İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

ETMEN TABANLI DAĞITILMIŞ OLAY SİSTEMİ

DOKTORA TEZİ

Y. Müh. Özgür Koray ŞAHİNGÖZ

Anabilim Dalı : KONTROL VE BİLGİSAYAR MÜHENDİSLİĞİ

Programı : KONTROL VE BİLGİSAYAR MÜHENDİSLİĞİ

MAYIS 2006

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

ETMEN TABANLI DAĞITILMIŞ OLAY SİSTEMİ

DOKTORA TEZİ

Y. Müh. Özgür Koray ŞAHİNGÖZ
(504982126)

Tezin Enstitüye Verildiği Tarih : 28 Ekim 2005
Tezin Savunulduğu Tarih : 18 Mayıs 2006

Tez Danışmanı :	**Prof.Dr. A. Coşkun SÖNMEZ**
Diğer Jüri Üyeleri	**Prof.Dr. Bülent ÖRENCİK (İ.T.Ü.)**
	Prof.Dr. H. Levent AKIN (B.Ü.)
	Yrd.Doç.Dr. D. Turgay ALTILAR (İ.T.Ü.)
	Yrd.Doç.Dr. A. Gökhan YAVUZ (Y.T.Ü.)

MAYIS 2006

Ö N S Ö Z

Günümüze kadar geliştirilen tüm olay sistemleri olay kavramını, veri kaynağı / veri üreticisi tarafından üretilen basit formatta bir veri olarak algılamışlar ve çalışmalarını üretilen bu verinin, uygun formatta gösterilimi, işlenmesi ve hızlı şekilde yayılması üzerine yoğunlaştırmışlardır. Bu tez çalışmanın amacı ise: "Olay kavramının üretim niteliğini geliştirerek, alışılagelen sadece ham veri üretiminin yanı sıra, eylemlerin (action) de üretildiği ileri bir düzeye yerleştirerek bu nitelikteki olayların üretilmesine, dağıtılmasına ve işlenmesine olanak sağlayan bir etmen tabanlı dağıtılmış olay sistemi altyapısı geliştirmektir".

Tez çalışması süresince bana yol gösteren, önerileri ve görüşleri ile tezin oluşmasını ve bilimsel niteliğinin yükselmesini sağlayan değerli hocam Prof.Dr. A. Coşkun SÖNMEZ'e teşekkür ederim.

Bana bu tez çalışması yapabilme imkanı sağlayan, mensubu olmakla her zaman gurur duyduğum, Hava Kuvvetleri Komutanlığı'na ve İstanbul Teknik Üniversitesi'ne de teşekkürü bir borç bilirim.

Son olarak, çalışmalarım esnasında benden hiçbir zaman desteğini esirgemeyen sevgili eşim Nazan ŞAHİNGÖZ'e sonsuz teşekkürlerimi sunarım.

Özgür Koray ŞAHİNGÖZ MAYIS 2006

İÇİNDEKİLER

KISALTMALAR **v**
TABLO LİSTESİ **vi**
ŞEKİL LİSTESİ **vii**
ÖZET **x**
SUMMARY **xv**

1. GİRİŞ **1**
1.1. Tezin Amacı 4
1.1.1. Etmenler 5
1.1.2. Kayıt/Yayım Haberleşme Modelinin Özellikleri 6
1.1.3. ABDES Sisteminin Özellikleri 7
1.1.4. Tezin Uygulama Alanları 9
1.1.4.1. Dağıtılmış Ortamda Gizli Bilgi (Şifre) Dağıtımı 9
1.1.4.2. Yazılım Güncelleme İşlemleri 9
1.1.4.3. Virüs Tanımlarının Güncellenmesi 10
1.1.4.4. Fiyat Ayarlaması 10
1.1.4.5. Dağıtılmış Geniş Çaplı Hesaplama Ortamı Sağlanması 10
1.1.4.6. Yayın Dağıtım/Bildirim Sistemleri 11
1.2. Tezin Organizasyonu 11

2. OLAY SİSTEMLERİ **13**
2.1. Dağıtılmış Olay Sistemi 14
2.2. Dağıtılmış Olay Sistemlerinin Tarihsel Gelişimi 15
2.3. Olay Tabanlı Haberleşme Düzeyleri. 16
2.4. Mesajlaşma Modeli 17
2.4.1. Uçtan Uca Haberleşme 17
2.4.2. Kayıt/Yayım Haberleşmesi 18
2.5. Kayıt Mekanizmaları 19
2.5.1. Kanal Tabanlı Kayıt/Yayım Sistemleri 20
2.5.2. Konu Tabanlı Kayıt/Yayım Sistemleri 21
2.5.2.1. Düz Konu Tabanlı Model 22
2.5.2.2. Hiyerarşik Konu Tabanlı Model 22
2.5.3. İçerik Tabanlı Kayıt/Yayım Sistemleri 23
2.5.4. Nesne-Tip Tabanlı Kayıt/Yayım Sistemleri 25

3. ETMENLER **26**
3.1. Etmen Örnekleri 28
3.2. Zeki Etmenler 30
3.3. Etmen mi? Nesne mi? 31
3.4. Gezgin Etmenler 32

3.4.1. Tarihsel Süreç 33
3.4.1.1. Uzaktan Yordam Çağırma 34
3.4.1.2. Uzaktan Değerlendirme 35
3.4.1.3. Talepte Kod Alma 35
3.4.1.4. Gezgin Etmen 36
3.4.2. Gezgin Etmen Kullanmanın Faydaları 37

4. KONUYLA İLGİLİ ÇALIŞMALAR 40
4.1. SIFT 40
4.2. ELVIN 41
4.3. Yeast/Ready/OmniNotify. 42
4.4. Java AWT: Delegasyon Olay Modeli. 42
4.5. CORBA Olay Servisi/CORBA Bildirim Servisi 43
4.6. JEDI 44
4.7. SIENA Olay Sistemi 45
4.8. GRYPHON 47
4.9. REBECA 48
4.10. LESUBSCRIBE 49
4.11. ECho/JECho 50
4.12. STEAM 50
4.13. DADI 50
4.14. Hermes 51
4.15. Fuego Olay Servisi 51
4.16. OBVENT 51

5. ABDES SİSTEMİ 54
5.1. ABDES Sisteminin Sahip Olduğu Özellikler 55
5.1.1. Özerklik 55
5.1.2. Tip/Agvent Tabanlı Kayıt 56
5.1.3. Bilgi Saklama 58
5.1.4. Kullanıcı Tanımlamalı Agvent Tipleri 59
5.1.5. Kendi Kendini Yönlendiren Agventler 59
5.1.6. Çift Taraflı Filtreleme 60
5.2. ABDES Sisteminin Altyapısı 61
5.3. Mesaj/Agvent Akışı 62
5.4. ABDES Sisteminin Bileşenleri 64
5.4.1. Yayımcı 64
5.4.2. Kayıtçı 66
5.4.3. Agvent Sunucu 68
5.5. Agventler 70
5.5.1. Agvent Sınıfının Yapısı 71
5.5.1.1. Agvent Sınıfının Temel Özellikleri 71
5.5.1.2. Agvent Sınıfının Temel Davranışları/Metotları 72
5.5.2. Bir Agvent Örneği 75
5.5.3. Agventin Yaşam Döngüsü 76
5.6. Duyurular 77
5.7. Kayıtlar 78
5.8. ABDES Sisteminde Filtreleme 79
5.8.1. Kayıtçının Filtrelemesi 80
5.8.2. Yayımcının Filtrelemesi 80

6. ABDES SİSTEMİNDE BİLGİ TABANI YÖNETİMİ 82
6.1. Kalıcı Nesneler 86
6.1.1. İlişkisel Veritabanları 87
6.1.2. Dosya Sistemleri 88
6.1.3. Nesne Veritabanları 89
6.1.4. Varlık Çekirdekleri 90
6.2. Java Veri Nesneleri 91
6.3. Kullanılan JDO Uygulama Ara yüzü 93

7. ABDES SİSTEMİNDE YÖNLENDİRME 96
7.1. Olay Sistemlerinde Yönlendirme 96
7.1.1. Olay Verilerinin Yayını 96
7.1.2. Kayıt Mesajlarının Yayını 97
7.1.3. Duyuruların Yayını 98
7.2. ABDES Sisteminde Yönlendirme 100
7.2.1. Duyuruların Yönlendirilmesi 101
7.2.2. Kayıt Mesajlarının Yönlendirilmesi 102
7.2.3. Agventlerin Yönlendirilmesi 102
7.3. Hata Kotarma 105
7.3.1. Duyuru Mesajlarının Dağıtımında Hata Kotarma 106
7.3.2. Kayıt Mesajlarının Dağıtımında Hata Kotarma 106
7.3.3. Agventlerin Dağıtımında Hata Kotarma 108

8. UYGULAMA ÖRNEĞİ 111
8.1. PublicationAgvent'in Tasarımı 114
8.1.1. Agvent Sınıfının Bir Alt Sınıfı Olarak Tanımlama 115
8.1.2. PublicationAgvent'in Sınıf Değişkenlerinin Belirlenmesi 116
8.1.3. PublicationAgvent'in Metotların Belirlenmesi 117
8.1.4. PublicationAgvent Duyurusunun Yapılması 119
8.1.5. Kayıtçıların Duyuru Listesini Almaları 120
8.1.6. Kayıt Mesajının Oluşturulması 121
8.1.7. Kayıtçı Üzerinde Haberleşme 122

9. TEST SONUÇLARI 125
9.1. Nesne Veri Tabanı İşlemleri 127
9.2. Dağıtım İşlemleri 130
9.2.1. Duyuru Mesajlarının Dağıtım İşlemleri 133
9.2.2. Kayıt Mesajlarının Dağıtım İşlemleri 134
9.2.3. Agvent Dağıtım İşlemleri 135

10. SONUÇ VE ÖNERİLER 138
10.1. ABDES Sisteminin Değerlendirmesi 139
10.2. Geliştirme Önerileri 141

KAYNAKLAR 144

ÖZGEÇMİŞ 154

KISALTMALAR

Agvent : Agent Event
RPC : Remote Procedure Call
SID : Selective Information Dissemination
URL : Uniform Resource Locator
OBS : Olay Bildirim Sistemi
JMS : Java Message Service
HOS : Hiyerarşik Olay Sistemi
AWT : Abstract Window Toolkit
PDA : Personal Digital Assitant
REV : Remote Evaluation
COD : Code on Demand
JVM : Java Virtual Machine
RMI : Remote Method Invocation
SIFT : Stanford Information Filtering Tool
DSTC : Distributed Systems and Technology Centre
Ready : REliable, Available, Distributed Yeast
GUI : Graphical User Interface
JDK : Java Developer Kit
CORBA : Common Object Request Broker Architecture
OMG : Object Management Group
QoS : Quality of Service
JEDI : Java Event Based Distributed Infrastructure
SIENA : Scalable Internet Event Notification Architecture
Obvent : Object Event
RDBMS : Relational Database Management System
XML : Extensible Markup Language
UML : Unified Modeling Language
AS : Agvent Sunucu
JDBC : Jaba Database Connectivity
SQL : Structured Query Language
ODBMS : Object Database Management Systems
J2EE : Java 2 Platform, Enterprise Edition
EJB : Enterprise JavaBeans
EJBQL : Enterprise JavaBean Query Language
JDO : Java Data Objects
ADM : Anahtar Dağıtım Merkezinden
JCE : Java Cryptography Extension
API : Application Programming Interface

TABLO LİSTESİ

Sayfa No

Tablo 3.1. Taşınabilir Kod Paradigmaları 34
Tablo 5.1. ABDES Sistemi Tarafından Desteklenen Özellik ve Davranış Tipleri ... 73
Tablo 5.2. ABDES Sistemi Tarafından Desteklenen Karşılaştırma Operatörleri 73
Tablo 6.1. JDO'nun Diğer Tekniklerle Karşılaştırılması 92
Tablo 6.2. JDO Uyumlu Yazılım Geliştiren Şirketler 93
Tablo 9.1. Test İşlemlerinde Kullanılan Sistem Özellikleri 126
Tablo 9.2. Üretilen Duyuruların Özellikleri 127
Tablo 9.3. Test Topolojilerindeki Atlama Miktarları 130
Tablo 9.4. Dağıtım Sürelerinin Hesaplanmasında Kullanılacak Ölçütler 132
Tablo 9.5. Duyuru Mesajlarına Yapılan Temel İşlemlerin Süreleri 133
Tablo 9.6. Kayıt Mesajlarına Yapılan Temel İşlemlerin Süreleri 134
Tablo 9.7. PublicationAgvent Boyutu Tablosu 135
Tablo 9.8. Agventlerin Dağıtımında Agvent Sunucu Üzerinde Yapılan İşlemler .. 136
Tablo 10.1. ABDES Sisteminin Diğer Olay Sistemleri ile Karşılaştırılması 140

ŞEKİL LİSTESİ

Sayfa No

Şekil 1.1 : Kayıt/Yayım Haberleşme Modeli 7
Şekil 2.1 : İstemci/Sunucu ve Olay Tabanlı Haberleşme Modelleri 13
Şekil 2.2 : Olay Tabanlı Haberleşme Düzeyleri 16
Şekil 2.3 : Olay Tabanlı Uçtan Uca Haberleşme Modeli 18
Şekil 2.4 : Kayıt/Yayım Haberleşme Modeli 19
Şekil 2.5 : Kanal Tabanlı Kayıt/Yayım Sistemi 21
Şekil 2.6 : Düz Konu Modeli 22
Şekil 2.7 : Hiyerarşik Konu Modeli 23
Şekil 2.8 : İçerik Tabanlı Kayıt/Yayım Sistemi 24
Şekil 3.1 : Fiziksel Ortamında Bir Etmenin Çalışması 27
Şekil 3.2 : Programlama Modelleri 31
Şekil 3.3 : Uzaktan Yordam Çağırma 34
Şekil 3.4 : Uzaktan Değerlendirme 35
Şekil 3.5 : Talepte Kod Alma 36
Şekil 3.6 : Gezgin Etmen 36
Şekil 4.1 : SIFT Mimarisi 41
Şekil 4.2 : JEDI Olay Sistemi 45
Şekil 4.3 : SIENA Olay Modelinin Yapısı 46
Şekil 4.4 : SIENA Olay Siteminde Olay Verisi ve Filtre Tanımlaması 47
Şekil 4.5 : GRYPHONda Olayların Karşılaştırma ağacından filtrelenmesi 48
Şekil 4.6 : REBECA Sisteminin Topolojisi 49
Şekil 5.1 : Olay Verisi Örneği 56
Şekil 5.2 : Kayıt Filtresi Örnekleri 57
Şekil 5.3 : Kayıt Sınıfının UML Diyagramı 58
Şekil 5.4 : ABDES Sisteminin Altyapısı 61
Şekil 5.5 : ABDES Sisteminde Kullanılan Temel Mesajlaşma Modeli 62
Şekil 5.6 : Yayımcı Altsistemi Yapısı 64
Şekil 5.7 : Kayıtçı Altsisteminin Yapısı 66
Şekil 5.8 : Bir Agvent Sunucusunun İç Yapısı 68
Şekil 5.9 : Agvent Sınıfının UML Diyagramı 71
Şekil 5.10 : Bir Agvent Altsınıfı Örneği 75
Şekil 5.11 : Bir Agventin Yaşam Döngüsü 76
Şekil 5.12 : PublicationAgvent İçin Bir Oluşturulan Duyuru Mesajı Örneği 77
Şekil 5.13 : Duyuru Sınıfının UML Diyagramı 78
Şekil 5.14 : Özellik Filtresi ile Yapılan Bir Kayıt Örneği 79
Şekil 5.15 : Davranış Filtresi ile Yapılan Bir Kayıt Örneği 79
Şekil 5.16 : Kayıt Sınıfının UML Diyagramı 80
Şekil 6.1 : Agvent Sunucusu Üzerinde Veri Tabanları 82
Şekil 6.2 : Komşu Sınıfının UML Diyagramı 82
Şekil 6.3 : Duyuru Sınıfının UML Diyagramı 83
Şekil 6.4 : Kayıtçı Sınıfının UML Diyagramı 84

