduğu *Singleton* tasarım deseni ile geliştirilen ImageServerHolder sınıfının addWorkstation() metodu ile kaydedilir. Bu işlemin amacı, dönen sorgu sonuçlarının hangi sistem tarafından sorgulandığını, görüntüye ait çalışmanın tekil numarasını (Study Instance UID) ve hangi sistemden döndüğünü tutarak, görüntü

çağrıldığında sadece görüntünün bulunduğu sunucuya isteği göndermektir. ImageServerHolder sınıfının tamamı Ek 3'ten incelenebilir.

query() metodu NetworkApplicationEntity sınıfının getServers() metodundan aldığı tüm sunucular için bir FindSCU sınıfını yaratır ve her birinin query() metoduna sorguyu ve isteğin yapıldığı sistemin AE ismini gönderir. Sonuçların tamamını topladıktan sonra metodun çağrıldığı yer olan cfind() metoduna geri döndürür. query() metodunun içeriği aşağıdaki gibidir.

```
private List<DicomObject> query(DicomObject data, String
callingAETitle){
	List<DicomObject> found = new ArrayList<DicomObject>();

	ImageServerHolder.getInstance().addWorkstation(callingAETitl
e);

	for (HostInformation serverHost : outAE.getServers()) {
		try {
			FindSCU findSCU = new FindSCU(outAE,
serverHost);

			findSCU.connect();
			found.addAll(findSCU.query(data,
callingAETitle));
			findSCU.release();
		} catch (InterruptedException e) {
			e.printStackTrace();
		} catch (IOException e) {
			e.printStackTrace();
		} catch (ConfigurationException e) {
			e.printStackTrace();
		}
	}

	return found;
}
```

FindSCU gelen isteği bağlantı kurulan sisteme (uygulama birimine), Association sınıfının cfind() metodu ile iletir. cfind() metodundan geri dönen DimseRSP nesnesinin içeriğini sırayla dolaşır ve getDataset() metodu ile sonuçları tutan DICOM nesnelerini (DicomObject) bir DicomObject listesine atar. Ayrıca her sorgu sonucu daha önce belirtildiği gibi *Singleton* tasarım deseni kullanılarak geliştirilen ImageServerHolder sınıfının addImageToServer() metodu

ile sorguyu yapan sistemin listesine eklenir. Son olarak sorgu sonucu DicomObject listesi halinde geri döndürülür. FindSCU sınıfının içerisindeki query() metodu detaylı incelenebilmesi için aşağıda verilmiştir.

```
public List<DicomObject> query(DicomObject data, String
callingAETitle) throws IOException, InterruptedException{
	List<DicomObject> results = new ArrayList<DicomObject>();

	...

	DimseRSP rsp = assoc.cfind(cuid, priority, data, tsuid,
cancelAfter);

	while(rsp.next()){
		DicomObject cmd = rsp.getCommand();
		if(CommandUtils.isPending(cmd)){
			DicomObject dataset = rsp.getDataset();

			results.add(dataset);

			try {

	ImageServerHolder.getInstance().addImageToServer(callingAETi
tle, dataset.getString(Tag.StudyInstanceUID),
assoc.getCalledAET());
			} catch (ObjectNotFoundException e) {
				e.printStackTrace();
			}
		}
	}

	return results;
}
```

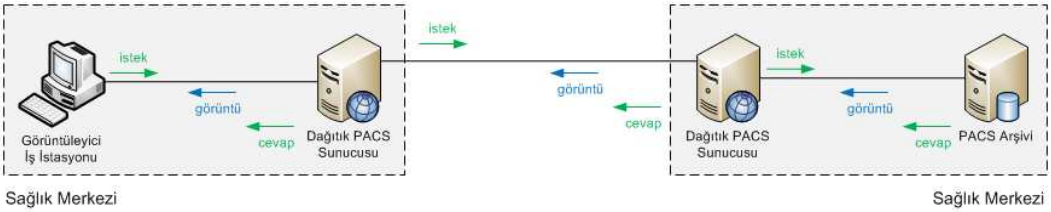
Sunuculara gönderilen FindSCU isteklerinin sonuçları, ilk isteğin geldiği yer olan FindSCP sınıfının cfind() metodundaki results listesinde toplanır. Birden fazla sonucu, sorguyu gönderen sisteme döndürmek için DimseRSP sınıfından türetilen MultiDimseRSP kullanılır. DimseRSP sınıfı ile Association sınıfı üzerinden geriye tek bir sonuç döndürebilirken, MultiDimseRSP ile birden fazla sonuç döndürülebilmesi sağlanmıştır. DimseRSP sınıfı *Iterator* tasarım deseni kullanılarak geliştirildiği için bu sınıfı genişleten MultiDimseRSP sınıfı *Iterator* tasarım deseninin özelliklerini barındırır. MultiDimseRSP sınıfından yaratılan rsp nesnesi WriteMultiDimseRSP sınıfına gönderilir ve executor nesnesi ile farklı bir iş parçacığında çalıştırılır.

```
private void cfind(Association as, int pcid, DicomObject rq,
DicomObject data) throws IOException {
      ...
      DimseRSP rsp = new MultiDimseRSP(cmdrsp, results);
      ...
      cmdrsp = rsp.getCommand();
      if (CommandUtils.isPending(cmdrsp))
      {
            executor.execute(new WriteMultiDimseRsp(as, pcid,
rsp));
      }
}
```

WriteMultiDimseRSP sınıfı içerisinde sonuçlar Association sınıfının writeDimseRSP() metodu ile tek tek sorgu sahibine döndürülür. WriteMultiDimseRSP ve MultiDimseRSP sınıfı Ek 3'te görülebilir.

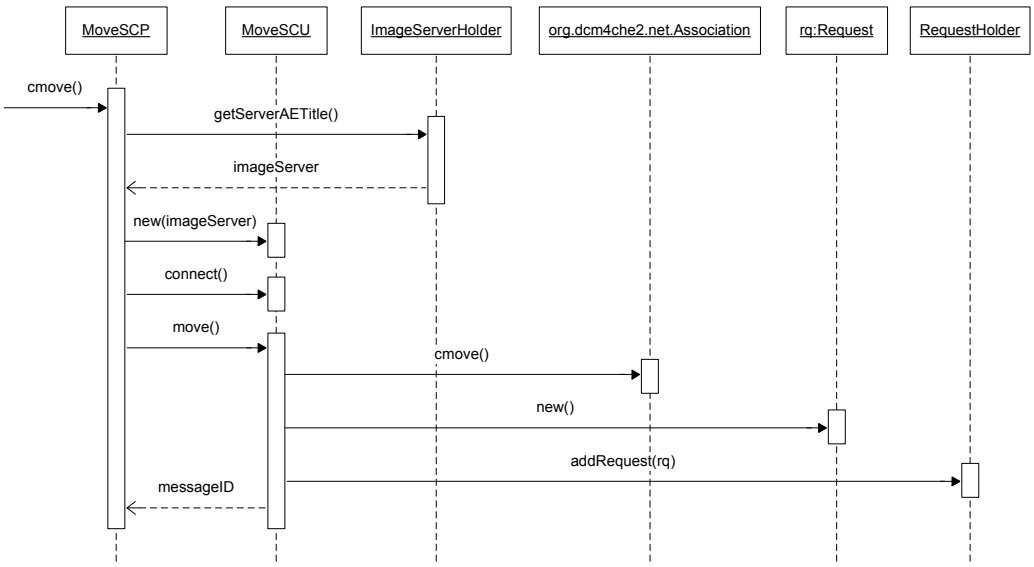
5.2.1.4. Görüntü çağırma

Doktor bir önceki adımda görüntüleyiciye dönen sorgu sonuçlarından görmek istediğini seçerek görüntüyü çağırma işlemini başlatır. Görüntü çağırma isteği dağıtık PACS sunucusuna iletilir. İsteği alan dağıtık PACS sunucusu, sorgulama işleminin sonucunda dönen görüntü bilgilerinin hangi dağıtık PACS sunucudan geldiğini bildiği için isteği, doğrudan ilgili sunucuya yönlendirir. Görüntü çağırma isteğini alan diğer dağıtık PACS sunucusu, isteği sağlık merkezindeki PACS sunucusuna gönderir. PACS sunucusu görüntüyü farklı bir bağlantı açarak dağıtık PACS sunucusuna gönderir. Ayrıca görüntüyü gönderdikten sonra görüntü çağırma isteğinin cevabı olarak kaç görüntü gönderdiği, kaç görüntü daha göndereceği gibi bilgileri de sunucuya iletir. Aynı şekilde; dağıtık PACS sunucusu, ilk isteği yapan dağıtık PACS sunucusuna cevabı ve farklı bir bağlantı ile görüntüyü gönderir. Son olarak dağıtık PACS sunucusu kaç görüntü gönderdiği, kaç görüntü daha göndereceği gibi bilgileri cevap olarak ve ayrı bir bağlantı ile görüntüyü iş istasyonuna gönderir. Sistemler arasında yapılan bu mesaj ve görüntü alışverişini Şekil 5.10 basit bir şekilde göstermektedir.



Şekil 5.10 Görüntü Çağırma İşleminde Bileşenler Arası Mesaj Alışverişi

Dağıtık PACS sistemine gelen görüntü çağırma isteğini nasıl işlediği Şekil 5.11'daki ardıl etkileşim diyagramında verilmiştir.



Şekil 5.11 Görüntü Çağırma İşlemi için Ardıl Etkileşim Diyagramı

Görüntü çağırma isteği dağıtık PACS sunucusuna geldikten sonra, MoveSCP sınıfının cmove() metodu otomatik olarak çalıştırılır. Gelen istek doğrultusunda, görüntünün hangi sunucuda olduğu bilgisi *Singleton* tasarım deseni kullanılarak geliştirilen ImageServerHolder sınıfının getServerAETitle() metodu ile öğrenilir. İlgili sunucuya bağlantı bir MoveSCU yaratılarak sağlanır ve istek MoveSCU sınıfının move() metoduna gönderilir. move() metodu ilgili sunucuya gönderilen mesajın tekil numarasını döndürür. Bu mesaj numarası daha sonra, gelen görüntünün hangi move mesajına istinaden geldiği bilgisi için kullanılacaktır. Bu işlemleri yürüten cmove() metodunun içeriği aşağıdaki gibidir.