Şekil 6.5 : Kayıt Sınıfının UML Diyagramı 85
Şekil 6.6 : JDO Uyumlu Bir Uygulamanın Bir Veri Deposuna Erişimi Diyagramı . 92
Şekil 6.7 : JDO Veri Erişim Modeli 93
Şekil 7.1 : Bir Olay Sistemi Topolojisi 96
Şekil 7.2 : Olay Verisinin Yayını 97
Şekil 7.3 : Kayıt Mesajlarının Yayını 98
Şekil 7.4 : Olay Verisinin Yönlendirilmesi 98
Şekil 7.5 : Duyuru Mesajlarının Yayını 99
Şekil 7.6 : Kayıt Mesajının Yönlendirilmesi 99
Şekil 7.7 : Olay Verisinin Yönlendirilmesi 100
Şekil 7.8 : Bağlı Bulunulan AS Üzerindeki Aşama 103
Şekil 7.9 : Agventin Kayıtçının Bağlı Bulunduğu AS Üzerindeki Çalışması 105
Şekil 7.10 : Kayıt Mesajlarının ABDES Sisteminde Yönlendirilmesi 106
Şekil 7.11 : Kayıt Mesajlarının Yönlendirilmesinde bir AS'nun Çökmesi Durumu 107
Şekil 7.12 : Kayıt Mesajının Alternatif Rota Üzerinden Yönlendirilmesi 107
Şekil 7.13 : Kayıt Mesajlarının Yayın ile Gönderilmesi 108
Şekil 7.14 : Agventlerin Yönlendirilmesinde bir AS'nun Çökmesi Durumu 108
Şekil 7.15 : Agventin Alternatif Rota Üzerinden Yönlendirilmesi 109
Şekil 7.16 : Duyuru Mesajının Yeniden Yayınlanması 109
Şekil 7.17 : Oluşan Alternatif Rota Üzerinden Agventin Yönlendirilmesi 110
Şekil 8.1 : PublicationAgvent Sınıfının UML Diyagramı 115
Şekil 8.2 : Duyuru Oluşturma Ana Ekranları 119
Şekil 8.3 : Duyuru Oluşturma Yardımcı Ekranları 120
Şekil 8.4 : Duyuru Listelerini Alma Ekranı 120
Şekil 8.5 : Kayıt Ekranları 121
Şekil 8.6 : Kullanılabilecek Kayıt Mesajları ve Haberleşme Örnekleri 122
Şekil 8.7 : Agvent Haberleşme Ekranı 124
Şekil 9.1 : Duyuru Tablosunun Boyutu 127
Şekil 9.2 : ObjectDB'ye Kayıt Ekleme Süreleri 128
Şekil 9.3 : ObjectDB'de Sorgulama Süreleri 129
Şekil 9.4 : ObjectDB'den Kayıt Silme Süreleri 129
Şekil 9.5 : Test Topolojilerindeki Atlama Miktarları 131
Şekil 9.6 : 200 ve 300 Agvent Sunuculu Rasgele Üretilmiş ABDES Sistemi 131
Şekil 9.7 : 1000 Agvent Sunuculu Rasgele Üretilmiş ABDES Sistemi 132
Şekil 9.8 : Duyuru Mesajlarının Dağıtım Sürelerinin Karşılaştırması 133
Şekil 9.9 : Kayıt Mesajlarının Dağıtım Sürelerinin Karşılaştırması 134
Şekil 9.10 : İki Konak Arası Agvent Gönderim Süresi 136
Şekil 9.11 : Bir bildiri verisi içeren agventin dağıtım süresi 137
Şekil 9.12 : Bir kitap verisi içeren agventin dağıtım süresi 137

ETMEN TABANLI DAĞITILMIŞ OLAY SİSTEMİ

ÖZET

Internet genişliğindeki her yere yayılmış geniş alan ağlarında ölçeklenebilirlik oldukça önemli ve sistemin her katmanında sağlanması gereken bir özelliktir. Bu özellikteki sistemler, haberleşme altyapısında potansiyel olarak çok sayıda dinamik katılımcı içermesi gerekebilir. Bu gibi sistemlerin merkezi yapıda değil, dağıtılmış yapıda gerçeklenmesi gerekmektedir. Olay tabanlı haberleşme bu şekildeki büyük ölçekli sistemlerin geliştirilmesinde yeni bir model olarak karşımıza çıkmaktadır. Sahip olduğu zayıf bağlılık özelliği sayesinde sistem bileşenleri rahatlıkla ölçeklenebilir hale getirilebilmekte ve sistem geliştiricilere basit bir programlama modeli sunulabilmektedir.

Kayıt/yayım haberleşme modeli özellikle olayların ana haberleşme mekanizması olarak kullanıldığı *Dağıtılmış Olay Sistemleri* için bir fonksiyonel model olarak görülmektedir. Bir olay sistem üzerinde ilgi duyulan bir eylemin olması durumunda oluşan bildirim (veri yapısı) olarak görülmektedir. Sistem bileşenleri ya olay kaynağı olarak tanımlanmakta ve yeni olay verilerini yayımlamaktadır ya da olay alıcısı olarak tanımlanmakta ve olay verilerine onun tanımlamaları (alanları) üzerinden kayıt olmaktadır. Kayıt/yayım haberleşme katmanı ise olay verilerinin dağıtımından sorumludur ve sistemin etkinliği için tüketicilerin tanımlamalarına göre olayları konusuna veya içeriğine göre filtrelemektedir-süzülmektedir.

Kayıt/yayım paradigması üreticilerin (yayımcıların) ve tüketicilerin (kayıtçıların) tamamıyla bağıntısızlaştırılması üzerine kurulmuştur. Olay Servisi, Kayıtçılar ve Yayımcılar arasındaki mesaj transferini üç boyutta bağıntısızlaştırarak sağlamaktadır; zaman bağımsızlığı, uzay bağımsızlığı ve akış bağımsızlığı. Kayıt/yayım sistemleri kayıt mekanizmalarına göre dört ana sınıfta gruplandırılabilir. Tarihsel gelişim açısından bu sınıfların her biri aşağıda incelenmiştir.

Kanal-Tabanlı Kayıt: Gerçeklenmesi en kolay olarak bilinen kayıt mekanizması kanal mekanizmasıdır. Kayıtçı bir kanala kayıt olur ve o kanalı dinlemeye başlar. Uygulamalar olaylar gerçekleştikçe ürettikleri bildirimleri doğrudan bir veya daha kanal üzerine yazarlar. Olay servisinde, olay verisinin görülen kısmı ise gönderildiği kanalın tanımlayıcısıdır. Olay servisine gönderilen bu mesaj o hattı dinleyen herkese ulaştırılır.

Konu-Tabanlı Kayıt: Bazı sistemler çıkan ihtiyaç üzerine kanal tabanlı yapıyı daha esnek bir adresleme ile değiştirerek konu tabanlı kayıt sistemlerini geliştirmişlerdir. Bu durumda olay verileri adresi tanımlayan *konu* ve kalan bilgileri içeren *detay* olarak iki ana parçaya ayrılırlar. Kanal yapısından temel farkı Kayıtçıların özel bir tanımlama ile birden fazla konu üzerine kayıt olmalarına olanak sağlanmasıdır. Olay servisi gelen mesajın konusunu inceleyerek ilgili Kayıtçılara göndermektedir.

İçerik-Tabanlı Kayıt: Filtreleme-süzme işlemlerinin faydası konu tabanlı sistemlerde ortaya çıkınca bu filtrelenebilen alanın sadece konu kısmı değil mesajın tamamı üzerine yapılması amaçlanmış ve içerik tabanlı kayıt sistemleri geliştirilmiştir. İçerik tabanlı sistemler yapı olarak konu tabanlı sistemlere çok benzerdir. İçerik tabanlı sistemlerin olay verisinin tamamına hakim olması sebebi ile gerekli kısıtlamalar, filtrelemeler olay servisi üzerinden halledilebilmektedir.

Tip-Tabanlı Kayıt: Tip tabanlı **(**Nesne tabanlı) kayıt mekanizması olay verisine daha yapısal bir yaklaşımda ulaşabilmek amaçlı olarak geliştirilmiştir. Olaylar çoğu sistem tarafından genelde alt seviyede önceden tanımlanmış mesaj tipleri olarak görülmekte ve bu da sisteme çok az bir esneklik kazandırmaktadır. Bu yapıyı esas alan Tip tabanlı Kayıt/yayım sistemi olayları birer nesne olarak ifade etmekte ve bunlar birer *obvent (object-event)* olarak adlandırılmaktadır.

Yukarıdaki tarihsel sürece bakılınca bir sonraki adımda etmenlerin olay sistemlerinde kullanılması olduğu görülmektedir ve bizde tez çalışmamızda bunu gerçekleştirmeye çalıştık.

Daha önce geliştirilen dağıtılmış olay sistemlerinde olay verileri yapısal olmayan katar listeleri, kayıt benzeri yapılar, ad-tanımlamalı özellikler, LISP deyimleri veya XML dokümanları gibi yinelemeli yapılar veya serileştirilebilen olay nesneleri olarak tanımlanmıştır. Yukarıda kayıt modelleri tanımlanan bu dağıtılmış olay sistemleri olay verilerini basit seviyede mesaj olarak tanımlayan sistemlerdir. Sadece tip tabanlı kayıt modelini kullanan dağıtılmış olay sistemleri olay mesajlarını nesne olarak tanımlamakta fakat bir özerklik vermemektedir. ***Bizim çalışmamız dağıtılmış olay sistemleri kullanılan olay verilerini zeki ve gezgin etmenler olarak tanımlayan bir yaklaşım getirmektedir.***

Bu tez çalışmasında gezgin etmenleri Kayıtçılar ve Yayımcılar arasında bir aracı olarak kullanan *Etmen Tabanlı Dağıtılmış Olay Sistemi altyapısı- ABDES Sistemi* sunulmaktadır. ABDES Sistemi kayıt/yayım protokolünü gerçeklemekte olup zayıf bağlı sistem varlıkları arasında çoklu etkileşime olanak sağlamaktadır.

ABDES Sistemi üç ana bileşenden oluşmaktadır; Agvent Sunucu, Yayımcı ve Kayıtçı.

- *Agvent Sunucular* gelen agventler için bir çalışma platformu hazırlamaktan ve gerekli yönlendirme işlemleri için duyuru ve kayıt mesajlarını saklamaktan sorumludur.
- *Yayımcılar,* Agvent Sunucuların bilgi tabanlarını inceleyerek hedef Kayıtçılarını seçen ve kendisini kopyalayarak bu makinelere yönlendiren agventleri üretmekte ve yayımlamaktadır.
- *Kayıtçılar* ilgi duydukları agvent tiplerini belirleyerek sistem tarafından değerlendirilecek filtreleri (ölçütleri) oluştururlar.

Sistemin bileşenleri arasındaki haberleşme Java RMI üzerinden sağlanmaktadır. Yapısal tarafsızlık özelliğinden dolayı Java ve RMI fiziksel olarak dağıtılmış olan heterojen makinelerde kullanılabilmekte ve bileşenlere saydam bir arayüz sağlayabilmektedirler. Sistemdeki mesaj ve etmen akışı aşağıdaki gibi işlemektedir.

1. Öncelikle bir Yayımcının sisteme üreteceği agvent tiplerini ilan etmesi *(Duyuru_Yap çağrısı)* gerekmektedir. Bu ilan daha sonra bütün agvent sunuculara yayın ile iletilir.
2. Kayıtçı bir Agvent Sunucuya bağlandığı zaman ondan, sistemde mevcut bulunan duyuru listelerini alabilir *(Duyuru_Listesi_Al çağrısı).* Tüm Agvent Sunucular aynı duyuru listelerine sahiptir.
3. Eğer Kayıtçı bir agvent tipi üzerine kayıt olmak isterse bağlı bulunduğu Agvent Sunucuya bir kayıt mesajı göndererek *(Kayıt_Ol çağrısı)* kendisini kayıt etmesini ister. Bu mesaj gerekli Agvent Sunuculara doğru sistem tarafından yönlendirilir ve mesaj Agvent Sunucuların bilgi tabanında saklanır.
4. Bir Yayımcı bir olayı gözlemlediği zaman ilgili agventi üretir ve bağlı bulunduğu Agvent Sunucuya gönderir *(Yayım çağrısı).* Agvent ilk Agvent Sunucuya ulaştığı zaman aktif hale gelerek Agvent Sunucunun bilgi tabanından aldığı bilgilere göre hedef makineleri (Agvent Sunucuları ve Kayıtçıları) belirler.
5. Bu bilgiler ışığında agvent kendi kopyalarını üreterek bunları hedef makinelere doğru yönlendirir *(Bildir çağrısı).*
6. Kayıtçı bir kayıt silme *(Kayıt_Sil çağrısı)* işlemi yapana kadar yayımlanan bu agventleri alır. Agvent bir Kayıtçıya ulaştığı zaman onunla haberleşmeye başlar. Kayıtçı etmenine istediği bir bilgiyi verir; özel bir bilgisi varsa bunu pazarlayabilir veya agventin üretilme amacına uygun olarak Kayıtçının bilgi tabanında bazı düzeltmeler/güncellemeler yapabilir.
7. Yayımcı bir tipteki agventi artık yayımlayacağı zaman bunu bir duyuru silme mesajı ile sisteme bildirir *(Duyuru_Sil çağrısı).* Bu durumda başka bir Yayımcı da aynı agvent tipini üretmiyorsa sistemdeki ilgili Kayıtçılara bu olay bildirilerek *(Bildir_Duyuru_Sil çağrısı)* yeni durumdan haberdar olmaları sağlanır.

ABDES Sistemi diğer dağıtılmış olay sistemlerinden aşağıda bahsedilen farklı özellikleri ile ayırt edilmektedir.

- *Özerk Olaylar:* Daha önceden gerçekleştirilen dağıtılmış olay sistemlerinde olay lar kayıt-benzeri yapılar, katar listeleri ve tuple tabanlı yapılar gibi basit mesajla r olarak tanımlanmışlardır. Tip tabanlı sistemde olaylar birer nesne olarak tanıml anmış olup biraz daha şahsiyet kazandırılmış olsa da bunlar gene de özerk varlık lar değildirler.

  ABDES Sisteminde ise olaylar basit mesajlar olarak tanımlanmamış olup, tam tersine kendi amaçları, inançları ve davranışları olan birer gezgin ve zeki etmenler olarak tanımlanmışlardır. Bir agvent, agvent sunucusuna ulaştığı zaman sunucu üzerindeki kayıt tablosunu kontrol ederek hedef makineleri seçer ve belirli işlemlerin sonucunda kendisini bu hedeflere gönderir. Bu yaklaşım sayesinde agvent sunucularının karmaşıklığının azaltılması amaçlanmaktadır.
- *Agvent Tabanlı Kayıt Mekanizması:* Geliştirilmiş bulunan dağıtılmış olay sistem lerinde Kayıtçılar ya bir kanal üzerine, yada bir konu üzerine, yada bir içerik üze rine kayıt olmaktadırlar. ABDES Sisteminde ise Kayıtçılar agvent tipleri üzerine kayıt olmaktadırlar. Örneğin bir Kayıtçı bir agvent üzerine kayıt olurken "Agv_t ype1" sınıfının örneği olan bir agvent üzerine kendi, ilgili agventin daha önce an ons edilen özellikleri ve davranışlarına uygun ölçütleri belirleyerek kayıt olabil mektedir.
- *Bilgi Saklama:* Daha önce geliştirilen olay sistemlerinde bir olay sunucusu kendi sine gelen mesajın içeriğine bakarak hedef makinelere gönderilmesini sağlamakt

adır. ABDES Sisteminde ise yayımlanan agvent Agvent Sunucu üzerinde hangi hedeflere gideceğini araştırmakta, Kayıtçıları seçmekte kendisini kopyalayarak b unlara gönderilmelerini sağlamakta ve bu sayede olay verilerinin gizliliğinin esa s olduğu uygulamaların gereksinimleri karşılanabilmektedir.

- *Kullanıcı/Uygulama Tanımlamalı Agventler:* Dağıtılmış olay sistemleri genellikl e önceden tanımlanmış tip ve modelde olay verileri kullanmaktadırlar. Bu neden le yeni bir olay tipi eklemek için, sistem yöneticisinin gerek dağıtım servisinde g erekse sistemin katılımcıları olan Kayıtçı ve Yayımcı bileşenlerinde programsal değişiklikler yapmak durumundadır.
  ABDES Sisteminde ise bir Yayımcı kendi yayımlayacağı agvent tipini belirlemekte ve bunun filtrelenebilecek olan özelliklerini ve davranışlarını sisteme anons etmektedir. Bir kez bu anons işlemi yapıldıktan sonra Kayıtçılar bu agvent üzerine kayıt olmaya başlamaktadır.
- *Kendi Kendini Yönlendiren Agventler:* Daha önceleri geliştirilen olay sistemlerin de olay sunucuların ana görevi olay verilerinin yönlendirilmesi idi. Agventin gez gin etmen yapısından dolayı her agvent ilgili olay tüketicilerine ulaşması için izl emesi gereken yolu, Agvent Sunucunun imkanlarından en alt seviyede faydalana cak şekilde belirler. Agvent Sunucu sadece o düğüme ulaşan agventlerin çalışma sı için uygun bir platform sağlamaktan ve kendine gelen duyuru/kayıt bilgilerini saklayarak sunucu üzerinde çalışan agventlerin bu bilgi tabanından etkin şekilde faydalandırmaktan sorumlu olur.
- *Çift Taraflı Filtreleme:* Olay sistemleri olay mesajlarının olay tüketicilere etkin dağıtımını sağlamak için filtreleme mekanizması kullanmaktadırlar. ABDES Sist emi hem tüketici filtrelemesini hem de üretici filtrelemesini desteklemektedir. B u sayede Yayımcı, ürettiği agventin belirli ölçütleri sağlamayan Kayıtçılara (olay tüketicilere) gönderilmesini engelleyebilmektedir. Burada yönlendirme işlemi ag ventin kendisi tarafından yapıldığı için bu filtreleme işlemi de agvent tarafından kolaylıkla sağlanmaktadır.