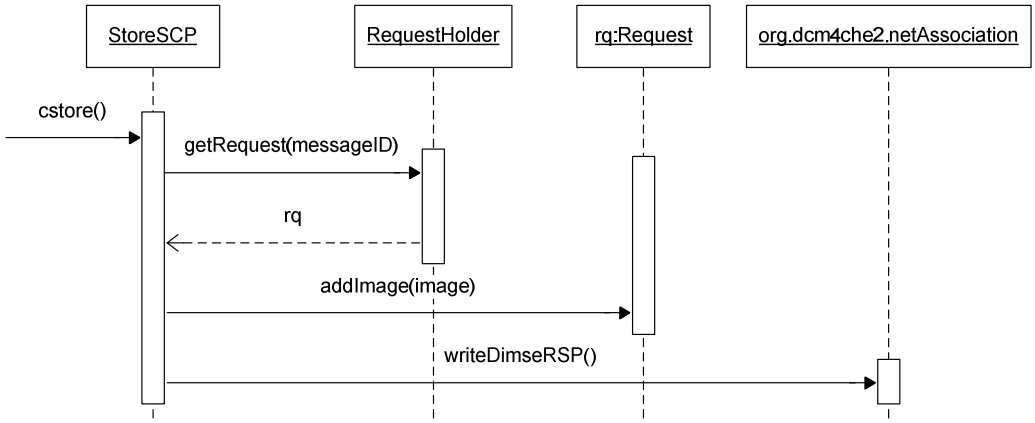
```
private void cmove(Association as, int pcid, DicomObject cmd,
DicomObject data){
      ...
      try {
            String imageServer =
ImageServerHolder.getInstance().getServerAETitle(as.getCallingAET(
), data.getString(Tag.StudyInstanceUID));

            MoveSCU moveSCU = new MoveSCU(outAE,
outAE.getHostByAETitle(imageServer), as.getLocalAET());

            moveSCU.connect();
            messageID = moveSCU.move(data);
            moveSCU.release();
      } catch (IOException e) {
            e.printStackTrace();
      } catch (InterruptedException e) {
            e.printStackTrace();
      } catch (ObjectNotFoundException e) {
            e.printStackTrace();
      } catch (ConfigurationException e) {
            e.printStackTrace();
      }
      ...
}
```

MoveSCU sınıfının move() metodu, gelen isteği ilgili sunucuya iletmek görevini üstlenmiştir. Bu metod, isteği Association sınıfının cmove() metodu ile karşı tarafa iletir. Sonrasında geri gelecek resimlerin ilgili mesajlar ile eşleştirilebilmesi için üretilen mesaj numarası ile bir Request nesnesi üretilir ve bu nesne *Singleton* tasarım deseni ile geliştirilen RequestHolder isimli sınıfa addRequest() metodu ile eklenir. MoveSCU sınıfında bulunan move() metodunun kod içeriği aşağıda verilmiştir.

```
public int move(DicomObject data) throws IOException,
InterruptedException{
      ...

      assoc.cmove(cuid, priority, data, tsuid, moveDestinationAE,
rspHandler);

      ...

      RequestHolder.getInstance().addRequest(new
Request(rspHandler.getMessageID()));
```

```
        assoc.waitForDimseRSP();

        return rspHandler.getMessageID();
}
```

Yukarıda anlatılan işlemler neticesinde, isteğin ilgili sunucuya iletilmesi işlemi gerçekleştirilmiş olur. İsteği alan sunucu istenilen görüntüyü C-Store işlemi ile isteği yapan sunucuya gönderir. Dağıtık PACS sunucusunun gelen görüntüleri nasıl aldığı Şekil 5.12'deki ardıl etkileşim diyagramında görülebilir.



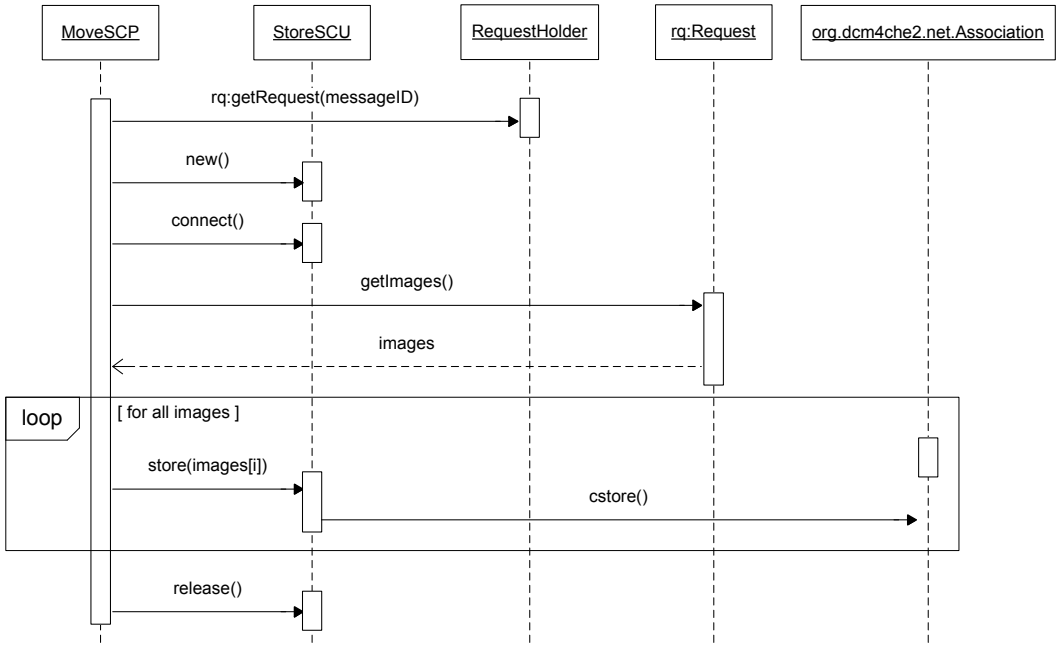
Şekil 5.12 Görüntü Çağırma İşlemi için Ardıl Etkileşim Diyagramı (devamı)

Görüntü çağırma isteği neticesinde dağıtık PACS sunucusuna gelen görüntüler, StoreSCP sınıfının cstore() metoduna yönlendirilirler. Gelen görüntü transfer sözdizimi tekil numarası (transfer syntax uid) ile birlikte bir DicomImage nesnesinin içerisinde tutulur. Bu nesne, gelen görüntünün mesaj numarasına göre (MoveOriginatorMessageID) *Singleton* tasarım deseni kullanılarak geliştirilen RequestHolder sınıfının içerisinde bulunan ilgili Request nesnesinin addImage() metodu ile eklenir. Son olarak, görüntüyü alan dağıtık PACS sunucusu gönderen tarafa alındı mesajını iletir. StoreSCP sınıfının cstore() metodundan bir kesit aşağıda sunulmuştur.

```
public void cstore(Association as, int pcid, DicomObject rq,
PDVInputStream dataStream, String tsuid) throws
DicomServiceException, IOException {
        ...

        DicomObject image = dataStream.readDataset();
```

```
        RequestHolder.getInstance().getRequest
(rq.getInt(Tag.MoveOriginatorMessageID)).addImage(new
DicomImage(tsuid, image));

        as.writeDimseRSP(pcid, rsp);

        ...
}
```

Dağıtık PACS sunucusu, son olarak, gelen tüm görüntüleri isteği yapan sisteme gönderir. Şekil 5.13'te görüntülerin isteği yapan sisteme, dağıtık PACS sunucusu tarafından nasıl gönderileceği ardıl etkileşim diyagramı ile gösterilmiştir.



Şekil 5.13 Görüntü Çağırma İşlemi için Ardıl Etkileşim Diyagramı (devamı)

cmove() metodunun devamında, gelen görüntüleri isteği yapan sisteme göndermek için gerekli olan işlemler yapılır. İlk olarak RequestHolder sınıfından gönderilen mesaj numarası ile ilgili Request nesnesi çekilir. Daha sonra isteği yapan sisteme, kaç görüntünün gönderileceği bilgisi sendMoveRSP() metodu ile move isteğinin cevabı olarak iletilmelidir. Ardından alıcı tarafa StoreSCU sınıfı ile farklı bir bağlantı açılır. Alınan tüm görüntüler Request nesnesinden getImages() metodu ile çekilerek, açılan bağlantı üzerinden tek tek StoreSCU sınıfının store() metodu ile isteği yapan tarafa gönderilir. Her görüntü gönderimi

sonrasında gönderilen görüntü sayısı, kalan görüntü sayısı gibi bilgiler sendMoveRSP() metodu ile C-Move-Rq mesajına cevap olarak gönderilir. Görüntü gönderimi bittikten sonra RequestHolder sınıfı içerisinden ilgili Request silinerek işlem tamamlanır. MoveSCP sınıfındaki cmove() metodunun yukarıdaki işlemleri yapan kod parçacığı aşağıda verilmiştir.

```
private void cmove(Association as, int pcid, DicomObject cmd,
DicomObject data){
	...

	Request rq =
RequestHolder.getInstance().getRequest(messageID);

	rq.setStudyInstanceUID(cmd.getString(Tag.StudyInstanceUID));

	int i = 0;
	int total = rq.getImages().size();

	sendMoveRSP(i, total, as, pcid, rsp);

	try {
		StoreSCU scu = new StoreSCU(inAE,
inAE.getHostByAETitle(cmd.getString(Tag.MoveDestination)));
		scu.connect();

		for (DicomImage image : rq.getImages()) {
			scu.store(image,
cmd.getString(Tag.MoveDestination), cmd.getInt(Tag.MessageID));

			i++;

			sendMoveRSP(i, total, as, pcid, rsp);
		}

		scu.release();
	} catch (ObjectNotFoundException e) {
		e.printStackTrace();
	} catch (ConfigurationException e) {
		e.printStackTrace();
	} catch (IOException e) {
		e.printStackTrace();
	} catch (InterruptedException e) {
		e.printStackTrace();
	}
```

```
        RequestHolder.getInstance().removeRequest(messageID);
}
```

5.2.1.5. Dağıtık PACS sunucuları arası güvenlik

Dağıtık PACS sunucuları arasında güvenliği sağlayabilmek için daha önce bahsedilen TLS/SSL kullanılmıştır. TLS iletişiminde kullanılacak olan güvenlik ile ilgili dosyalar (node.cert, node.jks, trust.jks), projenin tls klasörü içerisinde yer almaktadır. Bu güvenlik dosyaları JRE içerisinde bulunan keytool yazılımı yardımı ile oluşturulmuştur. Komut satırı üzerinde bu dosyaları oluşturan komutlar aşağıda verilmiştir.

```
keytool -genkey -keyalg RSA --keystore node.jks -alias node
keytool -export -file node.cert -keystore node.jks -alias node
keytool -import -file node.cert -keystore trust.jks -alias node
```

ServiceController sınıfı içerisinde diğer dağıtık sunucular ile iletişimde kullanılan bağlantının güvenliğini etkinleştiren initWanTLS() metodu ve sertifikalara ait şifreler aşağıda verilmiştir.

```
private static final String KEYSTORE_PASSWORD =
"j?pZO|p$f&Sp+3$A";
private static final String TRUSTSTORE_PASSWORD =
"m8+/rc#|!bMn@mZ*";
private static final String KEY_PASSWORD = ",`EF.e4!L|4F'5gx";
...
private void initWanTLS(){
        wanAE.setAsWanInterface(true);

        wanConnection.setTls3DES_EDE_CBC();
        wanConnection.setTlsNeedClientAuth(true);

        try{
                KeyStore keyStore = loadKeyStore("../tls/node.jks",
KEYSTORE_PASSWORD.toCharArray());
                KeyStore trustStore = loadKeyStore("../tls/trust.jks",
TRUSTSTORE_PASSWORD.toCharArray());
                wanDevice.initTLS(keyStore,
KEY_PASSWORD.toCharArray(), trustStore);
        }catch(GeneralSecurityException e){
                e.printStackTrace();
        } catch (IOException e) {
                e.printStackTrace();
        }
}
```

Dağıtık PACS sistemleri arasında TLS bağlantısını etkinleştirebilmek için ilk olarak wanAE nesnesinin setAsWanInterface () metodu çalıştırılır. Böylece ApplicationEntity sınıfından üretilen wanAE nesnesi ile güvenli bağlantılar kurabilmesi sağlanır. Ardından wanAE nesnesine ait NetworkConnection sınıfının setTls3DES_EDE_CBC() metodu ile 3DES (Triple Data Encryption Standard) şifrelemesi yapılacağı setTlsNeedClientAuth() metodu ile de istemcinin yetkilendirilmesi gerekliliği belirtilir. Son olarak şifrelemede kullanılacak dosyalar yüklenir ve wanDevice nesnesinin initTLS() metodu çalıştırılarak dağıtık PACS sistemleri ile kurulacak bağlantıların güvenli olması sağlanır.

Diğer dağıtık PACS sunucularına bağlantı sağlanırken açılan bağlantının da (NetworkConnection) güvenliği etkinleştirilmelidir. Fakat AbstractSCU sınıfını genişleten sınıflar, hem dağıtık PACS sunucularına hem de sağlık merkezi içerisindeki sistemlere bağlantı kurabilmek için tasarlanmıştır. Diğer dağıtık PACS sunucularına bağlanırken daha önce bahsedilen setAsWanInterface() metodunun ApplicationEntity nesnesinde yarattığı değişiklikten yararlanılır. Kısaca kullanılan ApplicationEntity nesnesinin güvenliği etkinleştirilmiş ise o ApplicationEntity üzerinden kurulacak uzak bağlantılar TLS ile şifrelenmelidir. Aşağıda bu işlemin nasıl yapıldığı kısa bir kod parçacığı ile anlatılmaktadır.

```
if (localAE.isWanInterface()){
      remoteConn.setTls3DES_EDE_CBC();
}
```

Ayrıca servis sunan her SCP içerisinde, o SCP'ye bağlantı sağlayan sistemlerin doğrulaması yapılmaktadır. Örneğin MoveSCP sınıfının cmove() metodunda bu işlemin nasıl yapıldığı aşağıdaki kod parçacığı ile gösterilmiştir.

```
public void cmove(Association as, int pcid, DicomObject cmd,
DicomObject data) throws DicomServiceException, IOException {
      if (inAE.authenticate(as.getCallingAET())) {
            ...
      } else {
            as.abort();
      }
}
```

İsteği gönderen sistemin AE ismi, ilgili ApplicationEntity nesnesinin authenticate() metodu ile doğrulanır. Eğer sistem doğrulanmaz ise bağlantı koparılır. ApplicationEntity sınıfında doğrulama işlemini gerçekleştiren authenticate() metodu aşağıda incelenebilir. Bu metod içerisinde bağlantı kurmaya

çalışan sistemin konfigürasyon dosyası veya isim sunucusu ile sisteme tanıtılmış olması beklenir.

```
public boolean authenticate(String aeTitle){
	try {
		HostInformation host = getHostByAETitle(aeTitle);

		return host != null;
	} catch (ObjectNotFoundException e) {
		return false;
	}
}
```

5.2.2. İsim sunucusu

İsim sunucusu, dağıtık PACS sunucularının erişim bilgilerini tutar ve bu bilgileri tüm dağıtık PACS sunucuları ile paylaşır. Bu yüzden isim sunucusu tüm dağıtık PACS sunucularının erişebildiği farklı bir sunucu üzerinde çalışmalıdır. İsim sunucusu aşağıdaki kod parçacığı ile başlatılır.

```
NameServer obj = new NameServerImpl();
NameServer stub = (NameServer)
UnicastRemoteObject.exportObject(obj, 0);

Registry registry = LocateRegistry.createRegistry(1099);
registry.bind("PacsNameServer", stub);
```

NameServer arayüzünü gerçekleştiren NameServerImpl sınıfından nesne yaratılarak bir kütük (stub) oluşturulur. Daha sonra 1099 portunu dinleyen yeni bir Registry nesnesi yaratılır ve daha önce oluşturulan kütük bu nesneye PacsNameServer ismiyle bağlanır. Yani isim sunucusu, dağıtık PACS sistemlerinden gelen istekleri 1099 portundan dinlemeye başlar.

İsim sunucusunun içerisinde bulunan NameServerImpl sınıfı NameServer arayüzünü gerçekleştirir. Dağıtık PACS sunucularına diğer sunucuların erişim bilgilerini sunan getRemoteServers() metodunun gerçekleştirimi aşağıdaki gibidir.

```
public Vector<HostInformation> getRemoteServers() throws
RemoteException {
	ConfigurationParser parser = new
ConfigurationParser("servers.xml");
	...
	return parser.getApplicationEntities(AEType.SERVER);
```

```
}
```

getRemoteServers() metodu servers.xml isimli bir yapılandırma dosyasından tüm dağıtık PACS sunucularının erişim bilgilerini okur ve bu bilgileri uzak metodu çağıran istemciye geri döndürür. servers.xml yapılandırma dosyası temel olarak daha önce bahsedilen configuration.xml dosyası ile benzer özellikleri taşımaktadır. Örnek bir servers.xml dosyasının içeriği aşağıda verilmiştir.

```
<Configuration>
      <Servers>
            <Server>
                  <AETitle>DIPACS1</AETitle>
                  <Hostname>node1.dipacs.org</Hostname>
                  <Port>104</Port>
            </Server>
            <Server>
                  <AETitle>DIPACS2</AETitle>
                  <Hostname>node2.dipacs.org</Hostname>
                  <Port>104</Port>
            </Server>
            <Server>
                  <AETitle>DIPACS3</AETitle>
                  <Hostname>node3.dipacs.org</Hostname>
                  <Port>104</Port>
            </Server>
      </Servers>
</Configuration>
```

Yukarıdaki örnekte node1.dipacs.org, node2.dipacs.org ve node3.dipacs.org alan adları üzerinden erişilebilecek üç dağıtık PACS sunucusunun erişim bilgileri bulunmaktadır.

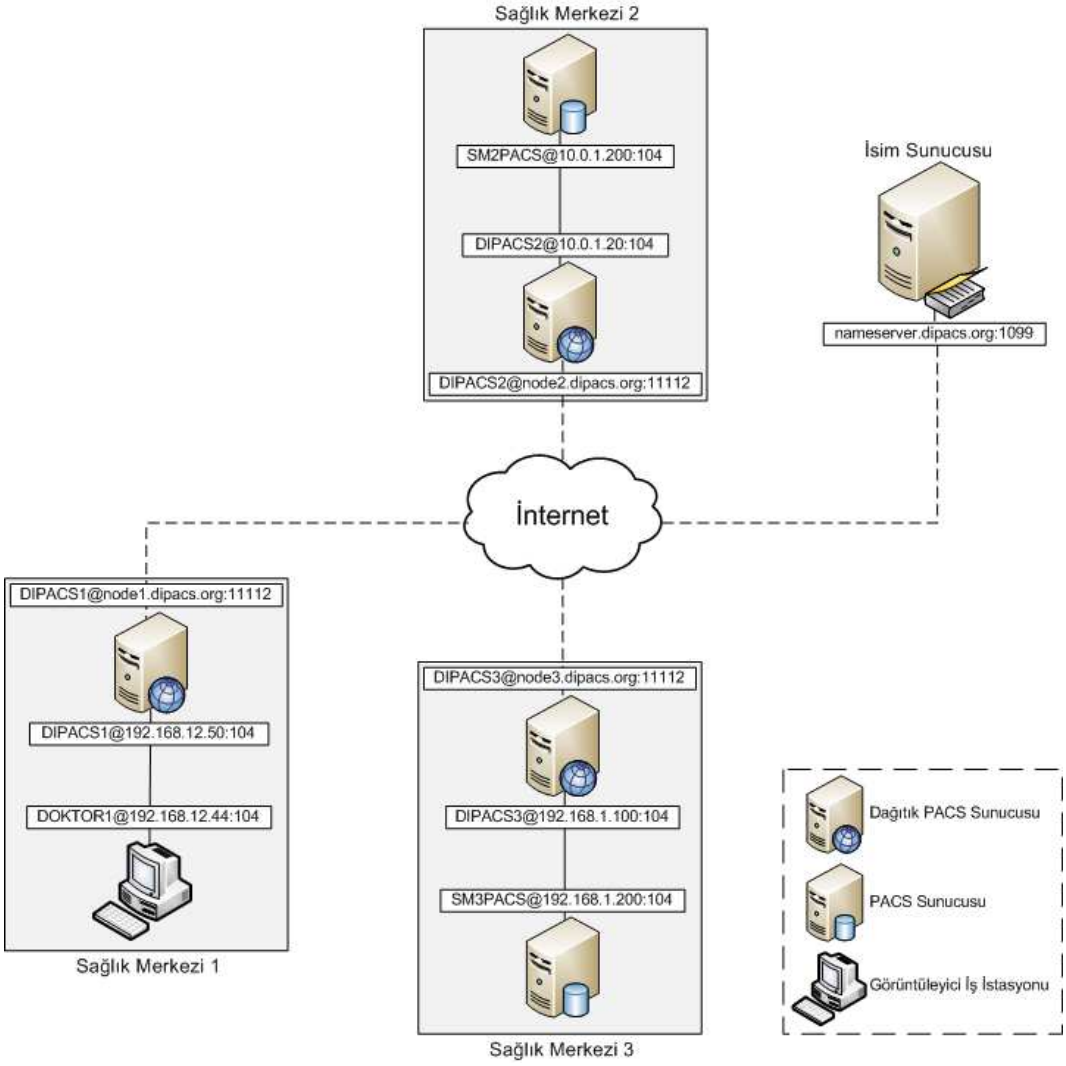
5.3. Örnek Bir Senaryo

Dağıtık PACS sisteminin çalışabilmesi için bazı dosya ve klasörler gerekmektedir. Gerekli olan dosyalar ve açıklamaları:

- *bin*: Uygulamaların çalıştırılmasında kullanılan komutlar ve yapılandırma dosyaları
    - *dipacs.bat*: Dağıtık PACS sunucusunu çalıştırmak için kullanılan Windows yığın (batch) dosyası

  - *nameserver.bat*: İsim sunucusunu çalıştırmak için kullanılan Windows yığın dosyası
  - *configuration.xml*: Dağıtık PACS sunucusununun XML (Extensible Markup Language) yapılandırma dosyası
  - *servers.xml*: İsim sunucusununun XML yapılandırma dosyası
- *lib*: Kullanılan kütüphaneler
  - *dipacs.jar*: Dağıtık PACS sunucusunun gerçekleştirimini içeren Java arşivi
  - *dipacs-nameserver*.jar: İsim sunucusunun gerçekleştirimini içeren Java arşivi
  - *dipacs-common.jar*: Dağıtık PACS sunucusu ve isim sunucusunun ortak kullandıkları bileşenleri içeren Java arşivi
  - *dcm4che-core-2.0.21.jar*: dcm4che2 kütüphanesinin çekirdeğini içeren Java arşivi
  - *dcm4che-net-2.0.21.jar*: dcm4che2 kütüphanesinin DICOM iletişimini gerçekleştiren Java arşivi
  - *log4j-1.2.13.jar*: log4j, kayıt tutma kütüphanesi
  - *slf4j-api-1.5.0.jar*: Kayıt tutma arayüzü
  - *slf4j-log4j12-1.5.0.jar*: Kayıt tutma arayüzünün log4j için gerekli olan gerçekleştirimi
- *tls*: Güvenli iletişimde kullanılacak sertifika ve dosyalar
  - *node.cert*
  - *node.jks*
  - *trust.jks*
- *logs*: Sunucu ve isim sunucusu tarafından tutulan kayıtları içeren dosyalar
  - *server.log*
  - *nameserver.log*

Bu bölümde, geliştirilen uygulamaların yukarıda anlatılan dosyalar ile nasıl çalıştırıldığı, nasıl yapılandırıldığı ve sistemin nasıl çalıştığı örnek bir senaryo üzerinden anlatılacaktır. Şekil 5.14’de örnek senaryo içerisindeki bileşenlerin yapılandırmaları ve birbirlerine nasıl bağlandıkları görülebilir.