Bu temel özelliklerin yanı sıra geliştirilen ABDES Sistemi

- Ölçeklenebilir yapısı sayesinde Agvent Sunucuların sayısı artırılmakta ve daha fazla tüketiciye ulaşılabilmektedir.
- Kayıt ölçütlerini kolay ve anlamlı olarak belirlemeye olanak sağlayan kayıt modeli sayesinde tüketicilerin kendi ölçütlerini kolaylıkla tanımlamasına olanak sağlamaktadır.
- Kullanılan Duyuru ve Kayıt mesajları sayesinde sistem üzerinde dolaşan mesajların sayısı en aza indirmektedir.
- Hata Kotarma mekanizması sayesinde Duyuru ve Kayıt Mesajlarının dağıtımı veya Agventlerin yönlendirmeleri sırasında bir Agvent Sunucunun çökmesi halinde alternatif yollar üzerinden hedef makineye ulaşım olanağı sağlamaktadır.
- Bilgi Tabanı Yönetiminde Nesne Veri Tabanları kullanılması ile Duyuru ve Kayıt mesajlarının Agvent Sunucular üzerinde birer nesne olarak saklanmasına ve bu nesnelere hızlı şekilde erişilmesine olanak sağlamaktadır.

Sistem performansını ölçmek için ABDES Sistemi üzerinde bazı testler yapılmıştır. Bu test sonuçları değerlendirildiğinde ABDES Sisteminin genişleyebilir yapısı ve anlamlı bir kayıt modeli olduğu ve sistemin özellikle fazla sık üretilmeyen, fakat büyük boyutlu, değerli ve gizli verilerin kullanıldığı olay dağıtım sistemleri içi uygun olduğu değerlendirilmektedir.

AGENT BASED DISTRIBUTED EVENT SYSTEM

SUMMARY

In Internet-wide and ubiquitous systems, scalability is vital and must be ensured at all layers. In these systems, a messaging middleware may potentially have millions of dynamic clients and, therefore, must be implemented in a distributed fashion. Event-based communication has become a new paradigm for building large-scale distributed systems of this type. It has the advantages of loosely coupling communication partners, being extremely scalable, and providing a simple application programming model.

The publish/subscribe communication paradigm has been recognized as a functional model particularly for *Distributed Event Systems in which*, events are the basic communication mechanism. An event can be seen as a notification (data structure) that something of interest has occurred within the system. Components either act as event sources and publish new events, or event sinks and subscribe to events by providing a specification of events that are of interest to them. A publish/subscribe communication layer is then responsible for disseminating events; for efficiency, it can often also filter events by topic or content, according to client specifications.

Publish/subscribe programming paradigm is characterized by the complete decoupling of producers (Publishers) and consumers (Subscribers). The event service that provides message transfers between Publishers and Subscribers can be decomposed along three dimensions, time decoupling, space decoupling, flow decoupling. Publish/subscribe systems can be classified into four groups according to their subscription mechanism. According to historical development these mechanisms are discussed in detail below.

Channel-based subscription: The simplest subscription mechanism is what is commonly referred to as a *channel*. Subscribers subscribe and listen to a channel. Applications explicitly notify the occurrence of events by posting notification to one or more channels. The part of an event that is visible to the event service is the identifier of the channel to which the event has been sent. Every notification posted to a channel is delivered by the event service to all Subscribers listening that channel.

Subject-based subscription: Some systems extend the concept of a channel with a more flexible addressing mechanism that is often referred to as *subject-based* addressing. In this case, an event notification consists of two different parts: a well-known attribute, *the subject* that determines the address, which is followed by *the remaining information* of the event data. The main difference with respect to a channel is that subscriptions can express interest in many subjects/channels by specifying some form of expression to be evaluated against the subject of a notification.

Content-based subscription: By extending the domain of filters to the whole content of notifications, some researchers obtain another class of subscriptions called content-based. Content-based subscriptions are conceptually very similar to subject-based ones. However, since they can access the whole structured content of notifications, an event server gets more freedom in encoding the data upon which filters can be applied and that the event service can use for setting up routing information.

Type-based subscription: Type based (Object based) publish/subscribe model is a new model of subscription that has been developed to access event data in a more structured manner. Events are often viewed as low-level messages, and a predefined set of such message types are offered by most systems, providing very little flexibility. To overcome this deficiency, type-based publish/subscribe mechanism manipulates events as objects, called *obvents (object-event).*

According to this historical development, described above, the next step should be usage of agents in event systems and in our thesis we decide to realize this.

In early Distributed Event Systems, event data is represented by unstructured lists of strings, record-like structures with positional or name-based identification of attributes, recursive structures, such as LISP expressions or XML documents, and serializable event objects. Previously developed Distributed Event Systems whose subscription models defined above, are representative systems which define events as low level messages. Only event systems with type based subscription define events as objects but it also doesn't provide them with autonomy. ***Our work proposes a new approach for distributed event systems through a model in which events are represented by mobile intelligent agents.***

In this thesis, we present an agent based *Distributed Event System Framework, the ABDES System* framework, which exploits mobile agents as mediators between Publishers and Subscribers of events. The ABDES System implements the publish/subscribe protocol, thus enabling many-to-many interaction of loosely coupled entities

ABDES System consist of three main components; Agvent Servers, Publishers and Subscribers. The roles of these components are as follows;

- *Agvent Servers* are responsible for providing an execution platform for incoming agvents and storing advertisement/subscription messages for necessary routing operations.
- *Publishers* publish agvents of pre-specified type that search the knowledge bases of Agvent Servers to select lists of Subscribers, clone and route themselves to target Subscriber.
- *Subscribers* determine agvent types which they are interested in and describe a filter to be processed by the system.

Communication between system components is carried out using Java RMI. Due to architecture neutrality, Java and RMI handle geographically distributed heterogeneous machines and provide a transparent view to the participants. A sample message/agvent flow of the system is listed as follows

1. Firstly, a Publisher advertises its agvent type to the system. This advertisement is dispatched to all Agvent Servers in the dispatch service via a broadcast message.
2. If a Subscriber is connected to an Agvent Server, it can obtain the advertisement list *(Get_Advertisement_List)* which includes a list of agvent types currently available on the system. All Agvent Servers hold the same advertisement list.
3. If a Subscriber decides to subscribe on an agvent type, it sends a subscription message to the Agvent Server *(Subscribe)* it is connected to in order to have itself registered. Next, this message is routed to necessary Agvent Servers by the system. The details of the subscription are stored in subscription tables of knowledge bases in each Agvent Server.
4. Whenever a Publisher observes an event, it creates and sends an agvent to the Agvent Server *(Publish)*. When the agvent arrives on the Agvent Server, it starts to execute its pre-specified code to select its targets (neighbor Agvent Servers and/or registered Subscribers) according to the information present in the subscription table.
5. Next, the agvent creates its clones and sends each one to a different target on the list, and after completing this task, it disposes itself *(Notify)*.
6. A Subscriber continues to receive published agvents until it issues an unsubscribe request for that particular agvent type *(Unsubscribe)*. When an agvent arrives on a Subscriber, it starts to communicate with the Subscriber. It delivers the Subscriber agent a message, which could either contain a complex event data, such as a secret password, a negotiation for an e-commerce or etc., or a request for an action to be carried out, such as updating its database according to incoming data carried by the agvent, running an update routine for its software, or etc., according to the execution logic of the current application.
7. When a Publisher stops publishing a certain type of agvent, it informs the system through an unadvertise message *(Unadvertisemet)*. In this case, if no other Publishers of that agvent are present, the system sends out a message to Subscribers registered on that agvent type, informing them of the new situation *(Notify_Unadvertisement)*.

The ABDES System differs from other distributed event systems with its distinct characteristics that are described below.

- *Autonomous Events:* In most event systems, events are defined as low-level messages, which consist of record-like structures, list of strings, tuple-based structures, etc. In type-based systems, events are defined as objects and viewed as main component of the system. Nevertheless, they are not autonomous.
  In the ABDES System, events are not viewed as simple messages. On the contrary, they are represented as mobile agents that have their own goals, beliefs and behaviors that they acquire at creation. When an agvent reaches to an Agvent Server, it examines the routing table of the server and selects its targets autonomously after certain processing. This approach reduces the load and complexity of Agvent Servers as well.
- *Agvent Based Subscription:* In most distributed event systems, Subscribers register on a channel, on a specific topic or a specific content of an event message. In the ABDES System, Subscribers register on agvent types. For example, a Subscriber can register on an agvent, which is an instance of “Agv_type1” class, specifying certain constraints based on its advertised attributes and behaviors.

- *Information Hiding:* In previously developed event systems, an event server can, actually has to, access the content of the published event data before it can dispatch the data to the registered targets. In the ABDES System, the published agvent itself searches the knowledge base of the Agvent Server, selects the registered Subscribers, clones itself and sends each agent clone to a Subscriber on the selected list. Therefore, the Agvent Server has no access to the content of the published event data, which simplifies its role and consequently facilitates the server development process. Information hiding also meets requirements of certain applications where confidentiality of event data is essential.
- *User/Application Defined Agvent Types*: Distributed event systems generally use predefined event types. Therefore, to add a new event type you have to make programmatic changes in the dispatch service and also at Publisher and Subscriber sites.
  In the ABDES System, a Publisher creates its own agvent type and declares its properties and behaviors through an advertisement message sent to the Dispatch Service Once an agvent type is announced on the Dispatch Service, Subscribers can register on agvents of that type.
- *Self Routing Agvents:* In previously developed event systems, routing of events is the main duty of the Agvent Servers. Because of the mobile agent structure of agvents, route of the agvent is determined by itself with minimum help of Agvent Servers. Role of an Agvent Server is providing an execution platform for agvents and storing advertisement/subscription messages for necessary routing operations.
- *Double Side Filtering:* Event systems uses filter mechanism for dispatching event messages to event consumers. ABDES System enables not only Subscriber side filtering, but also Publisher side filtering. Therefore Publisher can define some criterion for selecting Subscribers. Because routing operations are determined by agvents, Publisher side filtering is enabled.

Apart from these distinctive characteristics ABDES System has the following properties.

- By scalable structure of the system, it is possible to increase the number of Agvent Servers and reach more consumers.
- By subscription model, it is possible to define expressive subscription criterions by consumers.
- By using advertisement and subscription messages, the number of total messages on the system is decreased to minimum level.
- By fault tolerance mechanism of the system, in case of collapsing an Agvent Server, it is possible to dispatch advertisement messages, subscription messages and agvents from alternative route.
- By using object databases, advertisement and subscription messages are stored as objects on Agvent Servers. By this way it is possible to reach these data quickly.

In order to measure systems performance several tests have been made on ABDES System. On comparing these test results we conclude that ABDES System has an expressive subscription model and is a scalable system. It is especially suitable for events which are not generated frequently but contain large, valuable and secret data.

1. GİRİŞ

Internet'in dünya üzerinde hızla yaygınlaşması, dağıtılmış sistemlerin kullanım ölçeğini önemli ölçüde genişletmiştir. Günümüzde dağıtılmış sistemler sadece bir yerel alan ağındaki veya bir metropolitan ağındaki bilgisayarlardan öte, dünya çapında yayılmış binlerce bilgisayardan oluşabilmektedir. Geliştirilen bir uygulama sistemin amacı gereği bu varlıkların konumları, davranışları ve işlevleri sistemin çalışma süresince her zaman değişikliğe uğrayabilmektedir. Geleneksel istemci-sunucu mimarisinin kullanıldığı küçük ölçekli dağıtılmış sistemler uçtan uca ve senkron haberleşmelerinden dolayı statik uygulamalardan öteye geçememiştir. Bu tür haberleşmenin büyük ölçekli sistemlerde kullanılması ise uygulamaların hantal ve kullanışsız olmalarına neden olmuştur. Bu kısıtlamalar, dinamik olan ve bağlantısız uygulamaların çalıştığı bir ortamda, daha esnek haberleşme modellerini kullanan yazılım sistemlerinin geliştirilmesi gerekliliğini ortaya çıkarmıştır.

Yazılım sistemlerinin geniş bir alanı tepkisellik üzerine geliştirilen sistemlerden oluşmaktadır. Tepkisel sistemler bir işlevin yapılması, üst seviyede kontrol akışı veya doğrudan programlama yerine, belirli olayların olması durumunda tetiklenmektedir. *Olay Bildirim Sistemi* olarak adlandırılan bu gibi sistemler fonksiyonelliklerini ise oluşan olaylara karşı belirli eylemleri yerine getirerek sağlamaktadırlar. Özellikle bilgi tabanlı sistemlerin geliştirilmesinde kullanılan bu gibi tepkisel sistemlere örnek olarak borsa/hisse takip sistemleri, akademik yayın bildirim sistemleri, akış kontrol sistemleri, süreç analiz sistemleri, grafik kullanıcı ara yüzleri, ağ yönetim sistemleri, güvenlik takip sistemleri ve bütünleşik geliştirme ortamları verilebilir.

Günümüzün bilgi tabanlı sistemlerinde, basit manadaki bir bilgi uzayında iki farklı kategori bulunmaktadır. Birinci kategorideki sistem varlıkları, başka sistem varlıklarından, kişilerden veya araçlardan elde edilen veriler gibi *faydalı bir veriye* sahiptir. İkinci kategorideki sistem varlıkları ise bu veriye sahip olmamakla birlikte, kendi görevlerini yerine getirmeleri için bu veriye ihtiyaç duymakta ve bu veriye sahip olmak/bilgilendirilmek istemektedirler. Çeşitli tipte ve sayıda verinin dolaştığı bilgi tabanlı sistemlerde üretilen her veri, sistemdeki varlıkların büyük bir kısmı için faydalı bir bilgi olmayabilir. Bir veri, çalışma amacına göre belirli bir gruptaki sistem varlıklarının işlerine yararken, diğer varlıklar için önem arz etmeyen bir bilgi de

olabilir. Bu verinin ihtiyaç sahibi sistem varlığına ulaştırılması için iki temel yol kullanılmaktadır. Bunlar:

- *Periyodik gönderim*. İhtiyaç sahibi sistem varlığı belirli aralıklarla, veri sahibi sistem varlığından sahip olduğu son verileri ister.
- *Olayın gerçeklenmesinde gönderim*. Veriye ihtiyaç duyan sistem varlığı, veriye sahip olan sistem varlığına isteğini ilettikten sonra, istek duyulan mesajın gelmesi durumunda (olayın gerçeklenmesi durumunda) istek sahibine ilgili veriyi gönderir.

Olay Bildirim Sistemi verinin üretilmesini/ulaşmasını bir olay olarak kabul eden, bu ikinci stratejinin somut hale getiren sistem yapısıdır. Burada kastedilen olayın genel bir anlamı vardır. Örneğin, olay:

- *bir ofis ortamında,* ofisin sıcaklığının değişmesi, bir kapının kapanması, bir telefonun çalması, bir kişinin bir odaya girmesi veya çıkması, oda nem değerinin belirli bir limitin üzerine çıkması,
- *bir bilgisayar sisteminde,* bir kişinin bir oturum açması, bir yazılım çalıştırması, bir belgeyi açması,
- *bir borsa uygulamasında,* hisse senetlerinin fiyatlarının değişmesi, işlem hacminin belirli bir miktarın üzerine çıkması,
- *bir oyun yazılımında,* bir silahın ateşlenmesi, bir canavarın ölmesi,
- *bir akademik yayın bildirim sisteminde,* bir sempozyum sonucu üretilen bildiri kitapçığının basılması veya yeni bir kitabın piyasaya çıkması,
- *bir web sunucusunda,* bağlantı sayısının belli bir sayının üzerine çıkması, diskteki yerin belirli bir sınırın altına inmesi gibi eylemler olarak görülebilir.

Burada bahsedilen örnekler sınırlı sayıda olmakla beraber, gerek sanal alemde, gerekse gerçek dünyada ne kadar farklı ve fazla olayın cereyan ettiğini göstermektedir. Olay verisi meydana gelen bir olayın, bilgisayarda tanımlanmış veri formatıdır. Örneğin olay verisi;

- sıcaklık değişmesi ile elde edilen yeni sıcaklığı tutan
  “sicaklik=30°C” ,
- bilgisayar sisteminde açılan bir oturumu açan kişinin bilgilerini tutan
  “user_id=’Sahingoz’; makine_adi=’Sahin’; Giris_Zamani=’2005/01/02,11:15:34’”,
- hisse senedinin fiyat değişikliğinde elde edilen fiyat değişim oranını tutan
  “hisse = ‘SAHOL’; degisim_oran=2.3%;”,

şeklinde olabilir.

Olay Bildirim Sistemlerinde amaç, üretilen verinin veya cereyan eden olayla ilgili verilerin istek sahibine en hızlı ve doğru şekilde ulaştırılmasının sağlanmasıdır. Bu verinin ulaştırılması için kullanılan haberleşme modeli *Olay Tabanlı Haberleşme Modeli* olarak adlandırılmaktadır. Olay tabanlı haberleşme modeli, dağıtılmış ve heterojen ortamlardaki asenkron olarak bağlanmış bileşenler arasındaki haberleşmeyi sağlayan bir paradigma olarak ortaya çıkan, ve günümüzde büyük ölçekli Internet servisleri [1], gezgin programlama ortamları [2, 3], eşzamanlı her yerde bulunan/çalışan (ubiquitous) programlar [4, 5] gibi uygulama alanlarında yaygın olarak kullanılmaktadır.

Olay tabanlı haberleşme modeli özellikle, bir veya daha fazla uygulama bileşeninin, başka bir uygulama bileşenindeki bir durumun değişmesine tepkide bulunmasının gerektiği uygulamalarda kullanılması faydalıdır. Olay tabanlı haberleşme yapısı gereği asenkron haberleşmeyi desteklemekte olup klasik istemci/sunucu modelinin ötesinde zayıf bağlı sistemlerde de gerçeklenebilmektedir. İsimsizliği[1] destekleyen yapısı sayesinde, çok sayıda isimsiz etkileşen bileşenin, merkezi bir kontrol olmadan çalışmasına olanak sağlayan uygulamalarda kullanılmaktadır. Ayrıca günümüzde kablosuz teknolojinin hızla genişlemesi ile olay tabanlı haberleşme modelinin kablosuz bir sistemdeki bileşenler arasındaki

- uzun süreli ve pahalı bağlantıların sağlanmasının zorluğu,
- bağlantısız etkileşimlerden dolayı haberleşme gecikmesinin yüksek olması,
- merkezi kontrolün eksikliği,
- heterojenlik,

gibi sistem gereksinimlerine cevap vermesi sebebiyle kablosuz ve gezgin sistemlerdeki haberleşme ihtiyaçlarına da cevap verebilmektedir.

Olay tabanlı özel yazılımlar, finans, haberleşme, zeki ortamlar, çoğul ortam, havacılık elektroniği, sağlık ve eğlence gibi farklı uygulama alanlarında kullanılmaktadır. Olay bildirim sistemleri ufak ölçekli merkezi sistemlerden, büyük ölçekli dağıtılmış sistemlere kadar her alanda kullanılmaktadır. Bir taraftan kişisel uygulama geliştirim platformlarında grafiksel kullanıcı ara yüzünün tasarımı gibi [6, 7] kullanıcı tarafından oluşturulan ve grafiksel bileşenin belirli bir özelliğinin değişmesi durumunda tetiklenecek bir eylem dizisini içermekteyken, diğer taraftan bir borsa uygulamasında sistemin hızlı çalışmasına yardımcı olması açısından hisse senetlerinin işlem tahtasındaki hareketlerinin sisteme kayıtlı aracı kurumlara/kullanıcılara eş zamanlı olarak gönderilmesini amaçlayan dağıtılmış, büyük ölçekli bir sistemi [8] içerebilmektedir. Aynı zamanda “zeki evlerin” ve “zeki

[1] Anonymity. Sistem üzerinden haberleşen varlıkların birbirlerin adlarını ve adreslerini bilmemesi.

binaların" geliştirilmesinin amaçlandığı sistemlerde [9] ise olay tabanlı özel yazılımlar, ışık ve kapı algılayıcıları gibi basit algılayıcılardan, binalar arasında hareket edebilen robot araçlar gibi çok sayıda uygulama bileşenini içerebilmektedir.

Etmen (agent) kavramı, nesneye yönelik yaklaşımın daha genişletilmiş bir hali olarak özellikle dağıtılmış sistemlerde kullanılmaktadır. Etmenler, bilgisayar bilimlerinin (dağıtılmış programlama, yapay zeka, gezgin programlama… vs) birkaç dalını birleştirdiği için çekici bir çalışma alanı oluşturmaktadır.

Etmenler, kısaca kişi veya kuruluşlar adına özerk olarak hareket edebilen program parçaları olarak düşünülebilirler. Ait olduğu kişinin işlerini yapabilirler ve bilgisayarlar arasında hareket edebilirler. Her etmen, görevini kendi iç karar mekanizmalarına göre yerine getirmek için kendi çalışma sürecine ve kendi hedefine sahiptir.

Son yıllarda etmen kavramı üzerine çok sayıda çalışma yapılmış ve farklı alanlarda etmen tabanlı çok sayıda uygulama geliştirilmiştir. Özellikle zeki ve gezgin etmenler üzerine yapılan çalışmaların artması, bu konuda belirli standartların ve uygulama platformlarının gelişmesi, etmen tabanlı programlamanın kolaylaşmasına ve çok etmenli (multiagent) sistemlerin yayılmasına sebep olmuştur. Bir konaktan başka bir konağa, etmenin hareket edebilmesine olanak sağlayan *gezgin etmen* yapısı, standart istemci/sunucu programlama modelinden farklı olarak, dağıtılmış sistemlerde sağlamış olduğu hesaplama yapısı sayesinde alternatif bir model olarak karşımıza çıkmaktadır.

Bu tez çalışmasında günümüzde yaygın olarak kullanılan *Dağıtılmış Olay Bildirim Sistemlerinin* ve *Etmen Sistemlerinin* özelliklerini birleştirerek bir *Etmen Tabanlı Dağıtılmış Olay Sisteminin*[2] tasarlanması amaçlanmıştır. Oluşturulacak bu sistemin özellikleri bir sonraki bölümde detaylandırılmıştır.

1.1. Tezin Amacı

Günümüze kadar geliştirilen tüm olay sistemleri olay kavramını, veri kaynağı / veri üreticisi tarafından üretilen basit formatta bir veri olarak algılamışlar ve çalışmalarını üretilen bu verinin, uygun formatta gösterilimi, işlenmesi ve hızlı şekilde yayılması üzerine yoğunlaştırmışlardır. Bu tez çalışmanın amacı ise:

> *Olay kavramının üretim niteliğini geliştirerek, alışılagelen sadece ham veri üretiminin yanı sıra, eylemlerin (action) de üretildiği ileri bir düzeye yerleştirerek bu nitelikteki olayların üretilmesine, dağıtılmasına ve işlenmesine olanak sağlayan bir etmen tabanlı dağıtılmış olay sistemi altyapısı geliştirmektir.*

[2] Tez çalışması boyunca Olay Sistemi, Olay Bildirim Sistemi ile eş anlamlı olarak kullanılmaktadır.

Bu düzeydeki bir olay, işlenebilir veriler üretebileceği gibi, kendine özgü bir kontrol akışına sahip olan bir eylemler dizisinin tetikleyicisi de olabilir. Söz konusu eylemler belirli bir amacı gerçeklemeye yönelik işlem dizisine, karar mekanizmalarına ve çoğunlukla birlikte üretildikleri bir veri kümesine sahip olacaklardır.

Bu tez çalışmasında, bir olayın tetikleyeceği eylem dizisinin ve bu eylemleri yerine getirmede kullanacağı veri bloğunun yer aldığı bir *gezgin etmen* yapısıyla temsil edilmesi öngörülmüştür. Bu nedenle, tezde kullanılan bir olayın yarattığı eylem dizisini temsil eden yapıya "agvent" (**ag**ent e**vent**) adı verilmiş olup sistem bir bütün olarak *"Etmen Tabanlı Dağıtılmış Olay Sistemi"* veya İngilizce karşılığı olan "Agent Based Distributed Event System" in kısaltması olarak *"ABDES Sistemi"* olarak adlandırılmaktadır.

ABDES Sistemi, hem gezgin ve zeki etmen yapısının, hem de kayıt/yayım modeliyle haberleşen olay tabanlı sistemlerin özelliklerini birleştiren bir Etmen Tabanlı Dağıtılmış Olay Sistemidir. ABDES Sisteminin içerdiği bu iki bileşenin temel özellikleri aşağıda detaylandırılmıştır.

1.1.1. Etmenler

Son yıllarda etmen kavramı üzerine çok sayıda çalışma yapılmış ve etmen tabanlı çok sayıda uygulama geliştirilmiştir. Etmen kavramı üzerinde evrensel kabul gören tanım bulunmamaktadır. Fakat bu konuda çalışma yapan araştırmacıların ve bilim adamlarının temelde üzerinde anlaştıkları konu, etmenlerin *özerk-özerk* varlıklar olmasıdır. Bunun haricindeki özelliklerde fikir birliğine sahip olunmamakta ve bu özellikler etmenin çalıştığı platformun gereksinimlerine uygun olarak kullanılmakta veya kullanılmamaktadır.

Genel bir etmen tanımını kabul etmek gerekirse bu konuda kapsamlı çalışmaları olan Woolridge ve Jennings bir etmeni şu şekilde tanımlamaktadır [10]:

> ***Etmen;*** *bir fiziksel çevre içinde yerleşmiş, kendi tasarım amaçlarına uygun olarak özerk eylemler yapabilme yeteneği olan bilgisayar sistemleridir / yazılımlarıdır.*

Etmenlerde esas olan özerklik olduğu için özerkliği de biraz açıklamak gerekir. Woolridge etmendeki özerkliği şu şekilde tanımlamıştır.

> ***Özerklik****; bir etmenin herhangi bir insanın veya başka bir sistemin müdahalesi olmadan, kendi durum değişkenlerinde, davranışlarında ve eylemlerinde kontrole sahip olmasıdır.*

Zeki etmenin ne olduğu aslında zekanın ne olduğu ile yakından ilgilidir. Buna cevap vermek karmaşık olmasına rağmen Woolridge ve Jennings zeki etmen kavramını şu şekilde açıklamışlardır [10].

> ***Zeki Etmen;*** *tasarım amaçlarına uygun olarak esnek ve özerk eylemleri olan etmenlerdir.*

Esnek eylemleri olan bir zeki etmende özerkliğin yanı sıra şu üç özelliğinin olması gerekmektedir.

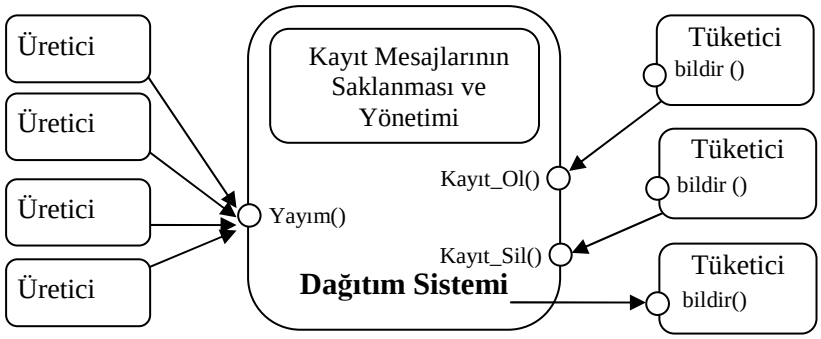
- ***Aktif Olma (amaca yönelik davranma) :*** Zeki etmenler, tasarım amacına uygun olarak, çalışma inisiyatifini ellerinde tutarak, amaca yönelik davranışlarda bulunurlar. Amaca yönelik davranış gösterecek bir sistem geliştirmek fazla karmaşık değildir. Basit anlamda PASCAL dilinde bir prosedür, C dilinde bir fonksiyon veya Java'da bir metot yazmak bu işlevi yerine getirebilir. Böyle bir yordam geliştirilirken, bu yordam bazı ön koşullara bağlanarak bunların gerçeklenmesi durumunda çalışması, bazı son koşullara bağlanarak, çalışma sonucunda bu hedeflere ulaşılması amaçlanır. Bunların yerine getirilmesi sırasında yapılan değişiklikler yordamın amacı olarak görülebilir.
- ***Sosyal Yetenek:*** Zeki etmenler, diğer etmenlerle (veya insanlarla) tasarım amaçlarını yerine getirmek için iletişimde bulunabilirler. Günümüzde milyonlarca bilgisayar, diğer bilgisayarlarla veya insanlarla veri alışverişinde bulunmaktadır. Fakat bu varlıklar arasında sekizli dizileri ile yapılan veri alışverişi aslında sosyal bir yetenek değildir. Gerçek hayat ile karşılaştırdığında, bir amaca ulaşmak için değişik kişilerle veya kuruluşlarla çalışmak gerekebilir Bunun için bu kişilere bir işi yaptırmak için, onların da amacını karşılayacak şekilde müzakere yapılması, bir anlaşmaya ulaşılması gerekebilir. Benzer şekilde etmenlerin de kendi amaçlarına ulaşabilmeleri için başka etmenlerle uygun bir *etmen haberleşme dili*[3] ile bağlantı kurarak bilgi ve görüş alışverişinde bulunmaları sosyal bir yetenek olarak görülür.
- ***Tepkisellik:*** Zeki etmenler mevcut çalışma ortamını alıcıları ile algılarlar tasarım amacına uygun olarak bu ortamı değiştirmek için meydana gelen olaylara tepkide bulunur.

1.1.2. Kayıt/Yayım Haberleşme Modelinin Özellikleri

Kayıt/Yayım sistemleri, çok sayıda veri üreticisi ile veri tüketicisinin bilgi alışverişinde kullandıkları bir haberleşme modelidir. Bu sistemlerde üretilen veriye

[3] Etmen haberleşme dili, farklı sistemler tarafından geliştirilmiş etmenler arasındaki haberleşme için kullanılır.

ihtiyaç duyan tüketicilerin, ihtiyaç duydukları verinin ölçütlerini belirleyerek sisteme bir *kayıt* mesajı ile kaydolmaktadır. Veri üreticileri de sisteme *yayım (publish)* yolu ile ürettikleri veriyi göndermektedirler. Gönderilen verinin ihtiyaç sahibi tüketicilere ulaştırılması ise dağıtım sisteminin görevidir. Örnek bir Kayıt/Yayım sisteminin haberleşme modeli Şekil 1.1'de gösterilmektedir.



Şekil 1.1 : Kayıt/Yayım Haberleşme Modeli

Kayıt/Yayım Haberleşme Modelinin temel özelliklerini şu şekilde gösterilebilir:

- ***Bir uçtan çok uca haberleşme (point-to-multipoint):*** Dağıtılmış olay sisteminin geliştirilmesi aşamasında en fazla faydalanılan özellik olan çoğul haberleşme yapısı sayesinde sisteme dahil olan tüketiciler ve üreticiler birbirlerinden doğrudan haberdar olmadan, kimliklerini bilmeden haberleşmektedirler. Üreticilerin ürettikleri ve sisteme gönderdikleri bir veri, sisteme daha önce kendi profili ile kayıt olmuş olan ilgili çok sayıdaki olay tüketicisine ulaştırılır. Üreticiler ve tüketiciler arasındaki veri akışı olay sistemi üzerinden sağlanır.
- ***Asenkron haberleşme:*** Üreticiler ve tüketiciler dağıtım sistemi üzerinden asenkron olarak haberleşirler. Bu sayede üretici, olay verisini sisteme gönderdikten sonra bunun tüketicilere ulaştırılması dağıtım sisteminin görevi olacaktır.
- ***Gevşek bağlı (loosely coupled) sistemlerde kullanıma uygunluk:*** Üreticiler ve tüketiciler sistemde aynı anda aktif durumda olmalarına gerek yoktur. Eğer üretici, tüketiciden sonra sisteme dahil olmuşsa tüketici, sisteme yeni katılan bu üreticinin de yayımlayacağı verilere ulaşabilecektir.

1.1.3. ABDES Sisteminin Özellikleri

ABDES Sisteminin, daha önceleri geliştirilen dağıtılmış olay sistemlerinden üstün olan temel özelliklerini kısaca şu ana başlıklar altında toplanabilir.

Özerklik: ABDES Sisteminde olaylar basit bir olay verisi olarak görülmez ve sistemin temel elemanı, birinci sınıf üyesi olarak işlem yapılır. Üretilme aşamasında

agventler birer gezgin etmen olarak üretilir ve bu aşamada, içlerine kendi amaçları, inançları (çevre bilgisi) ve davranışları yüklenir. Bu gezgin etmen, bir agvent sunucusuna ulaştığı zaman, kontrolü agvent sunucusuna devretmemekte, agvent sunucunun bilgi tabanını inceleyerek kendi hedef makinelerini özerk olarak seçmekte ve kendisini bu makinelere göç ettirmekte/kopyalamaktadır. Bu yaklaşım sayesinde olay sunucusu geliştirmenin zorluğunu ve karmaşıklığını da büyük ölçüde azaltılır ve yönlendirme işlemi agvent tarafından yapılması sağlanmış olur.