Şekil 5.14 Örnek Bir Dağıtık PACS Yapılandırması

Şekildeki bileşenlerin erişim bilgileri kutular içerisinde verilmektedir. Örneğin DOKTOR1@192.168.12.44:104, DOKTOR1 AE ismine sahip 192.168.12.44 IP adresi ve 104 üzerinden bağlantı kurabilecek bir görüntüleyici iş istasyonunu belirtmektedir. Ayrıca şekile dikkatli bakıldığında dağıtık PACS sunucularının iki erişim bilgisi bulunduğu görülebilir. Bunun nedeni dağıtık PACS sunucusunun sağlık merkezi içerisinde bulunan PACS bileşenleri ile iletişimde kullanacağı bir erişim noktası ve diğer dağıtık PACS sunucuları ile iletişimde kullanacağı farklı bir erişim noktasına ihtiyacı olmasıdır. Örneğin sağlık merkezi 1'de bulunan dağıtık PACS sunucusu 104 portundan aldığı istekleri 11112 portu üzerinden diğer dağıtık PACS sunucularına iletmektedir.

Ayrıca sistemin işleyişini deneyebilmek için açık kaynak olarak geliştirilen ClearCanvas PACS görüntüleyici[1] ve dcm4chee[2] PACS sunucusu kullanılmıştır. Konunun ilerleyen bölümlerinde her iki yazılımdan da alınan ekran görüntüleri verilecektir.

5.3.1. Sistemin başlatılması

Sistem başlatılmadan ya da bir dağıtık sunucu çalıştırılmadan önce, isim sunucusunun yapılandırılması ve çalıştırılması gereklidir.

İsim sunucusunu yapılandırmak için servers.xml isimli dosya kullanılır. Daha önce anlatıldığı gibi, bu dosya içerisine dağıtık sistemdeki tüm dağıtık PACS sunucularının erişim bilgileri girilmelidir. Şekil 5.14'teki sistem için servers.xml dosyasının içeriği şu şekilde olmalıdır.

```
<Configuration>
      <Servers>
            <Server>
                  <Title>Sağlık Merkezi 1</Title>
                  <AETitle>DIPACS1</AETitle>
                  <Hostname>node1.dipacs.org</Hostname>
                  <Port>11112</Port>
            </Server>
            <Server>
                  <Title>Sağlık Merkezi 2</Title>
                  <AETitle>DIPACS2</AETitle>
                  <Hostname>node2.dipacs.org</Hostname>
                  <Port>11112</Port>
            </Server>
            <Server>
                  <Title>Sağlık Merkezi 3</Title>
                  <AETitle>DIPACS3</AETitle>
                  <Hostname>node3.dipacs.org</Hostname>
                  <Port>11112</Port>
            </Server>
      </Servers>
</Configuration>
```

[1] http://www.clearcanvas.ca Erişim tarihi: 01.09.2009

[2] http://www.dcm4che.org Erişim tarihi: 01.09.2009

servers.xml dosyası düzenlendikten sonra isim sunucusu çalışmaya hazır hale gelir. İsim sunucusu nameserver.bat dosyası çalıştırılarak başlatılır. Bu işlemin komut satırı çıktısı şu şekildedir:

```
C:\dipacs\bin>nameserver.bat
diPACS Name Server started...
```

İsim sunucusu hazır hale getirildikten sonra dağıtık PACS sunucuları başlatılabilir. İsim sunucusunda olduğu gibi Dağıtık PACS sunucuları da yapılandırılmalıdır. Bunun için configuration.xml isimli dosya kullanılır.

Sağlık Merkezi 1'deki dağıtık PACS sunucusu için kullanılan yapılandırma dosyası:

```
<Configuration>
      <Servers>
            ...
      </Servers>
      <Workstations>
            <Workstation>
                  <AETitle>DOKTOR1</AETitle>
                  <Hostname>192.168.12.50</Hostname>
                  <Port>104</Port>
            </Workstation>
      </Workstations>
</Configuration>
```

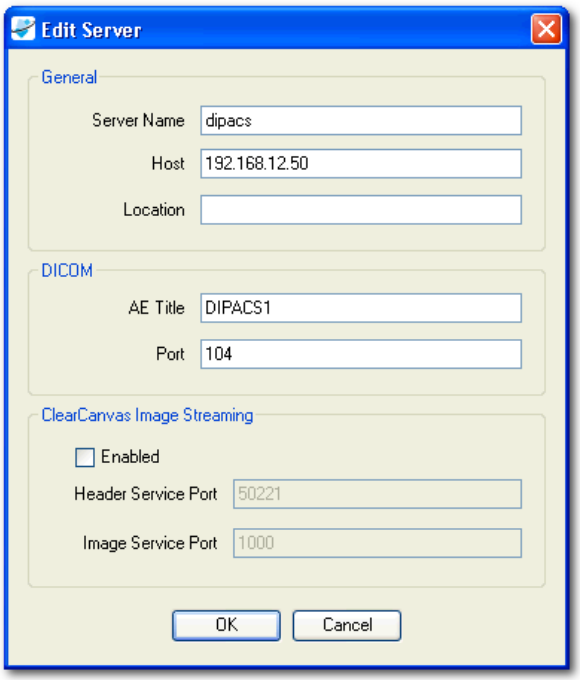
Sağlık Merkezi 2'deki dağıtık PACS sunucusu için kullanılan yapılandırma dosyası:

```
<Configuration>
      <Servers>
            <Server>
                  <AETitle>SM2PACS</AETitle>
                  <Hostname>10.0.1.200</Hostname>
                  <Port>104</Port>
            </Server>
      </Servers>
      <Workstations>
            ...
      </Workstations>
</Configuration>
```

Sağlık Merkezi 3 için de yapılandırma benzer şekilde olacaktır.

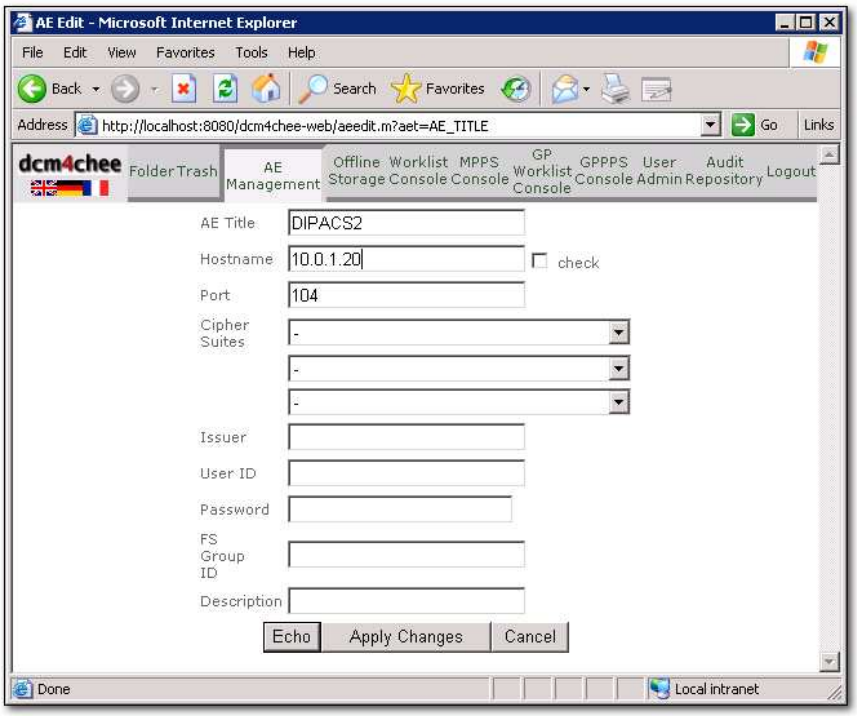
Yukarıdaki yapılandırma dosyalarının içeriklerinden de anlaşılacağı gibi, sağlık merkezi içerisindeki PACS sunucuları ve görüntüleyici iş istasyonları, yapılandırma dosyası ile dağıtık PACS sunucusuna tanıtılmalıdır. Ayrıca sağlık merkezi içerisindeki dağıtık PACS sunucusu da bu uygulamalara tanıtılmalıdır. Aksi halde dağıtık PACS sunucusu ile PACS bileşenleri arasında bağlantı sağlanamaz.

Sağlık Merkezi 1'de bulunan görüntüleyici iş istasyonuna dağıtık PACS sunucusu Şekil 5.15'de görüldüğü gibi tanıtılır.



Şekil 5.15 Dağıtık PACS Sunucusunun İş İstasyonuna Tanıtılması

Şekil 5.16'te de Sağlık Merkezi 2'de bulunan dağıtık PACS sunucusunun PACS arşivine tanıtıldığı ekran görülebilir.



Şekil 5.16 Dağıtık PACS Sunucusunun PACS Arşivine Tanıtılması

Dağıtık PACS sunucuları ile PACS bileşenlerinin birbirleri ile iletişime geçebilmeleri için yapılan yapılandırma işlemlerinden sonra dağıtık sunucular çalıştırılabilir.

Dağıtık sunucuları başlatabilmek için konunun başında da bahsedildiği gibi, dipacs.bat isimli yığın dosyası kullanılır. Fakat bu komut bazı parametreler almaktadır, isim sunucusu gibi tek komutla çalıştırılamaz. Dağıtık PACS sunucusunu çalıştırabilmek için şu komut ve parametreler kullanılır.

```
dipacs içAE içPort dışAE dışPort
```

Dağıtık sunucuların iç ve dış bacak erişim bilgileri komut satırı üzerinden parametreler ile atanmaktadır. Sağlık merkezi üzerinden erişilecek arayüz içAE ve içPort ile belirtilirken, dağıtık PACS sunucuları ile iletişim dışAE ve dışPort üzerinden sürdürülür.

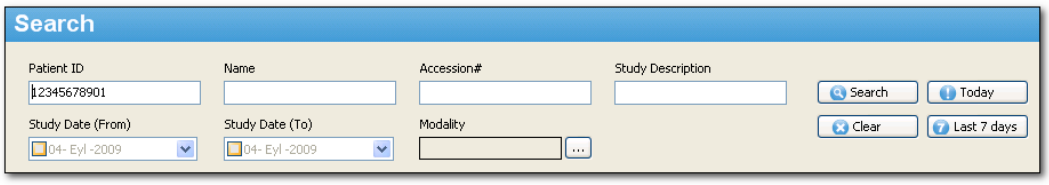
Şekil 5.14'de gösterilen dağıtık PACS sunucu uygulamaları çalışacağı sunucu üzerinde şu şekilde çalıştırılır:

```
dipacs DIPACS1 11112 DIPACS1 104
dipacs DIPACS2 11112 DIPACS2 104
dipacs DIPACS3 11112 DIPACS3 104
```

Dağıtık PACS sunucuları daha önce anlatıldığı gibi ilk olarak yapılandırma dosyası üzerinden, içerisinde bulunduğu ağdaki PACS bileşenlerini okur. Ardından isim sunucusu ile iletişime geçerek dağıtık PACS sunucularının bir listesini alır. Bu aşamadan sonra dağıtık sunucular ağın içinden ve dışından gelecek olan mesajları dinlemeye başlarlar.