Tip/Agvent Tabanlı Kayıt: ABDES Sisteminde, sisteme dahil olan kayıtçılar[4] ilgi duydukları olaylara, ilgili agventin tipini bildirerek kayıt olurlar. Olay bildirimlerinde aranan ölçütler kayıtçının sisteme göndereceği kayıt (*Kayit_Ol*) çağrısında ayrıntılı olarak yer alacaktır. Bu ölçütler sayesinde sadece agventin özellikleri üzerinden değil, aynı zamanda davranışları üzerinden de süzme yapılmasına olanak sağlanmaktadır. Bu sayede gerek aramada gerek filtrelemede daha esnek bir çalışma ortamı sağlanacaktır.

Bilgi Saklama: ABDES Sisteminde oluşan olaylar birer gezgin etmen yaratır ve bu etmenler agvent sunucular arasında kendini yönlendirir. Bu nedenle bir etmenin taşıdığı değerlere dış kullanıcılar tarafından erişilmesi mümkün olmaz, agvent sunucusu ve kayıtçı olay verisine ulaşamaz. Olay verisine ilgili etmenlerin kendi aralarında yürüttükleri kontrol işlemlerinden sonra erişilmesine olanak sağlanır (mesajlaşma ile). Bu sayede veri güvenliği sadece olay sunucusunun görevi olmaktan çıkar ve gönderilen agventin de güvenlik konusunda etkin bir şekilde rol alması sağlanır.

Kullanıcı Tanımlamalı Agvent Tipleri: ABDES Sisteminde olay üreticileri, sistem tarafından önceden tanımlanmış olan agvent tiplerini kullanmak zorunda olmayıp, kendi olay modellerine uygun yeni agventler tanımlayıp üretebilirler. Önceden tanımlanmış agvent tipleri sistemde mevcut ve kullanıyor olsa bile yeni agvent tiplerinin sisteme duyurulması (*Duyuru_Yap* çağrısı ile) ve kullanılması mümkündür.

Kendi Kendini Yönlendiren Agventler: Agventin gezgin etmen yapısından dolayı her agvent ilgili olayın tüketicileri olan kayıtçılara ulaşması için izlemesi gereken yolu, agvent sunucusunun imkanlarından en alt seviyede faydalanacak şekilde belirler. Yönlendirme işlemlerinde fazla etkin olmayan agvent sunucu sadece o düğüme ulaşan agventlerin çalışması için uygun bir platform sağlamaktan ve kendine gelen kayıt bilgilerini saklayarak sunucu üzerinde çalışan agventlerin bu bilgi tabanından etkin şekilde faydalandırmaktan sorumlu olur.

[4] Tezin işlenişinde olay tüketici ve kayıtçı eş anlamlı olarak kullanılmaktadır.

Çift Taraflı Filtreleme ABDES Sistemi olay tüketici/kayıtçı tarafından yapılan filtrelemeleri desteklemekle beraber, olay üretici/yayımcının[5] ürettiği agventinde kendi filtrelemesini yapabilmesine de olanak sağlamaktadır. Bu sayede agvent üretici, ürettiği agventin belirli ölçütleri sağlamayan kayıtçılara (olay tüketicilere) gönderilmesini engelleyebilmektedir. Burada yönlendirme işlemi agventin kendisi tarafından yapıldığı için bu filtreleme işlemi de agvent tarafından kolaylıkla sağlanmaktadır.

1.1.4. Tezin Uygulama Alanları

Internet gibi geniş ölçekli ağlarda çalışması planlanan ABDES Sisteminin özelliklerinden etkin şekilde yararlanılabilecek örnek uygulamalar olarak şunlar gösterilebilir.

1.1.4.1. Dağıtılmış Ortamda Gizli Bilgi (Şifre) Dağıtımı

Haberleşme güvenliğinin sağlanması dağıtılmış ortamlarda önemli bir servis kalitesidir. Gönderilen verilerin istenmeyen kişilerin eline geçmemesi için verinin şifrelenerek gönderilmesi sık tercih edilen bir modeldir. Bu modelde uzun süre aynı şifrenin (secret key) kullanılması şifre kırıcıların bu şifreyi çözme imkanlarını artıracaktır. 64 bitlik şifrelemenin yapıldığı RC5-64 projesinde [11] 15,769,938,165,961,326,592 farklı şifre test edilerek şifre kırılması sağlanmıştır. Bu nedenle belirli aralıklarla kullanılan şifrelerin değiştirilmesi ve diğer kullanıcılara gönderilmesi gerekir. Bu gibi kritik öneme sahip veri gönderilmesinde güvenlik kontrolünün sadece olay sistemindeki olay sunucusuna değil, aynı zamanda olay/şifre üretici tarafından tasarlanan agvente yüklemek güvenliği artıracaktır. Şimdiye kadar geliştirilen olay sistemlerinde olay sunucuları olay verilerini görebilir ve ihtiyaç durumunda yorumlayabilirlerdi. Geliştirilecek agvent sayesinde, üretilen ve dağıtılan agventin bilgi saklama (information hiding ve encapsulation) özellikleri sayesinde gerek olay sunucularının gerekse olay tüketicilerin/kayıtçıların agventin içeriğini görmesi engellenebilmektedir.

1.1.4.2. Yazılım Güncelleme İşlemleri

Yazılım şirketleri tarafından geliştirilen ürünlerin kullanımında zamanla yazılım hatalarıyla karşılaşılır. Yazılım Mühendisliği sürecinin en uzun süreli basamağı olan "Bakım ve Güncelleme" basamağında karşılaşılan bu hataların yazılımcı şirket tarafından düzeltilmesi gerekir. Geniş bir kullanıcı kitlesine sahip olan yazılımlarda çoğu şirkette böyle bir hata ile karşılaşılmamış ve düzeltme talebi gelmemiş olabilir. Bu gibi durumlarda, yazılım şirketi tarafından geliştirilecek bir agvent sisteme

[5] Tezin olay üretici ve yayımcı eş

gönderilerek, ilgili yazılımı kullanan şirketlerin bilgisayarlarına bu agventin ulaşması sağlanacaktır. Müşterilere ulaşan agvent kullanıcıların haberi olmadan gerekli düzeltmeleri yapacaktır.

1.1.4.3. Virüs Tanımlarının Güncellenmesi

Anti virüs programı geliştiricileri, geliştirdikleri yeni virüs tanımlamalarını içeren bir agventi dağıtılmış olay sistemine göndererek, kayıtlı olan kullanıcıların makinesinde gerekli güncellemelerin yapılması sağlayabilirler. Günümüzde ise bu virüs tanımlamaları, anti virüs firmaları tarafından geliştirildikçe, ilgili anti virüs firmasının web sayfasına konulmaktadır. Bu sayfayı takip eden firmalar, belirli aralıklarla bu tanımlamaları kendi makinelerine indirerek gerekli güncellemeleri yapmaktadırlar.

1.1.4.4. Fiyat Ayarlaması

Bir şirket kendi ürettiği veya dağıtımını yaptığı ürünlerin, bayileri aracılığı ile piyasaya pazarladığı bir uygulama platformunu düşünelim. Şirket bu ürünlerin fiyatlarında yapacağı bir fiyat değişikliğini veya uygulayacağı bir kampanyayı, doğrudan bayilerin bilgisayarlarına aktarabilmek için geliştireceği bir agventi, bayilerinin kayıtlı olduğu olay sistemine göndererek gerekli fiyat düzenlemelerinin otomatik olarak yapılmasını sağlayabilir. Bunun için bayilerin ek bir çaba göstermesine gerek olmayacağı gibi, bayiler arasında fiyat farklılığı da ortadan kalkacaktır.

1.1.4.5. Dağıtılmış Geniş Çaplı Hesaplama Ortamı Sağlanması

Büyük ölçekli ve yüksek işlem gücüne sahip süper bilgisayarlar oluşturmak yerine yüksek işlem gücü elde etmek için dağıtılmış olan çok sayıda makinenin işlemcilerinden faydalanmak günümüzde tercih edilen bir yaklaşımdır. Çalışması uzun süren, ancak kullanacağı veri bloğu (veya veri tanımlama bloğu) fazla geniş olmayan uygulamalarda böyle bir yaklaşımın kullanımı uygun olacaktır. Örneğin asal sayı bulma uygulamasını ele aldığımızda veri paketlerini 10^8 lik bloklar halinde istemci makinelere gönderilerek, bu bloktaki sayılar içinde asal sayı var olup olmadığını test eden bir uygulamayı agventin içine yerleştirerek olay sistemine gönderilmesi ve bu test işlemlerinin istemci makinelerde yürütülmesi düşünülebilir. Asal sayılara ulaşılınca istemci makine sonucu ya olay sistemine ya da agventi tanımlayan makineye gönderir.

1.1.4.6. Yayın Dağıtım/Bildirim Sistemleri

Sayısal Kütüphane uygulamasında yayınlanan/üretilen kitapların, dergilerin hatta daha detayda makalelerin olay verisi olarak kullanıldığı geniş ölçekli bir kütüphanecilik hizmeti veya yayın dağıtım sistemi olarak düşünülebilir. Bu sistemde olay üreticisi olan yayınevleri veya özelde düşünülebilecek tekil yazarlar, oluşturdukları yayının sistem üzerinden pazarlanmasını amaçlamaktadırlar. Sisteme kayıt olan olay tüketicileri (kayıtçılar)belirli ölçütlere sahip olay verilerinin (yayınların) kendine ulaşmasını istemektedir. Kendine ulaşan bilgiler neticesinde seçimini yapmakta ve seçilen yayımcı ile bağlantı kurarak ulaşmak istediği yayına doğrudan ulaşmaktadır. Olay verisi kendine ulaşana kadar kayıtçılar ve yayımcılar birbirlerini bilmemektedir.

1.2. Tezin Organizasyonu

Tez üç ana konu üzerinde bölümlendirilmiştir. Buların ilki tezin işlenmesine temel teşkil eden bölümlerden oluşup şu konu başlıkları ile detaylandırılmıştır.

Bölüm 2: "Olay Sistemleri" anlatılmış olup, olay sistemlerinin (özellikle dağıtılmış olay sistemlerinin) nitelikleri vurgulanmıştır. Olay sistemlerinin kullandıkları kayıt mekanizmaları, sunucu topolojileri, olay verisi tanımlama modelleri, olay verisi alma modelleri ve bazı servis kalitesi özellikleri tanımlanmıştır.

Bölüm 3: "Etmenler" in genel tanımı yapılarak temel özellikleri vurgulanmıştır. Gezgin ve zeki etmenlerin tarihsel gelişimi gösterilerek sahip oldukları özellikler detaylandırılmıştır.

Bölüm 4: Günümüze kadar olay sistemleri ile ilgili yapılan başlıca çalışmalar ve bu çalışmaların özellikleri *"Konu ile İlgili Çalışmalar"* başlığı altında incelenmiştir.

Bu sistemlerin eksik kalan yönlerini gidermek için geliştirilen ABDES Sistemin detaylarını gösteren ikinci bölüm şu başlıklar altında toplanmıştır.

Bölüm 5: Geliştirilen *"ABDES Sisteminin"* sahip olduğu temel özellikler detaylandırılmış olup, sistemin altyapısı ve temel bileşenleri gösterilmiştir. Bileşenler arasında dolaşan temel veriler olan duyurular, kayıtlar ve agventlerin yapıları gösterilerek, ABDES Sisteminde kullanılan filtreleme mekanizmasının çalışması açıklanmıştır.

Bölüm 6: Agventin hızlı bir şekilde gönderilmesini sağlamak için agventin, sunucular üzerinde istediği bilgiye hızlı şekilde ulaşması gerekmektedir. Bu

hızlı veri erişimi sağlamak için ABDES Sisteminde kullanılan *"Bilgi Tabanı Yönetiminin"* nasıl yapıldığı ve bu erişim için kullanılan Java Veri Nesneleri kavramı izah edilmiştir

Bölüm 7: "ABDES Sisteminde Yönlendirme" konusunda günümüze kadar kullanılan temel yönlendirme stratejileri örneklenmiş olup, ABDES Sisteminin bu sistemlerden farklı yaklaşımı gösterilmiştir. Aynı zamanda agvent sunucularda oluşabilecek bir çökme durumunda hata kotarma işlemlerinin de nasıl yapıldığı gösterilmiştir.

Üçüncü ve son bölümde ise elde edilen etkinlik test sonuçları ve geliştirme önerileri yer almaktadır.

Bölüm 8 "Uygulama Örneği" konu başlığı altında ABDES Sisteminin uygulama örneği olarak geliştirilen bir Yayın Dağıtım Sisteminin tasarımı gösterilerek, böyle bir uygulama geliştirmek için yapılması gereken işlemler basamak basamak gösterilmiştir.

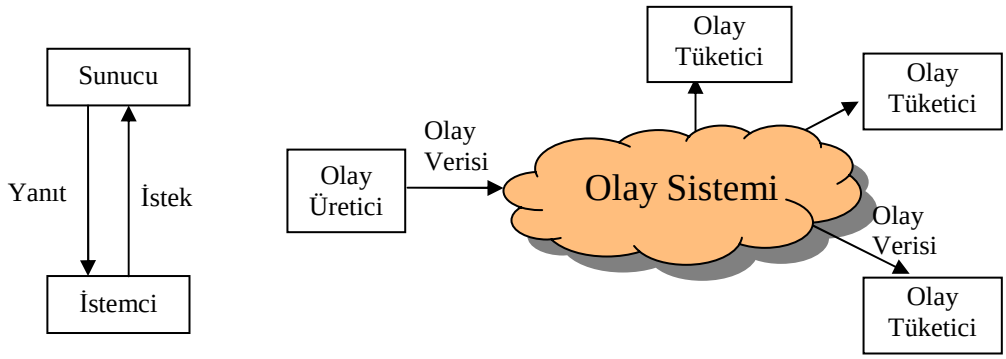
Bölüm 9: "Test Sonuçları" konu başlığı altında ABDES Sisteminin Performansının ölçülmesi amacıyla, Duyuru Mesajlarının, Kayıt Mesajlarının ve Agventlerin dağıtım süreleri ölçülerek değerlendirilmiştir.

Bölüm 10: Tez çalışmasının sonucunda *"Sonuç ve Öneriler"* gösterilmiş olup tezin genel bir değerlendirilmesi yapılarak, açık kalan konular belirtilmiş.

2. OLAY SİSTEMLERİ

Geleneksel istemci/sunucu hesaplama modellerinde [12] haberleşme için kullanılan arabirim *sıkı bağlı*, *senkron* ve *uçtan uca (point-to-point)* veri alışverişine olanak sağlayacak şekilde tasarlanmaktadır. Şekil 2.1.a'da gösterildiği gibi, bir istemci uzaktaki bir sunucu üzerindeki metodu çağırmak istediğinde (Remote Procedure Call-RPC) ilgili metodu uygun parametreler göndererek tetikler, ve bir sonucun dönmesini bekler. İstemci/sunucu modelini çalıştırabilmek için, istemcinin ve sunucunun birbirleri hakkında belirli bilgilere sahip olmalarını gerekmektedir (sunucu makinenin adresi, hangi metodun çağrılacağı, hangi parametrelerin gönderileceği, bu parametrelerin tipleri, vs).

Büyük boyutlu dağıtılmış sistemlerin kullanılmaya başlanmasıyla, *gevşek bağlı, asenkron,* ve *çok alıcılı (point-to-multipoint)* haberleşme ortamlarının kullanılması ihtiyacı ortaya çıkmıştır. Bu ihtiyacı karşılayabilmek için geliştirilen olay sistemlerinin amacı bir olayın (veya bir bileşik[6] olayın) olması durumunda gerekli olay verilerini bu olay ile ilgilendiğini beyan eden tüketicilere ulaştırılmasını sağlamaktır. Bunu yaparken oluşturulan olay verisinin içine hangi tüketicilere gideceği veya hangi üreticinin ürettiği ile ilgili adres verileri yazılmaz. Olay sistemleri, uygulamadan bağımsız olarak, olay tabanlı ve asenkron haberleşmeye imkan veren bir altyapı sağlamaktadır. Şekil 2.1.b'de bir olay sisteminin bileşenleri arasında bir olay verisinin nasıl transfer edildiği gösterilmektedir.



a) İstemci/Sunucu Haberleşmesi b) Olay Tabanlı Haberleşme

Şekil 2.1 : İstemci/Sunucu ve Olay Tabanlı Haberleşme Modelleri

[6] Bileşik olay birkaç olayın arka arkaya olması sonucu oluşan olay modelidir. Örneğin "bir odada sıcalık yükseliyor ve odanın aydınlığı artıyor ise alarmı çal veya itfaiyeye haber" ver gibi

Olay tabanlı haberleşmede, olay sistemine katılan üretici ve tüketiciler, mantıksal olarak merkezi olarak kabul edilen bir Mesaj Transfer Arabirimi (Olay Sistemi) üzerinden haberleşirler. Olay sistemi sadece bir olay sunucusundan oluşabileceği gibi, çok sayıda sunucunun (yönlendiricinin) farklı topolojilerde birbirleriyle bağlanmasından da oluşabilir. Olay sisteminin yapısı sisteme bağlanan uygulamalardan (üretici ve tüketicilerden) bağımsız ve saydamdır. Sistem içindeki sunucular zaman içinde, sistemdeki uygulamalardan bağımsız olarak yeniden yapılandırılabilir (yeni sunucular eklenebilir ve çıkarılabilir).

2.1. Dağıtılmış Olay Sistemi

İlk geliştirilen olay sistemlerinden olan Field [13], SUN ToolTalk [14], Yeast [15] incelendiğinde bu olay sistemlerinin tek makinede yada daha gelişmiş olanların ise bir yerel alan ağında çalıştıkları görülmektedir. Zamanla

- olay sistemlerinin kullanımının artması,
- sistemdeki olay üreticilerinin ve olay tüketicilerinin çoğalması,
- yerel alan ağlarında olan
    - yüksek bant genişliği,
    - düşük gecikme zamanı,
    - homojen platform,
    - güvenilir bağlantı
    - ve merkezi kontrol gibi sistem gereksinimlerinden uzaklaşılması,

olay sistemini dağıtılmış ortama geçirmeyi zorunlu kılmıştır.

Dağıtılmış olay sistemlerinde, sistemin altyapısını oluşturan dağıtılmış olay sunucuları belirli topolojiler çerçevesinde birleşerek, olay üreticilerden gelen olay verilerinin olay tüketicilerine hızlı ve güvenilir şekilde ulaştırılması amaçlanmaktadır. Bu amaçla geliştirilen sistemlerde;

- gerçek zamanlı borsa analiz uygulamaları,
- posta ve haber dağıtım sistemleri,
- veri madenciliği uygulamaları,
- indeksleme uygulamaları,
- geniş alana yayılmış şirketler için *iş akışı* uygulamaları geliştirilebilmiştir.

Dağıtılmış olay sistemlerini incelerken bu yapıya uygun olarak günümüzde yaygın kullanımı olan bazı Internet tabanlı teknolojilerden bahsetmek konunun daha iyi anlaşılmasını sağlayacaktır.

2.2. Dağıtılmış Olay Sistemlerinin Tarihsel Gelişimi

Olay Bildirim Sistemlerinin ilk örneklerinden biri olarak Seçici Bilgi Dağıtımı (Selective Information Dissemination-SID) üzerine yapılan çalışmalar gösterilebilir. SID çalışmaları ilk olarak bilim adamları ve araştırmacıların uzmanlaştıkları konu üzerinde çıkan yayınlar hakkında bilgilendirilmeleri amacıyla geliştirilmiştir. Bir kullanıcı kendi konusu ile ilgili anahtar kelimeleri belirleyerek sisteme kayıt olur ve ihtiyaç durumunda bu anahtar kelimeleri değiştirebilir. Bu sisteme örnek olarak geliştirilen SIFT [16] yapılan yayınlamanın sonucunda sisteme gönderilen olay verisinin anahtar kelimeleri sistem tarafından kontrol edilerek ilgili araştırmacılara gereken veri gönderilir.

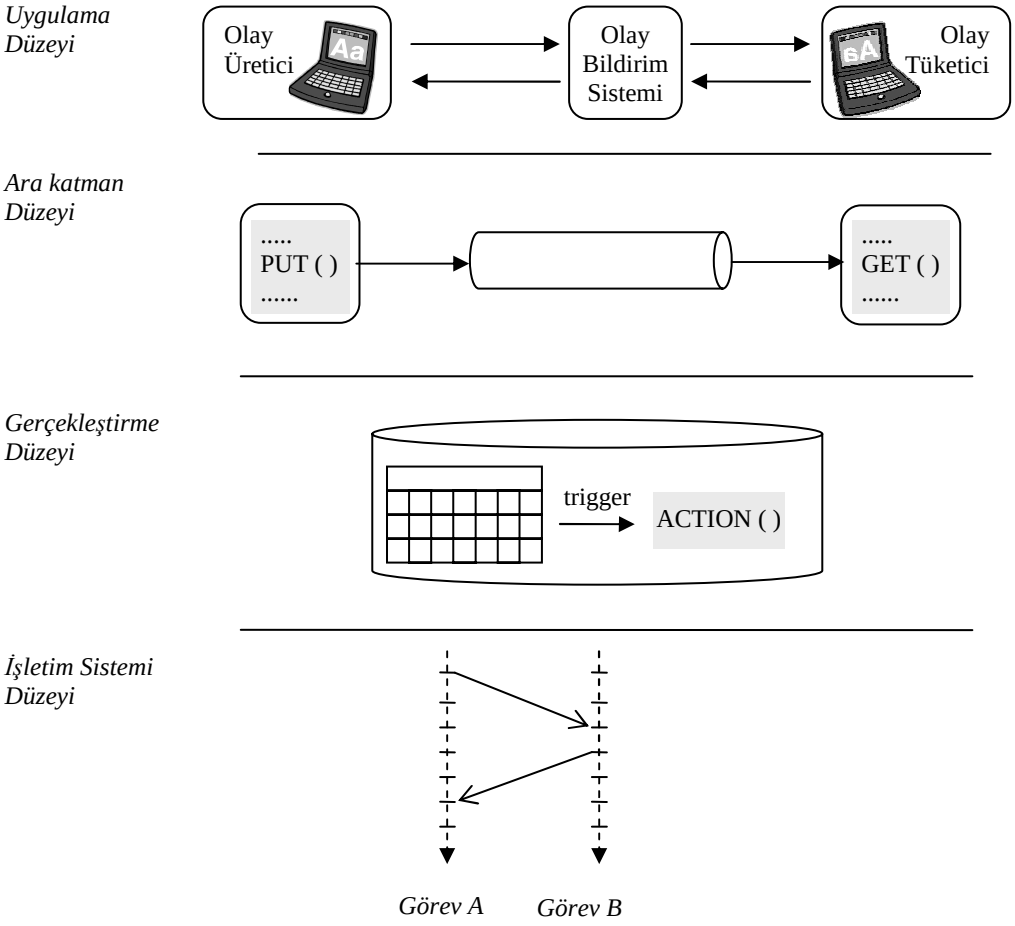
80'li ve 90'lı yıllarda araştırmacılar daha ziyade ufak boyutlu (ince taneli) verilerin dağıtımı ve etkin filtreleme metotları üzerinde çalışmışlardır. Bilgi Filtreleme Sistemleri, SID sistemlerinin yapısal olmayan veya yarı yapısal hali olarak geliştirilmiştir. Belkin ve Croft [17] bilgi filtreleme ve bilgi erişim işlemlerinin soyut kavramda aynı işlemler olduğunu göstermişlerdir. *Bilgi Erişimde* ihtiyaç duyulan bilgiye uygun olarak bir sorgulamanın hazırlanması ve bunun sistem üzerinde bir kez çalıştırılması hedeflenmişken, *Bilgi Filtrelemede* tanımlanan bir profil ile müşterilerin taleplerine uzun süreli cevap verilmesi amaçlanır.

Internet teknolojilerinin 90'lı yıllarda hızla gelişmesi ve yayılması ile itme-teknolojisi (Push-Technology) bilgi dağıtımı problemine bir çözüm olarak karşımıza çıkmıştır. Geliştirilen olay bildirim sistemlerinin günlük kullanıma yardımcı olması amaçlanmış ve sadece dokümanların yayınlanması aşamasında değil, aynı zamanda hisse senetlerinin anlık değişmesi, bir kitabın yayınlanması veya değişik yerlere yerleştirilmiş olan alıcıların bir hareketi algılaması gibi kullanıcının ihtiyaç/ilgi duyduğu olayların olması durumunda da gereken veri dağıtımını yapmayı amaçlamıştır. Bu servislerin kullanımı için olay üretici ve olay tüketici tarafında da bazı sistem yazılımlarının kurulması gerekir.

Günümüzdeki araştırma alanları ise geniş alandaki olay bildirim sistemleri üzerindeki dağıtılmış, etkin ve ölçeklenebilir sistemler üzerinde yoğunlaşmaktadır [18 - 20]. Mevcut sistemler, ya uygulamaya yönelik sistemler yada genel maksatlı sistemler olarak karşımıza çıkmaktadır.

2.3. Olay Tabanlı Haberleşme Düzeyleri.

Mesajlaşma ile haberleşme iyi bilinen ve dağıtılmış sistem geliştirilmesinde sık kullanılan bir modeldir. Olay Bildirim Sistemi (OBS) ise tek üreticiden çok sayıdaki tüketiciye mesaj göndermeye olanak sağlayan farklı düzeylerde soyutlanabilen bir haberleşme modelidir. Bu düzeyler Şekil 2.2'de gösterildiği gibi "Uygulama Düzeyinden" başlayarak "İşletim Sistemi Düzeyine" doğru azalan düzeylerde gerçeklenebilir.



Şekil 2.2 : Olay Tabanlı Haberleşme Düzeyleri

Uygulama Düzeyi sistemler, OBS'ler üzerinde 1968'ten beri yapılan çalışmalarda özellikle üzerinde uğraşılan sistemlerdir. Bu sistemlere örnek olarak sayısal kütüphaneler, hisse bildirim sistemleri, *acente bilgi sistemlerini*[7] verilebilir. Bu gibi sistemler genellikle üzerinde çalışan uygulamaya bağımlıdırlar ve belirli profiller ile sisteme kayıtlı olan müşterilere bu profillere uygun verileri ulaştırırlar. Bu düzeyde gerçekleşmiş bazı OBS'ler şunlardır; SIFT [21], SIENA [22], OpenCQ [23], GRYPHON [24], LESUBSCRIBE [25].

[7] Bir seyahat planlamasında ihtiyaç duyulan bilgileri seyahat acentaları üzerinden toplayarak gezi planlamasını kolaylaştıran sistemler.

Arakatman Düzeyindeki sistemlerde, sistem üzerindeki dağıtılmış yazılım bileşenleri aralarında özel olarak tanımlanmış olay mesajları/verileri ile haberleşirler. Bir üst düzeyden gelen (uygulama düzeyi) olaylar ham veri olarak tutulur ve işlenirler. Bu düzeyin örnekleri olarak; BEA Mesaj Sevisi [26], MQSeries [27], ve CORBA Bildirim Servisi [28] ve Java Mesaj Servisi [29] gösterilebilir.

Gerçekleştirme Düzeyindeki servisler bir veri tabanı tablosuna kayıt eklenmesi gibi genellikle iç olaylara göre işlemler yaparlar. Ara katman Düzeyindeki olaylar genellikle bu düzeydeki olaylarla eşleştirilir. Olay tabanlı haberleşme örneği olarak aktif veri tabanlarındaki kurallar ve tetiklemeler gösterilebilir.

İşletim Sistemi Düzeyinde olaylar, sistemdeki görevlerin senkronizasyonunda [30] ve hareketlerin görüntülenmesinde [31] kullanılır. Bu düzey aynı zamanda basit düzeyde olay yakalama işlemlerinde, grafiksel kullanıcı arabirimlerindeki fiillerin/hareketlerin yakalamasında programlama dilleri tarafından kullanılır (Java Swing [32] ve SWT [33]).

Tez çalışmasında Uygulama Düzeyinde bir sistem geliştirmesi hedeflenmiştir Aynı zamanda Ara katman ve Gerçekleme Düzeyinde hizmetlerde geliştirilmiş olup işletim sistemi düzeyindeki olay tabanlı haberleşmeye ilişkin destek Java'nın mevcut kütüphanelerinden sağlanmıştır.

Dağıtılmış Olay Sistemleri farklı yaklaşımlarla sınıflandırılabilirler. Bu yaklaşımları Mesajlaşma Modelleri, Kayıt Mekanizmaları, Sunucu Topolojileri, Olay Veri Modelleri, ve Veri Alma Modelleri olarak beş farklı kategoride incelenebilir.

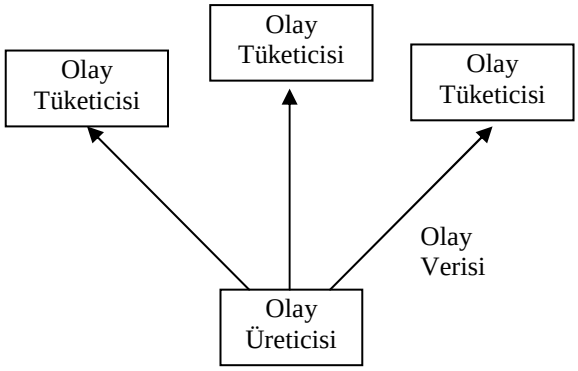
2.4. Mesajlaşma Modeli

Olay tabanlı sistemlerde, sistem içindeki bileşenler birbirlerinden bağımsız olarak çalışırlar. Bu bileşenlerin ortak bir görevi yerine getirebilmeleri için kendi aralarında konuşmaları, yani mesaj alışverişi yapmaları gerekmektedir. Günümüze kadar geliştirilen olay sistemleri incelenince, bu mesaj alışverişleri, *uçtan uca haberleşme* (her ne kadar olay sisteminin temel amacına aykırı olsa da) ve *kayıt/yayım haberleşme* modeli olarak iki ana sınıfta irdelenebilir.

2.4.1. Uçtan Uca Haberleşme

Uçtan uca (point-to-point) haberleşene olay sistemlerinde, Mesaj Transfer Arabirimi olmadan sistemdeki olay üreticilerin ve olay tüketicilerinin haberleşmelerine olanak sağlanmaktadır. Şekil 2.3'te de görülebileceği gibi, olay tüketicileri ilgilendikleri olay verisine ulaşabilmek için bu olayı üretecek olan olay üreticisiyle doğrudan bağlantı kurmak durumundadırlar. Göndereceklerı bir kayıt mesajı ile, ilgilendikleri

X olayını, olay üreticisine bildirir. Olay üretici de ilgili olay meydana gelince bu olaydan haberdar edilmek isteyen tüketicilere ya olay mesajının tamamını gönderir yada olay mesajının referansını göndererek tüketicilerin bu olay verisine ulaşmasına olanak sağlar.



Şekil 2.3 : Olay Tabanlı Uçtan Uca Haberleşme Modeli

Bu modelde, istemci/sunucu modelinde de olduğu gibi, tüketiciler, üreticilerin adreslerini ve nasıl kayıt olacaklarını bilmek durumundadırlar. Aynı olay ile ilgili birden fazla üretici var ise, tüketici her üretici için ayrı ayrı kayıt olması gerekmektedir. Olay üretici üzerinde çok sayıda olay tüketicinin kayıt olduğu uygulama platformunda olay üreticisi üzerinde yoğunluk oluşacaktır. Bu yoğunluktan dolayı mesaj transferinde bir yavaşlama oluşacağı gibi, üreticinin tamponlama kapasitesine göre veri kaybı oluşabilecektir. Aynı zamanda olay tüketicilere mesaj gönderme ve kontrol işlemlerinin yoğun olmasından dolayı olay üretici kendi asıl işi olan *olay üretmeye* yeterli zaman ayıramayacaktır.