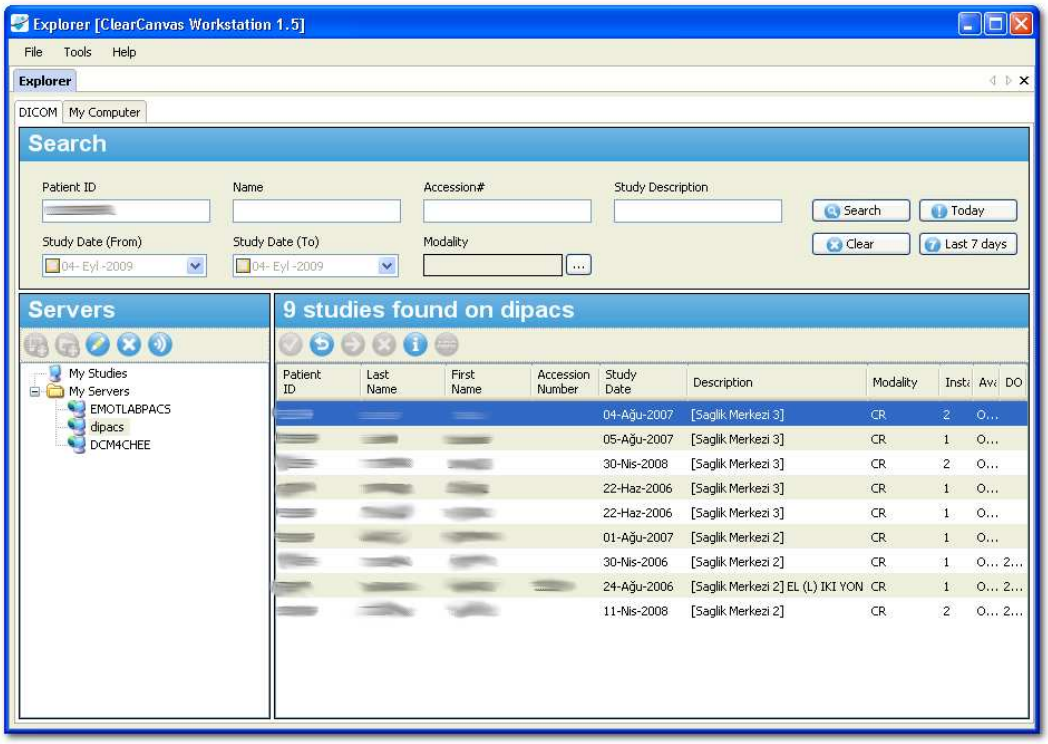
5.3.2. Sorgulama

Sağlık Merkezi 1'de bulunan DOKTOR1 AE ismine sahip iş istasyonu üzerinden bir hastanın dağıtık ağda sorgulanacağını varsayalım. Sorgulama işlemi görüntüleyici iş istasyonunun sunduğu Şekil 5.17'deki gibi bir sorgulama ekranı ile yapılır. Sorgulama ekranına hastanın PACS arşivlerinden bulunabilmesi için gerekli alanlar girilir. Dağıtık bir PACS'ta ise hastanın tekil tanımlayıcısını (identifier – ID) girmek oldukça mantıklıdır. Bu tarz bir dağıtık PACS sistemine bağlı PACS arşivlerinde kullanılacak tekil tanımlayıcı ise hastanın ulusal kimlik numarası olmalıdır. Şekil 5.17'deki örnekte 12345678901 kimlik numaralı hasta dağıtık PACS ağı üzerinden sorgulanmak istenmektedir.



Şekil 5.17 ClearCanvas PACS Görüntüleyici Sorgulama Ekranı

Hastayı bulabilmek için gerekli parametreler girildikten sonra sorgulama işlemi başlatılır. Şekil 5.18'de örnek bir sorgu sonucu gösterilmektedir. Hasta hakları yüzünden şekillerdeki sorgu sonuçlarında bulunan hasta isim ve tanımlayıcıları gizlenmiştir.

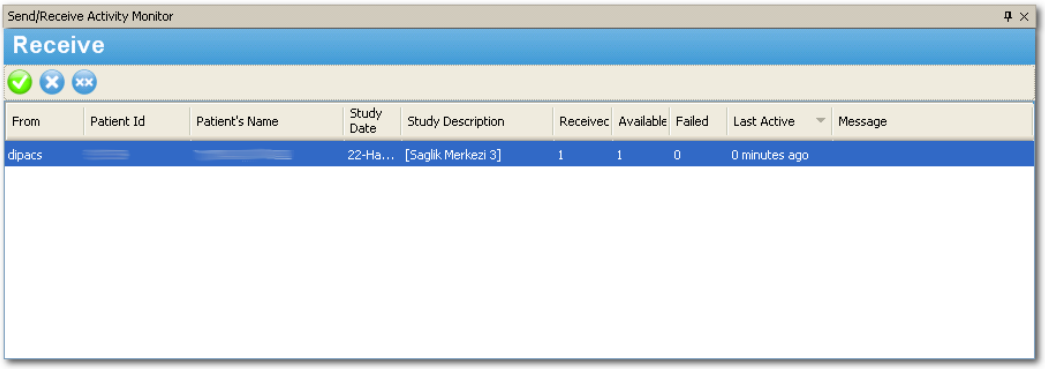


Şekil 5.18 Sorgulama Sonuçlarının Listesi

Şekil 5.18'de görüldüğü gibi iki farklı dağıtık PACS sunucusundan da görüntüler dönmüştür. Sorguyu diğer dağıtık PACS sunucularına ileten sunucu, gelen sonuçların Description alanlarına sağlık merkezlerinin isimlerini köşeli parantezler içerisine koyarak eklemektedir. Bu sayede görüntünün hangi sağlık merkezinden geldiği kolayca görülebilir. Birçok PACS görüntüleyici dönen sonuçların Description bölümünü sonuçlar ile birlikte listelediği için bu alan seçilmiştir.

5.3.3. Görüntü çağırma

Sorgulama işlemi ile aranan hasta ve görüntü bulunduktan sonra görüntüleyici yazılım aracılığı ile görüntü dağıtık PACS sunucusu üzerinden çağırılır. Sağlık merkezi içerisindeki dağıtık PACS sunucusu çağrılmak istenen görüntünün hangi sunucuda bulunduğunu sorgulama işlemi ile öğrendiği için ilgili dağıtık sunucu ile iletişime geçerek görüntüyü çağırır. Şekil 5.18'de görülen sonuçlardan istenilen çalışma seçilerek görüntü dağıtık sunucudan çağırılır. Görüntü Şekil 5.19'da görüldüğü gibi önce iş istasyonunun diskine depolanır.



Şekil 5.19 Görüntünün Çağrılması

Görüntü aktarımı tamamlandıktan sonra görüntü açılır. Şekil 5.20'de dağıtık PACS sunucusu üzerinden çekilen görüntünün iş istasyonunda nasıl görüntülendiği görülebilir.

Şekil 5.20 Dağıtık PACS Sunucusu Üzerinden Gelen Görüntünün Görüntülenmesi

6. SONUÇ

Tez projesi kapsamında, RMI teknolojisi ve DICOM standardı kullanılarak sağlık merkezleri içerisinde bulunan PACS arşivlerindeki tıbbi görüntülere erişim sağlayan bir dağıtık PACS sistemi geliştirilmiştir.

Birçok sağlık merkezinde kullanılmakta olan PACS ile sağlık merkezleri sınırları içerisinde kalan tıbbi görüntüler, dağıtık PACS sunucuları ile sistemi kullanan tüm kullanıcılara ulaştırılabilmektedir. Sağlık merkezleri arasında kurulan bu dağıtık görüntü ağı ile görüntülere tek bir sistem üzerinden güvenli ve kolay bir şekilde erişilmesine olanak sağlanmaktadır.

Daha önce de bahsedildiği gibi; PACS içerisindeki görüntüleyici iş istasyonları, görüntü arşivleri ve görüntüleme cihazları DICOM standardı ile iletişim sağlar. Dağıtık PACS sunucularının diğer sistemler ile iletişiminde DICOM standardının kullanılması sayesinde sağlık merkezlerinde bulunan DICOM uyumlu tüm PACS birimleri hiçbir yazılım veya donanım maliyetine gerek kalmaksızın kolay bir şekilde yapılandırılarak sisteme dahil olabilirler.

Ayrıca dağıtık sunucu yazılımı sadece sorgu ve görüntü aktarımı yaptığı için yüksek kapasiteli bir disk ünitesine ya da gelişmiş bir sunucuya gereksinim duymaz. İnternet bağlantısı olan ve JRE 6'nın sistem gereksinimlerini karşılayan herhangi bir bilgisayar üzerinde dahi çalışabilmektedir.

Dağıtık PACS sistemine alternatif olarak merkezi bir sunucu ile benzer bir sistem geliştirilebilir. Fakat sağlık merkezlerinde çekim yapıldıktan sonra, her görüntü sağlık merkezi içerisinde bulunan PACS arşivinin yanı sıra bu merkezi sunucuya da gönderilmelidir. Günümüzde İnternet bağlantıları DSL'in yaygınlaşması sonucu her ne kadar hızlanmış olsa da, yüksek çözünürlüklü tıbbi görüntülerin gönderimleri çok uzun sürecek ya da sisteme dahil olmak isteyen her sağlık merkezi daha yüksek bant genişliğine sahip bir İnternet bağlantısına sahip olması gerekecektir. Ayrıca birçok sağlık merkezinden gelecek görüntüleri uzun süre saklayabilmek için merkezi sunucunun çok yüksek kapasitede disk ünitelerine gereksinimi olacaktır. Disk gereksinimi haricinde, birçok sağlık merkezinden gelebilecek görüntü isteklerini karşılayabilmek için çok yüksek bant genişliği gerekecektir. Birçok sağlık merkezinin PACS sahibi olduğu günümüzde, arşivlerin dağıtık tek bir sistem üzerinden erişilebilir olması ile maliyet gereksinimi çok yüksek oranda azaltılmış olur.

7. ÖNERİLER

Hastalara ait görüntüler birçok sağlık merkezi arasında aktarılacağı için hasta bilgilerini izinsiz olarak kullanacak kişilerden korumak gerekmektedir. Kısacası kullanıcılar görüntülere sadece hastaların izin vermesi durumunda erişebilmelidir. Yani bir denetleme mekanizması gerekmektedir.