2.4.2. Kayıt/Yayım Haberleşmesi

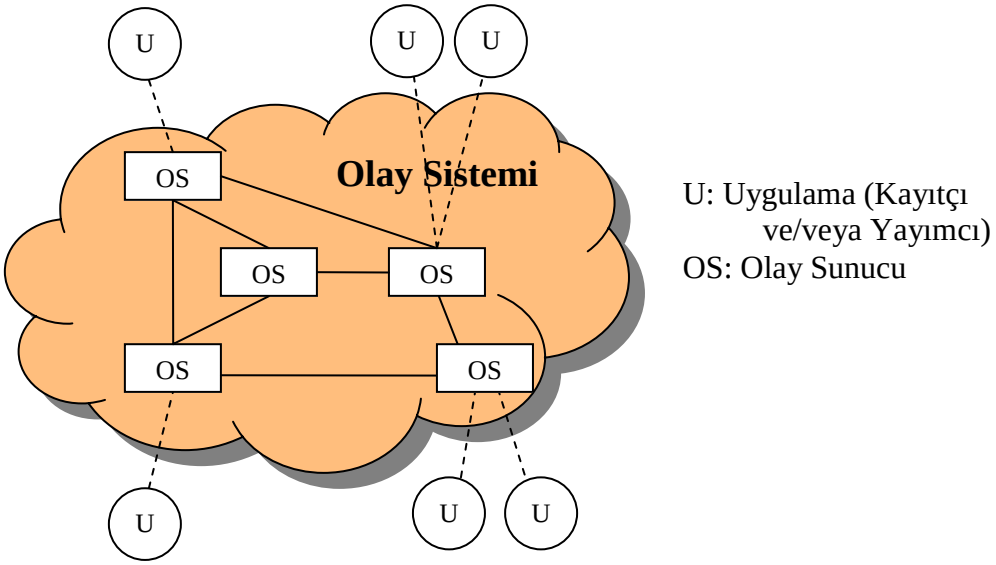
Olay sistemlerinin kullanıldığı uygulama modellerinin çeşitlenmesi ve gelişmesi, sistemde aktif olan olay üreticilerinin ve olay tüketicilerinin sayısının artmasına yol açmıştır. Uçtan uca haberleşme modeli ile, çok sayıda sistem bileşeninin olduğu durumlarda aşılması zor mesaj yoğunlukları ile karşılaşılmıştır. Buna alternatif olarak geliştirilen kayıt/yayım haberleşme modeli, olay sistemindeki olay üretici ve olay tüketicilerinin birbirleri ile habersiz olacakları, gönderilen bir olay verisinin çok sayıda tüketiciye ulaşmasına olanak sağlayan bir uçtan çok uca asenkron olarak haberleşebilen bir altyapı sağlamaktadır.

Kayıt/yayım haberleşmeli sistemlerde olay tüketiciler *kayıtçı*, olay üreticiler ise *yayımcı* olarak adlandırılmaktadır[8]. Şekil 2.4'te kayıt/yayım haberleşme modelini

[8] Tez çalışmasının değişik yerlerinde geliştirilen ABDES Sisteminden bahsederken **Kayıtçı** ile **Olay Üretici** ve **Yayımcı** ile de **Olay Tüketici** eş anlamlı olarak kullanılmaktadır.

kullanan basit bir olay siteminin yapısı gösterilmektedir. Bu modeli kullanan olay sistemlerinde üç temel bileşen vardır. Bunlar:

- ***Kayıtçı*:** Olay sistemine, ilgilendiği olayları belirterek kendini kayıt ettirir ve bu olayların gerçekleşmesi durumunda asenkron olarak gelen olay verilerini alarak işler.
- ***Yayımcı*:** Belirli olay mesajlarını/verilerini üreterek olay sistemine dağıtılmak üzere gönderir.
- ***Mesaj Transfer Arabirimi-Olay Sistemi*:** Yayımcıdan gelen olay mesajları alarak, bu mesajla ilgilendiğini bildiren kayıtçılara gönderir. Olay sistemi, çok sayıda olay sunucusunun bir topoloji çerçevesinde birleştirilmesinden oluşur. Sisteme gönderilen olay verisinin kayıtçılara ulaşması, bu olay sunucularının koordineli bir şekilde çalışması ile sağlanır.



Şekil 2.4 : Kayıt/Yayım Haberleşme Modeli

2.5. Kayıt Mekanizmaları

Kayıt mekanizması, bir kayıt/yayım sistemine katılacak olan olay tüketicilerin sisteme nasıl kayıt olacaklarını, kayıt olurken hangi verileri, hangi yapıya uygun olarak göndereceklerini ve bunların sistemde nasıl saklanılacağını ifade eder. Kayıt/yayım haberleşme modellerini kullanan olay sistemlerinde üç temel ilkel üzerinden haberleşme yapılır. Bunlar:

- ***Kayıt_Ol (subscribe):*** Belirli bir olay ile ilgilenen olay tüketici, ilgilendiği olayla ilgili ölçütleri belirterek sisteme kayıt olur.
- ***Kayıt_Sil (unsubscribe):*** İlgilendiği bir olaya artık ilgi duymayan tüketici olay sisteminden kaydını sildirmek amacıyla kullanır.

- ***Yayım (publish):*** Olay üretici, bir olayı ürettiği zaman bunu olay sistemine bildirmek amacıyla kullanır.

Bazı sistemlerde sistemin esnekliği ve daha etkin kullanılabilmesi için birkaç ilkel (advertise, unadvertise, vs) daha eklenmişlerdir. Olay sisteminin etkin ve verimli çalışabilmesi, gereksiz olay mesajlarının sistemde fazla dolaşmaması, sistemin hızlı ve verimli çalışabilmesi için en önemli ilkel *Kayıt_Ol (subscribe)* ilkelidir. Kayıt ilkelin tanımlanması ve kullanımı günümüze kadar geliştirilen çok sayıdaki olay sisteminde farklılıklar göstermektedir. Bu ilkelin kullanım modeline göre dört sınıfta incelenebilir. Bunlar:

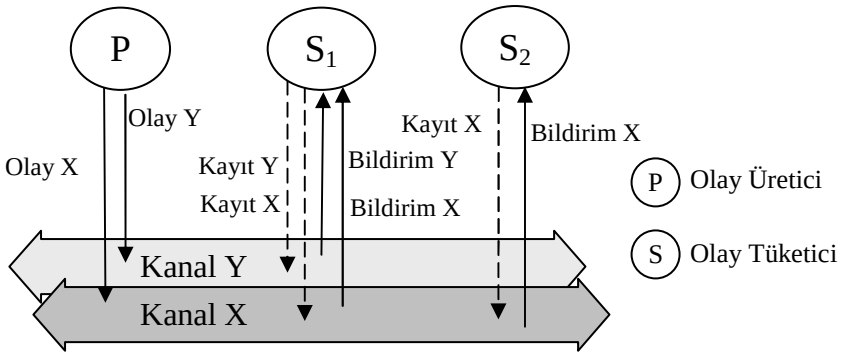
- Kanal Tabanlı Kayıt Mekanizmaları.
- Konu Tabanlı Kayıt Mekanizmaları.
- İçerik Tabanlı Kayıt Mekanizmaları.
- Nesne (Tip) Tabanlı Kayıt Mekanizmalarıdır.

2.5.1. Kanal Tabanlı Kayıt/Yayım Sistemleri

Kayıt/yayım sistemlerinin ilk örnekleri kanal tabanlı olarak geliştirilmiş ve birçok endüstriyel uygulamada kullanılmıştır. Burada "kanal" kavramı sistemdeki yayımcıların üretecekleri, kayıtçıların da kayıt olacakları, olay verilerinin yazılacağı ve/veya alınacağı hattı ifade etmektedir. Bir kayıtçı bir kanala kayıt olduktan (veya o hattı dinlemeye başladıktan) sonra o kanala gelen tüm mesajlardan haberdar olur. Yayımcıda olay verisini üretip amacına uygun kanala göndererek ilgili kayıtçılara ulaşmasını sağlar. Bu kanal adreslerinin önceden tanımlanmış olup yayımcı ve kayıtçıların bunları bilerek kayıt ve yayım işlemleri yapmalarına olanak sağlanmaktadır. Yayımcı tarafından üretilen olay aynı zamanda birden fazla kanala yazılabilir.

Kanal yapısında kanal isimleri IP Multicast[9]'lerin adresi gibi bir ağ adresi olarak görülebilir. Java programlama diliyle gerçeklenen bir haberleşme altyapısı olan IBus [34] kanal tabanlı haberleşme modeline uygun olarak nakil mekanizmasında IP Multicast yapısını kullanmaktadır. Günlük hayatta daha sık karşılaşılabilen bir örnek olaraktan mail-list yapısı da bir kanal tabanlı kayıt/yayım sistemine örnek teşkil edebilir.

[9] Her ne kadar IP Multicast soketleri genelde itme-modelli uygulamalarda kullanılsa da, kanal tabanlı kayıt/yayım modelli sistem tasarımlarında da kullanılmışlardır.



Şekil 2.5 : Kanal Tabanlı Kayıt/Yayım Sistemi

Şekil 2.5'te gösterilen kanal tabanlı kayıt/yayım sisteminde, S_1 kayıtçısı hem X hem de Y kanalına kayıt olmuş, S_2 kayıtçısı ise sadece X kanalına kayıt olmuştur. P yayımcısının ürettiği X ve Y olayları ilgili kanallara gönderilmekte, o kanal üzerine kayıt olan kayıtçılar da gönderilen olay verilerine ulaşabilmektedir. Kanal tabanlı kayıt/yayım sistemlerine, merkezi olay sistemi olarak Java AWT olay modelini [35], dağıtılmış olay sistemi olarak ta CORBA Olay Sistemini [36] örnek olarak verilebilir.

2.5.2. Konu Tabanlı Kayıt/Yayım Sistemleri

Kanal tabanlı kayıt/yayım sistemlerinde kanala ham olarak bir veri bloğu gönderilir ve bu kanala üye olan kayıtçıların bu veriyi alarak işlemeleri beklenir. Her kayıtçı kendi ilgilendiği olaylarla ilgili olan kanallara kayıt olarak olay verilerinin kendine gelmesini beklemektedir. Eğer uygulama platformunda kullanılacak kanal sayısı fazla olursa mevcut Mesaj Transfer Arabirimi mantıksal olarak çok sayıda bölüme ayrılmış olacaktır.

Geliştirilen konu tabanlı (topic based) kayıt/yayım sistemi, kanal tabanlı sistemin daha esnek bir adreslemeye olanak sağlayan modeli olarak görülüp, kayıtçıların ve yayımcıların ortak Mesaj Transfer Arabirimi üzerinden olay verilerini transfer etmelerine olanak sağlamaktadır. Olay verisi, olay mesajına çevrilirken, verinin konusu, olay mesajının başına yerleştirilmiş, kalan kısım ise detay olarak mesajın sonunda bırakılmıştır. Olay sunucuları ve kayıtçılar sadece bu konuyu görebilir ve ona göre işlem yapabilirler. Mesajın detay kısmı, veriyi işleyecek kayıt tarafından açılarak incelenir.

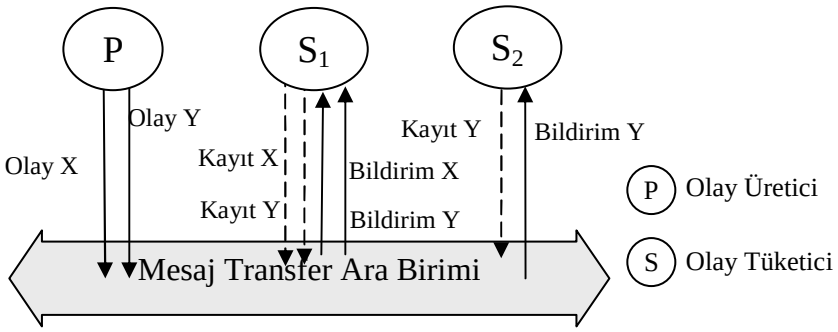
Bir konu üzerine kayıt olmayı günlük hayatta kullanılan bir dergiye veya gazeteye abone olma işlemi gibi düşünülebilir. Abone olan her şahısa, her basımdan birer örnek postayla veya gazete dağıtıcısı tarafından ulaştırılmaktadır. Bu örnekte kayıt olunan "konu" olarak derginin veya gazetenin adı kullanılmaktadır. Hiyerarşik bir yapı olarak incelediğinde ise bir gazetenin, siyaset, ekonomi, spor, magazin gibi, alt

birimleri veya bir derginin ilgili anahtar cümleciklere göre bölümlenmiş alt birimleri kayıt için kullanılan konular olarak ta düşünülebilir.

Konu tabanlı kayıt/yayım terimi de, *özne tabanlı*[10] kayıt/yayım terimiyle eşanlamlı olarak kullanılsa da bazı araştırmacılar özne tabanlı kayıt/yayım yapısında konu isimlerinin hiyerarşik bir yapıda [37] tutulması sebebiyle konu-tabanlı kayıt/yayımdan farklı olduğu görüşünü savunmaktadırlar. Özne tabanlı kayıt/yayım sistemi programcıya olay sisteminde kullanacağı konuların birbirlerini kapsama ilişkisini de koyabileceği daha hiyerarşik bir konu tanımlama modeli sunmaktadır. Konu tabanlı kayıt/yayım sistemleri, *Düz Konu Tabanlı modeller* ve *Hiyerarşik Konu Tabanlı Modeller* olarak iki farklı sınıfta incelenebilir.

2.5.2.1. Düz Konu Tabanlı Model

Düz konu modelinde, diğer kayıt/yayım modellerinde olduğu gibi, yayımcı, sisteme ürettiği olayla ilgili veriyi gönderir. Olay verisinin başında, verinin özelliğini/konusunu içeren bir başlık/konu alanı vardır. Genel yapısı Şekil 2.6'da görülen düz konu modelinde, Mesaj Transfer Ara Birimi ve sisteme kayıtlı olan kayıtçılar, sisteme gönderilen olay verisinin içeriğini bilmez, sadece olay verisinin başlığını yani konusunu görebilir. Görmüş olduğu bu konuya göre bu olay verisi ile ilgilenip ilgilenmediğine, ilgileniyor ise hangi olay tüketicilerine göndereceğine karar verir.



Şekil 2.6 : Düz Konu Modeli

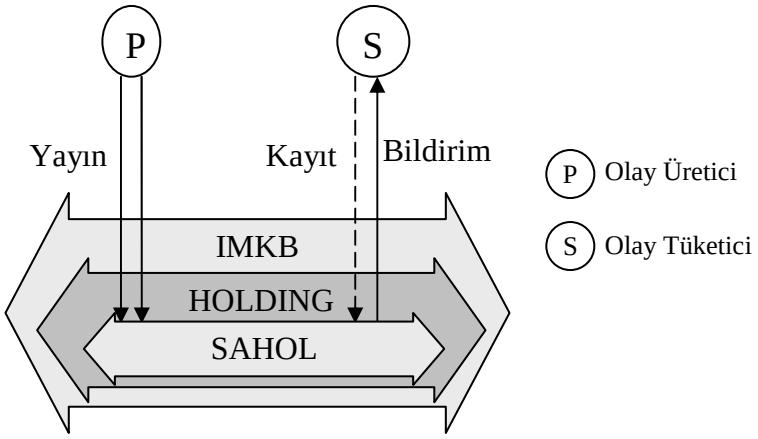
Konu tabanlı olarak geliştirilen çoğu kayıt/yayım haberleşme sisteminde, JEDI [38, 39], olay sisteminde olduğu gibi, düz konu tabanlı kayıt mekanizması kullanılmaktadır.

2.5.2.2. Hiyerarşik Konu Tabanlı Model

Hiyerarşik konu tabanlı modelde kayıtçı ile olay sistemi arasındaki kayıt mekanizmaları hiyerarşik olarak konulara bölümlenmiştir. Bir ağaç yapısına benzeyen bu modelde bir uçtaki konu, kendi altındaki bütün konuları kapsamaktadır.

[10] subject based

Diğer bir deyişle bir uca kayıt olan bütün kayıtçılar aynı zamanda o ucun altında yer alan konulara da kayıt olurlar. Benzer şekilde bir uca gelen bir mesaj o ucun hiyerarşik olarak üstünde bulunan konulara da gelmiş olarak işlem görür. Kök uç (ana konu) bir konudan oluşabileceği gibi, birden fazla kök ucun kullanılmasına da olanak sağlanabilir. Bu şekildeki kayıt mekanizmasında konu isimleri genellikle URL-benzeri gösterimlerle ifade edilebilmektedir .



Şekil 2.7 : Hiyerarşik Konu Modeli

Şekil 2.7'de görüldüğü gibi, bir konu "/IMKB/HOLDING/SAHOL" şeklinde hiyerarşik olarak ifade edilebilmektedir. Bu şekilde SAHOL konusu üzerine gelen bir olay verisi aynı zamanda HOLDING ve IMKB konusunda da gelmiş gibi işlem görmekte ve bu konular üzerine kayıtlı yayımcılara da bu olay verisi ulaştırmaktadır.

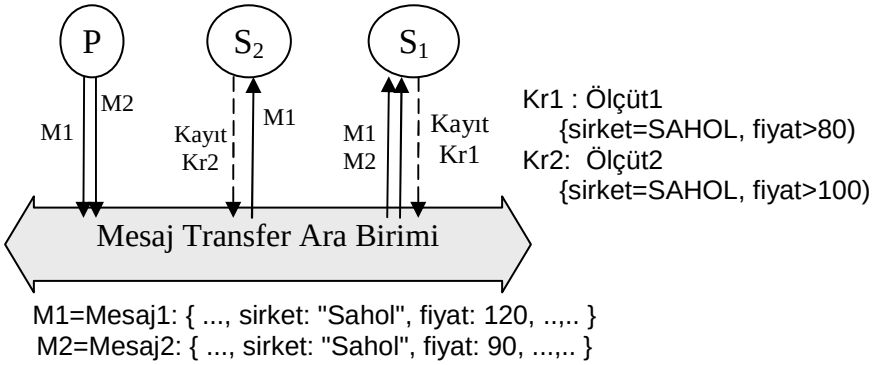
Bu modele uygun olarak geliştirilen SwiftMQ [40] mesajlaşma platformu tamamıyla Java Message Service (JMS) 1.0.2 [41] tabanlı olarak geliştirilmiş, hiyerarşik konu tabanlı kayıt/yayım modelini de destekleyen uçtan uca haberleşmeye de olanak sağlayan ticari bir uygulamadır.