Birçok sağlık merkezinde hastalara ait görüntüler PACS arşivlerinde RIS tarafından üretilen tekil hasta numaraları ile eşleştirilmiştir. Sağlık merkezleri bünyesinde türetilen tekil numaralar diğer sağlık merkezlerinde kullanılan numaralar ile aynı olabilir. Eğer dağıtık sistemde bu numaralar kullanılmaya kalkılırsa, sorgularda birçok hasta birbiri ile çakışacaktır. Bu yüzden hasta görüntülerine ulaşmak için hastanın tekil kimlik numarası (T.C. kimlik numarası) kullanılmalıdır. Fakat birçok PACS arşivi, görüntüyü hastanın tekil kimlik numarası ile tutmamaktadır. Görüntüleri tekil kimlik numarası ile sorgulayabilmek için sağlık merkezlerinde bulunan bir yazılımın, tekil kimlik numarasını PACS arşivinde kullanılan numaraya çevirmesi gerekir. Daha sonra dağıtık PACS sunucusu, elde ettiği bu numara ile PACS arşivini sorgulayabilir.

MR, BT, röntgen gibi birçok görüntüleme cihazı (modalite) olmasına karşın sağlık merkezlerinde bu cihazlardan sadece birkaçı bulunmaktadır. Sağlık merkezlerinin erişim bilgileri ile birlikte modalite tiplerinin bilgileri de bilinirse sorguların belirli dağıtık PACS sunucularına gönderilmesi sağlanarak sorgu süresi kısaltılabilir. Örneğin; bir ortopedi doktoru hastanın geçmiş anjio görüntüsünü sorgu sonucunda görmek istemez. Modalite seçimi zorunlu hale getirilerek sorguların daha kısa sürede sonuçlanması sağlanabilir.

Dağıtık PACS sunucularına görüntü işleme özellikleri kazandırılarak PACS arşivlerinde bulunan görüntüleri belirli kategorilere ayırması sağlanabilir. Bu sayede doktorların araştırmalarında kullanabilecekleri farklı bir sistem oluşturulabilir. Örneğin; “kırık ve Röntgen” sorgusu ile tüm dağıtık ağdaki kırık bulunan röntgen görüntülerini sorgulayabilmek sağlanabilir. Fakat bu tip sorgulamalar yapabilmek için özel görüntüleyici yazılımların geliştirilmesi gerekmektedir.

Ayrıca dağıtık PACS sunucuları arasındaki iletişimde verileri sıkıştırarak göndermek kullanıcıların daha az beklemesini sağlayacaktır. Özellikle düşük bant genişliğine sahip ağlarda yapılan görüntü aktarımlarında kullanılacak sıkıştırma

algoritmaları çok büyük önem arz etmektedir. Geniş çözünürlüğe sahip tıbbi görüntülerin aktarımını daha hızlı sağlayabilmek için gelecek çalışmalarda özellikle sıkıştırma işleminin üzerinde durulması planlanmaktadır.

Tez kapsamında daha çok dağıtık PACS ağının kurulması için gerekli iletişim altyapısı üzerine yoğunlaşılmıştır. Yukarıda bahsedilen fikirler dağıtık PACS sistemin iyileştirilmesi için yapılacak çalışmalar olarak belirlenmiştir.

KAYNAKLAR DİZİNİ

Apache, What is log4j?, http://logging.apache.org/log4j/1.2/index.html (Erişim Tarihi: 12 Ağustos 2009)

Balasingham, I., Ihlen, H., Leister, W., Roe, P., Samset, E., 2007, Communication of Medical Images, Text, and Messages in Inter-Enterprise Systems: A Case Study in Norway, Information Technology in Biomedicine, IEEE Transactions on , vol.11, no.1, 7-13pp

Becker, T., Simon, R., 2001, Digital image data transmission over long distances using image compression and broadband ISDN, Computers in Cardiology 2001 , 687-689pp

dcm4che2, dcm4che2 DICOM Toolkit, http://www.dcm4che.org/confluence/display/d2/dcm4che2+DICOM+Toolkit (Erişim Tarihi: 30 Ağustos 2009)

Dreyer, K.J., Hirschorn, D.S., Thrall, J.H., and Mehta, A., 2005, PACS: A Guide to the Digital Revolution (2nd ed.), Springer, 15-21pp

Gutierrez, M.A., Furuie, S.S., Carvalho, T.C., Ruggiero, W.V., Figueiredo, J.C.B., Yamaguti, M., Pilon, P.E., Paiva, P.B., Lopes, P., Sigulem, D., 1999, A superhighway network to exchange cardiac images in a metropolitan area, Computers in Cardiology 1999, 33-36pp

Gülcü, C., 2002, Short introduction to log4j, Apache Software Foundation, http://logging.apache.org/log4j/1.2/manual.html

Kara, S., Üstündağ S., 2002, Dağıtık Programlama, Ege Üniversitesi Bilgisayar Mühendisliği, http://ndn.netsis.com.tr/GokselUCER/Master/dagitikprogramlama.doc , 12-77

Khludov, S., Meinel, C., Noelle, G., Warda, F., 1999, PACS for teleradiology, Computer-Based Medical Systems, Proceedings, 12th IEEE Symposium on, 6-11pp

Khludov, S., Vorwerk, L., Meinel, C., 2000, Internet-orientated medical information system for DICOM-data transfer, visualization and revision, Computer-Based Medical Systems, CBMS 2000, Proceedings. 13th IEEE Symposium on , 293-296pp

Koutelakis, G.V., Lymperopoulos, D.K., 2006, PACS through Web Compatible with DICOM Standard and WADO Service: Advantages and Implementation, Engineering in Medicine and Biology Society, EMBS '06. 28th Annual International Conference of the IEEE , 2601-2605pp

Morales, M.A., Dalmiani, S., Mazzarisi, A., Marcheschi, P., Glauber, M., Bevilacqua, S., Carpeggiani, C., 2004, Cardiac surgery and cardiological data integration between remote structures, Computers in Cardiology, 2004, 593-596pp

NEMA Commitee, 2008, Digital Imaging and Communications in Medicine (DICOM) Standard - Version 3.0, National Electrical Manufacturers Association, PS3.1, PS3.3, PS3.4, PS3.6, PS3.7, PS3.8

Neri, E., Thiran, J.P., Caramella, D., Petri, C., Bartolozzi, C., Piscaglia, B., Macq, B., Duprez, T., Cosnard, G., Maldague, B., De Pauw, J., 1998, Interactive DICOM image transmission and telediagnosis over the European ATM network, Information Technology in Biomedicine, IEEE Transactions on , vol.2, no.1, 35-38pp

Pianykh, O.S., 2008. Digital Imaging and Communications in Medicine: A Practical Introduction and Survival Guide, Springer, 29-81pp, 115-166pp, 179-206pp

Sun Microsystems, Java RMI Whitepaper, http://java.sun.com/javase/technologies/core/basic/rmi/whitepaper/index.jsp (Erişim Tarihi: 4 Temmuz 2009)

Tanenbaum, A.S. and Steen, M., 2002, Distributed Systems: Principles and Paradigms, Prentice-Hall Inc, 1-16pp, 183-238pp

Thomas, S.A., 2000, SSL & TLS Essentials: Securing the Web, Wiley, 1-37pp

Tohme, W.G., Choi, I., Vasilescu, E., Mun, S.K., 2006, The Evolution of Distributed Diagnosis: Teleradiology As a Case Study, Distributed Diagnosis and Home Healthcare, 1st Transdisciplinary Conference on , 113-115 pp

Wahle, A., Builtjes, J.H., Oswald, H., Fleck, E., 1996, Secure inter-institutional image communication by using DICOM-to-DICOM gateways, Computers in Cardiology 1996, 309-312pp

Wikipedia, Picture Archiving and Communication System, http://en.wikipedia.org/wiki/Picture_archiving_and_communication_system (Erişim Tarihi: 10 Haziran 2009)

Wikipedia, Transport Layer Security, http://en.wikipedia.org/wiki/Transport_Layer_Security (Erişim Tarihi: 5 Ağustos 2009)

Wikipedia, log4j, http://en.wikipedia.org/wiki/Log4j (Erişim Tarihi: 10 Ağustos 2009)

Zhang J., Yu F., Sun J., Yang Y., Liang C., 2007, DICOM Image Secure Communications With Internet Protocols IPv6 and IPv4, Information Technology in Biomedicine, IEEE Transactions on , vol.11, no.1, 70-80pp

EKLER

Ek 1 Çizelgeler

Ek 1.1. Depolama SOP Sınıfları

Çizelge Ek 1.1 Depolama SOP Sınıfları

SOP Sınıfı	SOP Class UID
Computed Radiography Image Storage	1.2.840.10008.5.1.4.1.1.1
Digital X-Ray Image Storage - For Presentation	1.2.840.10008.5.1.4.1.1.1.1
Digital X-Ray Image Storage - For Processing	1.2.840.10008.5.1.4.1.1.1.1.1
Digital Mammography Image Storage - For Presentation	1.2.840.10008.5.1.4.1.1.1.2
Digital Mammography Image Storage - For Processing	1.2.840.10008.5.1.4.1.1.1.2.1
Digital Intra-oral X-Ray Image Storage - For Presentation	1.2.840.10008.5.1.4.1.1.1.3
Digital Intra-oral X-Ray Image Storage - For Processing	1.2.840.10008.5.1.4.1.1.1.3.1
CT Image Storage	1.2.840.10008.5.1.4.1.1.2
Enhanced CT Image Storage	1.2.840.10008.5.1.4.1.1.2.1
Ultrasound Multi-frame Image Storage	1.2.840.10008.5.1.4.1.1.3.1
MR Image Storage	1.2.840.10008.5.1.4.1.1.4
Enhanced MR Image Storage	1.2.840.10008.5.1.4.1.1.4.1
MR Spectroscopy Storage	1.2.840.10008.5.1.4.1.1.4.2
Ultrasound Image Storage	1.2.840.10008.5.1.4.1.1.6.1
Secondary Capture Image Storage	1.2.840.10008.5.1.4.1.1.7
Multi-frame Single Bit Secondary Capture Image Storage	1.2.840.10008.5.1.4.1.1.7.1
Multi-frame Grayscale Byte Secondary Capture Image Storage	1.2.840.10008.5.1.4.1.1.7.2
Multi-frame Grayscale Word Secondary Capture Image Storage	1.2.840.10008.5.1.4.1.1.7.3
Multi-frame True Color Secondary Capture Image Storage	1.2.840.10008.5.1.4.1.1.7.4
12-lead ECG Waveform Storage	1.2.840.10008.5.1.4.1.1.9.1.1
General ECG Waveform Storage	1.2.840.10008.5.1.4.1.1.9.1.2
Ambulatory ECG Waveform Storage	1.2.840.10008.5.1.4.1.1.9.1.3
Hemodynamic Waveform Storage	1.2.840.10008.5.1.4.1.1.9.2.1
Cardiac Electrophysiology Waveform Storage	1.2.840.10008.5.1.4.1.1.9.3.1
Basic Voice Audio Waveform Storage	1.2.840.10008.5.1.4.1.1.9.4.1
Grayscale Softcopy Presentation State Storage	1.2.840.10008.5.1.4.1.1.11.1
Color Softcopy Presentation State Storage	1.2.840.10008.5.1.4.1.1.11.2
Pseudo-Color Softcopy Presentation State Storage	1.2.840.10008.5.1.4.1.1.11.3
Blending Softcopy Presentation State Storage	1.2.840.10008.5.1.4.1.1.11.4

X-Ray Angiographic Image Storage	1.2.840.10008.5.1.4.1.1.12.1
Enhanced XA Image Storage	1.2.840.10008.5.1.4.1.1.12.1.1
X-Ray Radiofluoroscopic Image Storage	1.2.840.10008.5.1.4.1.1.12.2
Enhanced XRF Image Storage	1.2.840.10008.5.1.4.1.1.12.2.1
X-Ray 3D Angiographic Image Storage	1.2.840.10008.5.1.4.1.1.13.1.1
X-Ray 3D Craniofacial Image Storage	1.2.840.10008.5.1.4.1.1.13.1.2
Nuclear Medicine Image Storage	1.2.840.10008.5.1.4.1.1.20
Raw Data Storage	1.2.840.10008.5.1.4.1.1.66
Spatial Registration Storage	1.2.840.10008.5.1.4.1.1.66.1
Spatial Fiducials Storage	1.2.840.10008.5.1.4.1.1.66.2
Deformable Spatial Registration Storage	1.2.840.10008.5.1.4.1.1.66.3
Segmentation Storage	1.2.840.10008.5.1.4.1.1.66.4
Real World Value Mapping Storage	1.2.840.10008.5.1.4.1.1.67
VL Endoscopic Image Storage	1.2.840.10008.5.1.4.1.1.77.1.1
Video Endoscopic Image Storage	1.2.840.10008.5.1.4.1.1.77.1.1.1
VL Microscopic Image Storage	1.2.840.10008.5.1.4.1.1.77.1.2
Video Microscopic Image Storage	1.2.840.10008.5.1.4.1.1.77.1.2.1
VL Slide-Coordinates Microscopic Image Storage	1.2.840.10008.5.1.4.1.1.77.1.3
VL Photographic Image Storage	1.2.840.10008.5.1.4.1.1.77.1.4
Video Photographic Image Storage	1.2.840.10008.5.1.4.1.1.77.1.4.1
Ophthalmic Photography 8 Bit Image Storage	1.2.840.10008.5.1.4.1.1.77.1.5.1
Ophthalmic Photography 16 Bit Image Storage	1.2.840.10008.5.1.4.1.1.77.1.5.2
Stereometric Relationship Storage	1.2.840.10008.5.1.4.1.1.77.1.5.3
Ophthalmic Tomography Image Storage	1.2.840.10008.5.1.4.1.1.77.1.5.4
Basic Text SR	1.2.840.10008.5.1.4.1.1.88.11
Enhanced SR	1.2.840.10008.5.1.4.1.1.88.22
Comprehensive SR	1.2.840.10008.5.1.4.1.1.88.33
Procedure Log	1.2.840.10008.5.1.4.1.1.88.40
Mammography CAD SR	1.2.840.10008.5.1.4.1.1.88.50
Key Object Selection	1.2.840.10008.5.1.4.1.1.88.59
Chest CAD SR	1.2.840.10008.5.1.4.1.1.88.65
X-Ray Radiation Dose SR	1.2.840.10008.5.1.4.1.1.88.67
Encapsulated PDF Storage	1.2.840.10008.5.1.4.1.1.104.1
Positron Emission Tomography Image Storage	1.2.840.10008.5.1.4.1.1.128
RT Image Storage	1.2.840.10008.5.1.4.1.1.481.1
RT Dose Storage	1.2.840.10008.5.1.4.1.1.481.2
RT Structure Set Storage	1.2.840.10008.5.1.4.1.1.481.3
RT Beams Treatment Record Storage	1.2.840.10008.5.1.4.1.1.481.4
RT Plan Storage	1.2.840.10008.5.1.4.1.1.481.5
RT Brachy Treatment Record Storage	1.2.840.10008.5.1.4.1.1.481.6

RT Treatment Summary Record Storage	1.2.840.10008.5.1.4.1.1.481.7
RT Ion Plan Storage	1.2.840.10008.5.1.4.1.1.481.8
RT Ion Beams Treatment Record Storage	1.2.840.10008.5.1.4.1.1.481.9

Ek 1.2. Soyut Söz Dizimleri

Çizelge Ek 1.2 Önemli Soyut Söz Dizimleri

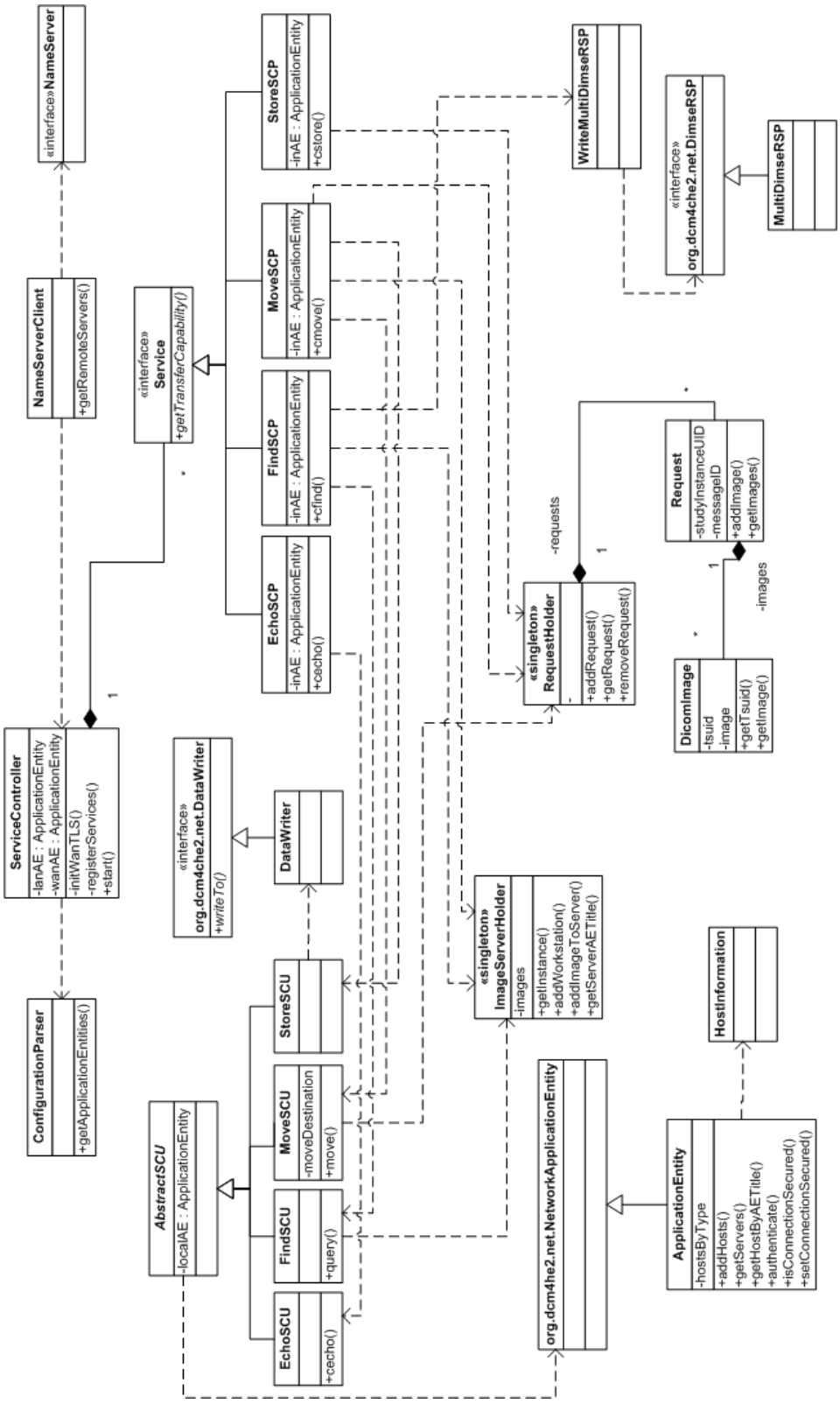
Soyut söz dizimi ismi	**Abstract Syntax UID**
Doğrulama	
Verification	1.2.840.10008.1.1
Sorgulama ve Görüntü Çağırma	
Study Root Query/Retrieve Information Model – FIND	1.2.840.10008.5.1.4.1.2.2.1
Study Root Query/Retrieve Information Model – GET	1.2.840.10008.5.1.4.1.2.2.3
Study Root Query/Retrieve Information Model – MOVE	1.2.840.10008.5.1.4.1.2.2.2
Patient Root Query/Retrieve Information Model – FIND	1.2.840.10008.5.1.4.1.2.1.1
Patient Root Query/Retrieve Information Model – GET	1.2.840.10008.5.1.4.1.2.1.3
Patient Root Query/Retrieve Information Model – MOVE	1.2.840.10008.5.1.4.1.2.1.2
Depolama	
CR Image Storage	1.2.840.10008.5.1.4.1.1.1
CT Image Storage	1.2.840.10008.5.1.4.1.1.2
MR Image Storage	1.2.840.10008.5.1.4.1.1.4
Ultrasound Image Storage	1.2.840.10008.5.1.4.1.1.6.1
NM Image Storage	1.2.840.10008.5.1.4.1.1.20
PET Image Storage	1.2.840.10008.5.1.4.1.1.128
Yazdırma	
Basic Film Session Class	1.2.840.10008.5.1.1.1
Basic Film Box Class	1.2.840.10008.5.1.1.2
Basic Grayscale Image Box Class	1.2.840.10008.5.1.1.4

Ek 1.3. Transfer Söz Dizimleri

Çizelge Ek 1.3 Önemli Transfer Söz Dizimleri

Transfer söz dizimi ismi	**Transfer Syntax UID**
Byte sıraları	
Implicit VR Little Endian	1.2.840.10008.1.2
Explicit VR Little Endian	1.2.840.10008.1.2.1
Implicit VR Big Endian	1.2.840.10008.1.2.2
Görüntü sıkıştırma biçimleri	
DICOM Explicit JPEG baseline 8-bit lossy compression	1.2.840.10008.1.2.4.50
DICOM Explicit JPEG baseline 12-bit lossy compression	1.2.840.10008.1.2.4.51
DICOM Explicit JPEG baseline lossless compression	1.2.840.10008.1.2.4.57
DICOM JPEG-LS lossless compression	1.2.840.10008.1.2.4.80
DICOM JPEG-LS near-lossless compression	1.2.840.10008.1.2.4.81
DICOM JPEG2000 lossless compression	1.2.840.10008.1.2.4.90
DICOM JPEG2000 lossy compression	1.2.840.10008.1.2.4.91
...	...

Ek 2 Sistemin UML Sınıf Diyagramı



Şekil Ek 2.1 Sistemin UML Sınıf Diyagramı

Ek 3 Sınıf gerçekleştirimleri

Ek 3.1. HostInformation Sınıfı

```
package com.emotlab.dipacs.common;

import java.io.Serializable;

public class HostInformation implements Serializable{
	/**
	 *
	 */
	private static final long serialVersionUID =
3848634002715389925L;

	private String title;
	private String aeTitle;
	private String hostname;
	private int port;

	public HostInformation(){ }

	public HostInformation(String aeTitle, String hostname, int
port) {
		this.setAETitle(aeTitle);
		this.setHostname(hostname);
		this.setPort(port);
	}

	public void setTitle(String title) {
		this.title = title;
	}

	public String getTitle() {
		return title;
	}

	public void setAETitle(String aeTitle) {
		this.aeTitle = aeTitle;
	}

	public String getAeTitle() {
		return aeTitle;
	}

	public void setHostname(String hostname) {
		this.hostname = hostname;
```

```
	}

	public String getHostname() {
		return hostname;
	}

	public void setPort(int port) {
		this.port = port;
	}

	public int getPort() {
		return port;
	}
}
```

Ek 3.2. ServiceController Sınıfı

```
package com.emotlab.dipacs.communication;

import java.io.FileInputStream;
import java.io.IOException;
import java.security.GeneralSecurityException;
import java.security.KeyStore;
import java.util.Vector;
import java.util.concurrent.Executor;

import org.dcm4che2.net.Device;
import org.dcm4che2.net.NetworkConnection;
import org.dcm4che2.net.NewThreadExecutor;
import org.dcm4che2.net.TransferCapability;
import org.dcm4che2.net.service.DicomService;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import com.emotlab.dipacs.common.CommonEnums.AEType;
import com.emotlab.dipacs.dicom.EchoSCP;
import com.emotlab.dipacs.dicom.FindSCP;
import com.emotlab.dipacs.dicom.MoveSCP;
import com.emotlab.dipacs.dicom.Service;
import com.emotlab.dipacs.dicom.StoreSCP;
import com.emotlab.dipacs.remote.NameServerClient;
import com.emotlab.dipacs.support.ConfigurationParser;

public class ServiceController {
```

```
	static Logger log =
LoggerFactory.getLogger(ServiceController.class);


	private static final String KEYSTORE_PASSWORD =
"j?pZO|p$f&Sp+3$A";
	private static final String TRUSTSTORE_PASSWORD =
"m8+/rc#|!bMn@mZ*";


	private static final String KEY_PASSWORD =
",`EF.e4!L|4F'5gx";


	private Device lanDevice = new Device("diPACS-DEVICE-LAN");
	private Device wanDevice = new Device("diPACS-DEVICE-WAN");


	private Executor lanExecutor = new
NewThreadExecutor("diPACS-THREAD-LAN");
	private Executor wanExecutor = new
NewThreadExecutor("diPACS-THREAD-WAN");


	private NetworkConnection lanConnection = new
NetworkConnection();
	private NetworkConnection wanConnection = new
NetworkConnection();


	private static ApplicationEntity lanAE = new
ApplicationEntity();
	private static ApplicationEntity wanAE = new
ApplicationEntity();


	private static final Service[] lanServices = new Service[] {
new EchoSCP(lanAE) , new FindSCP(lanAE, wanAE), new
StoreSCP(lanAE), (Service) new MoveSCP(lanAE, wanAE)};
	private static final Service[] wanServices = new Service[] {
new EchoSCP(wanAE) , new FindSCP(wanAE, lanAE), new
StoreSCP(wanAE), (Service) new MoveSCP(wanAE, lanAE)};


	public ServiceController(String lanAETitle, int lanPort,
String wanAETitle, int wanPort){
		prepareService(lanAETitle, lanPort, lanDevice, lanAE,
lanConnection, lanServices);
		prepareService(wanAETitle, wanPort, wanDevice, wanAE,
wanConnection, wanServices);


		initWanTLS();


		ConfigurationParser parser = new
ConfigurationParser("configuration.xml");

	lanAE.addHosts(parser.getApplicationEntities(AEType.WORKSTAT
ION), AEType.WORKSTATION);
```

```
      lanAE.addHosts(parser.getApplicationEntities(AEType.SERVER),
AEType.SERVER);

            NameServerClient nsClient = new NameServerClient();
            wanAE.addHosts(nsClient.getRemoteServers(),
AEType.SERVER);
      }

      private void prepareService(String aeTitle, int port, Device
device, ApplicationEntity ae, NetworkConnection connection,
Service[] services){
            device.setNetworkApplicationEntity(ae);
            device.setNetworkConnection(connection);

            ae.setNetworkConnection(connection);
            ae.setAssociationAcceptor(true);

            connection.setPort(port);

            ae.setAETitle(aeTitle);

            registerServices(ae, services);
      }

      private void initWanTLS(){
            wanAE.setAsWanInterface(true);

            wanConnection.setTls3DES_EDE_CBC();
            wanConnection.setTlsNeedClientAuth(true);

            try{
                  KeyStore keyStore =
loadKeyStore("../tls/node.jks", KEYSTORE_PASSWORD.toCharArray());
              KeyStore trustStore =
loadKeyStore("../tls/trust.jks",
TRUSTSTORE_PASSWORD.toCharArray());
              wanDevice.initTLS(keyStore,
KEY_PASSWORD.toCharArray(), trustStore);
            }catch(GeneralSecurityException e){
                  e.printStackTrace();
            } catch (IOException e) {
                  e.printStackTrace();
            }

            log.debug("Secure connection enabled with TLS
configuration on {}:{}", wanAE.getAETitle(),
wanConnection.getPort());
```

```
	}

	private KeyStore loadKeyStore(String url, char[] password)
throws GeneralSecurityException, IOException {
		KeyStore key = KeyStore.getInstance("JKS");

		FileInputStream in = new FileInputStream(url);
		try {
		    key.load(in, password);
		} finally {
		    in.close();
		}
		return key;
	}

	private void registerServices(ApplicationEntity ae,
Service[] services){
		Vector<TransferCapability> tcList = new
Vector<TransferCapability>();

		for (int i = 0; i < services.length; i++) {
			Service scp = services[i];
			ae.register((DicomService) scp);

			tcList.addAll(scp.getTransferCapability());
		}

		TransferCapability[] tc = new
TransferCapability[tcList.size()];
		for (int i = 0; i < tcList.size(); i++) {
			tc[i] = tcList.get(i);
		}

		ae.setTransferCapability(tc);
	}

	public void start(){
		try {
			lanDevice.startListening(lanExecutor);
			wanDevice.startListening(wanExecutor);

			System.out.println(lanAE.getAETitle() + "
Started listening on LAN on port : " + lanConnection.getPort());
			System.out.println(wanAE.getAETitle() + "
Started listening on WAN on port : " + wanConnection.getPort());
		} catch (IOException e) {
			e.printStackTrace();
```

```
            }
      }
}
```

Ek 3.3. ImageServerHolder Sınıfı

```
package com.emotlab.dipacs.communication;

import java.util.concurrent.ConcurrentHashMap;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class ImageServerHolder {
      static Logger log =
LoggerFactory.getLogger(ImageServerHolder.class);

      private final static ImageServerHolder instance = new
ImageServerHolder();

      private ConcurrentHashMap<String, ConcurrentHashMap<String,
String>> images = new ConcurrentHashMap<String,
ConcurrentHashMap<String,String>>();

      public static ImageServerHolder getInstance(){
            return instance;
      }

      public void addWorkstation(String workstationAETitle){
            if(images.containsKey(workstationAETitle)){
                  images.get(workstationAETitle).clear();
            }else{
                  images.put(workstationAETitle, new
ConcurrentHashMap<String, String>());
            }

            log.debug("Workstation is added to the ImageServer
list : {} ", workstationAETitle);
      }

      public void addImageToServer(String workstationAETitle,
String studyInstanceUID, String serverAETitle) throws
ObjectNotFoundException{
            if(images.containsKey(workstationAETitle)){

      images.get(workstationAETitle).put(studyInstanceUID,
serverAETitle);

                  log.debug("Image [{}] is connected with server
[{}] and added to the workstation list [{}] ", new Object[]{
studyInstanceUID, serverAETitle, workstationAETitle });
            }else{
                  throw new ObjectNotFoundException();
            }
      }

      public String getServerAETitle(String workstationAETitle,
String studyInstanceUID){
```

```
            return
images.get(workstationAETitle).get(studyInstanceUID);
        }
}
```

Ek 3.4. MultiDimseRSP Sınıfı

```
package com.emotlab.dipacs.dicom;

import java.util.List;

import org.dcm4che2.data.DicomObject;
import org.dcm4che2.data.Tag;
import org.dcm4che2.data.VR;
import org.dcm4che2.net.Association;
import org.dcm4che2.net.CommandUtils;
import org.dcm4che2.net.DimseRSP;

public class MultiDimseRSP implements DimseRSP
{
    private int count = -1;
    private final DicomObject cmd;
    private final List<DicomObject> data;

    public MultiDimseRSP(DicomObject cmd)
    {
        this(cmd, null);
    }

    public MultiDimseRSP(DicomObject cmd, List<DicomObject> data)
    {
        this.cmd = cmd;
        this.data = data;
    }

    public synchronized boolean next()
    {
        return count++ < data.size();
    }

    public final DicomObject getCommand()
    {
      if (count == data.size()){
            cmd.putInt(Tag.Status, VR.US, CommandUtils.SUCCESS);
      }

      return cmd;
    }

    public final DicomObject getDataset()
    {
      if (count == data.size()){
            return null;
      }

      data.get(count).putString(Tag.RetrieveAETitle, null,
"diPACS");
        return data.get(count);
    }
```

```
    public void cancel(Association a)
    {
        // NOOP
    }
}
```

Ek 3.5. WriteMultiDimseRSP Sınıfı

```
package com.emotlab.dipacs.dicom;

import org.dcm4che2.net.Association;
import org.dcm4che2.net.DicomServiceException;
import org.dcm4che2.net.DimseRSP;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

class WriteMultiDimseRsp implements Runnable {
      static Logger log =
LoggerFactory.getLogger(WriteMultiDimseRsp.class);

      private final Association as;

    private final int pcid;

    private final DimseRSP rsp;

    public WriteMultiDimseRsp(Association as, int pcid, DimseRSP
rsp) {
        this.as = as;
        this.pcid = pcid;
        this.rsp = rsp;
    }

    public void run() {
        try {
            try {
                do{
                  as.writeDimseRSP(pcid, rsp.getCommand(),
rsp.getDataset());

                  log.debug("C-FIND-RSP message is sent to : {}",
as.getCallingAET());

                  log.trace("C-FIND-RSP Command : \r\n{}",
rsp.getCommand());
                        log.trace("C-FIND-RSP Data : \r\n{}",
rsp.getDataset());
                }while (rsp.next());

            } catch (DicomServiceException e) {
                as.writeDimseRSP(pcid, e.getCommand(),
e.getDataset());
            }
        } catch (Throwable e) {
            as.abort();
        }
    }

}
```

ÖZGEÇMİŞ

Tolga Utku Onbay 1983 yılında İstanbul'da doğdu. Lise öğrenimini İstanbul Süleyman Nazif Lisesinde 2001 senesinde bitirdikten sonra Yakın Doğu Üniversitesi Bilgisayar Mühendisliği bölümünde lisans öğrenimine başladı. 2003 senesinde yatay geçişle geldiği Ege Üniversitesi Bilgisayar Mühendisliği öğrenimini 2006 senesinde tamamladıktan sonra aynı bölümde yüksek lisans eğitimine devam etti. Ayrıca 2006 senesinde çalışmaya başladığı EMOT ve Atakalp Hastaneleri Bilgi Sistemleri Biriminde halen yazılım mühendisi ve proje yöneticisi olarak görev almaktadır.