2.5.3. İçerik Tabanlı Kayıt/Yayım Sistemleri

Konu tabanlı modellerin sınırlamaları olan mesajın içeriğiyle ilgilenilememesi ve içeriği filtreleme işlemlerinin zorluğu, konu tabanlı sistemlerin genişleyebilirliğini ciddi ölçüde etkilemektedir. Geliştirilen içerik tabanlı kayıt/yayım modeli sayesinde olay sunucuları olay mesajının içeriği tamamıyla görülebilmekte ve hangi kayıtçının hangi veriyi alabileceği bu içeriğe göre filtrelenebilmektedir [42]. Kayıtçılar ilgilendikleri olayları ifade ederken artık olayın konusundan değil, olayın özelliklerinden bahsetmektedirler. Olaylar sabit ölçütlere göre sınıflandırılmamış (konu parametresi sadece "SAHOL" veya "DOHOL" alır gibi), çalışma zamanında

değişebilen parametrik değerlere göre de kayıt olma imkanı sağlanmıştır[11]. Her kayıtçı sadece kayıt olduğu özellikleri taşıyan olaylardan haberdar edilmekte, diğer olay mesajları dikkate alınmamaktadır.

Şekil 2.8'de içerik tabanlı bir olay haberleşme modelinin diyagramı görülmektedir. Burada mesaj uzayı tek bir konu olarak düşünülebilir. Her kayıtçı kendine özel ölçütleri belirleyerek bu ölçütler üzerinden sisteme kayıt olurlar. S_1 kayıtçısı Sabancı Holding hisselerinin fiyat 80 TL sının üzerindeyken haberdar edilmek istemiştir. S_2 kayıtçısı da Sabancı Holding hisselerinin fiyat 100 TL sının üzerindeyken haberdar edilmek istemiştir. Üretilen mesajlar M_1 ve M_2 de bu nitelikler görülebilmektedir. Mesaj Transfer Arabiriminin içinde bu mesajları kontrol eden olay sunucuları, M_1 mesajını hem S_1 hem de S_2 sunucusuna gönderirken, gerekli filtreyi sağlamadığı için M_2 mesajı sadece S_1 kayıtçısına ulaşmaktadır.



Şekil 2.8 : İçerik Tabanlı Kayıt/Yayım Sistemi

Kayıtçılar ilgilendikleri olaylara isim-değer ikililerini kullanarak kayıt olurlar. Bu ikililer uygun mantıksal operatörlerle birleştirilerek, bileşik kayıt modelleri oluşturabilirler. Bu şekilde oluşturulan (isim-değer ikilisi) kayıt mekanizması sayesinde filtreleme işlemleri kolaylaştırılmaktadır.

Kayıt/yayım modellerinde kayıtçıların sisteme dahil olmak, belirli olayların oluşması durumunda haber verilmesini sağlamak için sisteme, ilgi duydukları olayı/olayları ifade eden bir kayıt mesajı göndermeleri gerekmektedir. İçerik tabanlı kayıt/yayım modelinde kullanılan genelde iki temel kayıt modeli vardır. Bunlar:

- *Katarlar*: En sık karşılaşılan kayıt modelleri katar olarak sisteme gönderilmektedir. Filtreleme işlemleri SQL veya OMG'nin Filtreleme Kısıt Dili gibi, belirli bir kayıt gramerine uygun olarak ifade edilmekte, ve daha sonra da bu karakter dizisi sistem tarafından ayrıştırılmaktadır.

[11] Her nitelik için farklı ölçütlerle kayıt olma imkanı sağlanmıştır, "fiyatı 5000 TL sından fazla olan hisse senetleri" gibi

- *Şablon Nesneler:* Tuple tabanlı karşılaştırma modellerinden esinlenilmiştir. Kayıt olurken, kayıt olan nesne, kendi ilgi duyduğu ölçütleri içeren *T* tipindeki bir *t* nesnesi oluşturarak sisteme gönderir. Bu *t* nesnesindeki ölçütlere uyan olay mesajları kayıtçıya gönderilir. Sistem üzerinde "match()" benzeri bir metot çalıştırılarak bunun seçtiği olay verileri kayıtçıya gönderilir.

ELVIN [43], SIENA [22], LESUBSCRIBE [25], GRYPHON [24] ve REBECA [44] gibi, çoğu içerik tabanlı kayıt/yayım sisteminde olaylar basit veri tiplerinden oluşan niteliklerin bir kümesi olarak görülmektedir.

İçerik tabanlı kayıt modelini de tek bir dergiye abone olmak şeklinde yorumlanabilir. Dergiye tamamen üye olmak yerine ilgi duyulan alanlar ile ilgili makalelere, özel anahtar kelimeleri içeren makalelere, belirli yazarların yazdıkları makalelere, belirli boyutu aşmayan makalelere abone olma şeklinde gerçek hayattan örneklenebilir.

2.5.4. Nesne-Tip Tabanlı Kayıt/Yayım Sistemleri

Yukarıda bahsedilen üç genel yaklaşım ile ifade edilen modellerde uygulama tanımlamalı (application defined) olay modellerine izin verilmemekte, sadece sisteme daha önceden belirlenmiş modellere uygun olay mesajlarının gönderilmesine olanak sağlanmaktadır. Olay türüne bağlı olarak veri akışı değil, sistemde tanımlı mesaj modeli üzerindeki belirli parametresel değerlere göre olay mesajlarının dağıtılması yapılmaktadır. Üretilen olay türüne bağlı olarak olay verilerinin dağıtımını yapabilmek için geliştirilen Tip tabanlı kayıt/yayım sisteminde (type-based publish/subscribe) [45] üretilen olaylar uygulamalar tarafından tanımlanmış sınıfların[12] örneği olacak birer nesne olarak modellenmiştir. Bu modelde kayıtçılar sisteme, kendi ilgi duydukları sınıflar üzerinden kayıt olurlar ve bu nesnelerin genel (public) değişkenleri üzerinde kolaylıkla içerik tabanlı sorgulama ve filtreleme yapabilirler.

Tip tabanlı olay sistemi, sistemin bileşenlerinde kullanılması için herhangi bir sabit olay tipi geliştirmediği için yayımcılar tarafından üretilecek olay verilerinin de belirli bir şablona uygun olmasına gerek yoktur. Genellikle geliştirilen sistemlerde olaylar alt seviyede mesajlar olarak makineler arasında dolaşırlarken, bu sistem sayesinde nesneye yönelik bir yaklaşımla, makineler arasında nesnelerin dolaşmasına, böylece daha esnek bir yapı oluşturulmasına olanak sağlanmaktadır[13].

[12] bu modelde tanımlanmış sınıf kavramı, olay tipi olarak kullanılmaktadır
[13] Konu ile ilgili detaylar döndüncü kısımda verilmektedir.

3. ETMENLER

Bilgisayarlar ne yapacaklarını bilme konusunda çok iyi değillerdir. Onların düşünme ve muhakeme yetenekleri yoktur. Yapacakları her eylemin programcı tarafından önceden görülmüş, planlanmış ve kodlanmış olması gerekmektedir. Eğer bir bilgisayar programı, programcısının önceden öngörmediği bir durumla karşılaşırsa sonuç pek hoş olmaz (en iyi ihtimal sistem hata mesajı verir ve çalışmasına devam eder, en kötü ihtimal sistem çöker ve ciddi veri kaybına uğranır). Sıradan gerçek olan "Bilgisayarın insan gibi karar verememesi, karşılaşılmayan olaylara uygun tepkilerde bulunamaması ve muhakeme yeteneğinin olmaması", insanoğlunun bilgisayar ile olan ilişkisinin merkezinde yer almaktadır.

Bilgisayarlar ve onların üzerinde çalışan programlar bu gibi insansal özelliklerden ziyade itaatli, hazır bilgi içeren ve yaratıcı olmayan hizmetçiler olarak görülmektedir. Günümüzdeki birçok uygulama alanında ise bu tip programlar hayatımızı kolaylaştırıcı olarak kabul görmekte ve kullanılmaktadır. Fakat hızla gelişen teknoloji neticesinde, insanoğlu her türlü ihtiyacında bilgisayar kullanmaktadır. Böyle bir ortamda kendi kendine karar verebilen sistemlere olan ihtiyaç hızla artmaktadır. Bu gibi sistemler ve yazılımlar ise genel bir tabir olarak etmenler olarak adlandırılmaktadır.

Son yıllarda etmen kavramı üzerine çok sayıda çalışma yapılmış ve etmen tabanlı çok sayıda uygulama geliştirilmiştir. Bu konuları incelemeden önce etmen kavramını biraz daha detaylı incelemekte fayda vardır. Etmen kavramı üzerinde evrensel kabul gören tanım bulunmamaktadır. Fakat bu konuda çalışma yapan araştırmacıların ve bilim adamlarının temelde üzerinde anlaştıkları konu etmenlerin **özerk** varlıklar olmasıdır. Onun haricindeki özellikler ise etmenin çalıştığı platformun bir ihtiyacı olarak kullanılmakta veya kullanılmamaktadır.

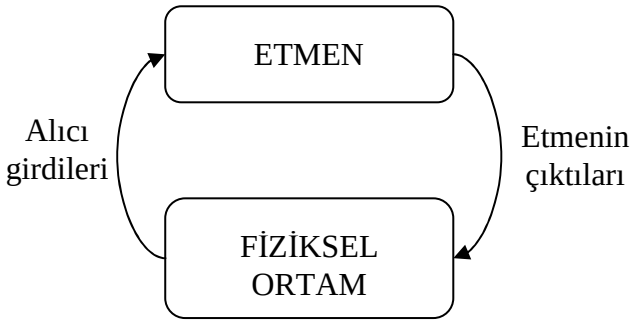
Yine de genel bir tabir olarak etmenin tanımını kabul etmek gerekirse, Woolridge ve Jennings'in yapmış olduğu etmen tanımı dikkate alınabilir [10]. Onlara göre

> ***Etmen**; bir fiziksel çevre içinde yerleşmiş olan, kendi tasarım amaçlarına uygun olarak özerk eylemler yapabilme yeteneği olan bilgisayar sistemleridir / yazılımlarıdır.*

Aynen etmen kavramının tanımı üzerindeki muğlaklık gibi özerk kavramında da muğlaklık olmaması için bunu biraz daha detaylandırmak gerekmektedir. Woolridge'e göre

> ***Özerklik**; bir etmenin herhangi bir insanın veya başka bir sistemin müdahalesi olmadan, kendi durum değişkenlerinde, davranışlarında ve eylemlerinde kontrole sahip olmasıdır.*

Şekil 3.1'de etmen kavramının üst seviye ifadesi verilmiştir. Bu diyagramdan da görüleceği gibi, bir etmen yaptığı eylemler sonucunda kendi çalışma ortamını ve fiziksel çevresini etkileyebilmektedir. Gerçek hayatta ise belirli karmaşıklığın ötesindeki sistemlerde etmen kendi çalışma ortamı üzerinde tam bir kontrole sahip olmalarına izin verilmemektedir. Örneğin bir elektronik ticaret uygulamasında [46] etmen gerekli tedarikçileri bularak mal temini için gerekli rezervasyon işlemlerini ayarlamakta, ancak paranın ödenmesi ve malın teslim alımı işlemleri için mutlaka kullanıcı müdahalesine/onayına ihtiyaç duyulmaktadır. Bu gibi telafisi güç işlemlerde etmen sistemine en fazla belirli alanlarda çalışma ortamını etkileyecek kısmi kontrol yetkisi verilmektedir.



Şekil 3.1 : Fiziksel Ortamında Bir Etmenin Çalışması

> ***Bir etmenin repertuarı;** o etmenin amacına ulaşmak için, çevreden algıladığı ve kendi bilgi tabanındaki verilere göre yerine getirebileceği eylemler kümesini ifade etmektedir.*

Bu repertuar etmenin çevreyi etkileyebilme kapasitesini göstermektedir. Burada karşımıza çıkan temel problem ise etmenin daha önce karşılaşmadığı ve/veya etmen geliştirici tarafından tanımlanmamış olan bir problem karşısında, tasarım amacına uygun olarak bu eylemlerden hangisini seçeceğine karar vermesidir. Bu karar verme süreci çeşitli çevresel değişkenler tarafından etkilenmektedir.

> ***Bir etmenin çevresel değişkenleri;** o etmenin karar verme sürecini etkileyebilecek olan donanımsal ve / veya yazılımsal varlıklardır.*

Russel ve Norvig bu çevresel değişkenleri şu şekilde sınıflandırmışlardır [47]:

- ***Erişilebilir-Erişilemez:*** Erişilebilir çevrede çalışan etmen, bu ortam ile ilgili olan tüm verilere güncel olarak erişebilir. Günümüzde gerek gerçek dünyadaki değişkenler, gerekse Internet gibi geniş alan ağları gibi karmaşık ortamların değişkenleri ise erişilemez olarak değerlendirilmektedir. Burada ayırım mantıksal 0 veya 1 (doğru veya yanlış) şeklinde yapılmamaktadır, kısmi erişilebilir değişkende olmaktadır. Bir sistemdeki değişkenler ne kadar erişilebilir ise o çevrede çalışacak etmen geliştirmek o kadar kolaydır.
- ***Kararlı-Kararsız:*** Kararlı bir çevrede yapılacak her eylemin oluşturacağı tek bir etki vardır, yani çalıştığı çevre üzerinde aynı eylem bir kez yapıldığında A etkisini, diğer kez yapıldığında ise B etkisini yapabilme durumu yoktur. Gerçek dünyada bir eylemin etkileyebileceği çok sayıda değişken olması sebebiyle karasızlık hakimdir. Kararsız sistemler ise etmen tasarımcıları için ciddi problemler oluşturmaktadırlar.
- ***Epizodik-Epizodik Olmayan:*** Epizodik ortamda bir etmenin performansı bazı somut olaylara bağımlıdır, başka bir senaryodaki bir etmenin performansı ile bağlantısı yoktur. Epizodik ortamlar bir etmenin mevcut durum karşısında hangi eylemi yapacağına kolay karar verebilmesi açısından etmen geliştiriciler tarafından tercih edilen ortamlardır.
- ***Statik-Dinamik:*** Statik çevre, etmenin yaptığı değişiklik haricinde herhangi bir yazılım ve/veya donanımsal varlık tarafından değişenlere erişilemediği çevredir. Dinamik çevrede ise ortamı etkileyen değişik faktörler (etmenler) bulunmaktadır. Gerçek dünya dinamik bir özelliğe sahiptir ve böyle dinamik ortamlar etmen geliştiriciler için zor ortamlardır.
- ***Ayrık-Sürekli:*** Bir ortam eğer sonlu sayıda eyleme ve anlayışa sahip ise ayrık[14] olarak tanımlanır. Bir satranç oyunu ayrık ortama örnek olarak verilirken, taksi sürme eylemi ise sürekli ortama bir örnektir.

3.1. Etmen Örnekleri

Etmenler, getirmiş oldukları yeni yaklaşımla yazılım dünyasında hemen hemen her alanda değişik şekillerde etkin hale gelmişlerdir. Etmen paradigması, soyut ve esnek yapısı sayesinde karmaşık yazılım sistemlerinin geliştirilmesinde de bir çığır açmıştır. Etmenlerin etkin olarak kullanılabildiği bazı örnekleri şu şekilde sıralanabilir:

[14] discrete

- *Bir kontrol sistemi*, basit bir etmen olarak düşünülebilir. Bunun en basit örneği olarak ise bir klima verilebilir. Klimada oda ısısını kontrol eden bir alıcı vardır. Bu alıcı iki tür sinyal gönderir. Oda ısısı düşük, oda ısısı uygun. Bu sinyallere göre termostatın yapacağı eylem ise belirlidir. Isıtmayı aç, ısıtmayı kapat.

  Oda ısısı düşük → ısıyı aç

  Oda ısısı uygun → ısıtmayı kapat

  Bir uzay mekiğinin kontrol sistemi veya bir nükleer reaktörün kontrol sistemi gibi, karmaşık kontrol sistemlerinin ebette daha zengin karar yapılarına sahip olmaları gerekmektedir.

- *Elektronik Ticaret: B*ir elektronik ticaret sisteminde, müşteri aradığı ürün veya ürünlerin listesini bir etmene yerleştirmek suretiyle onun kendi adına alış veriş yapmasına olanak sağlanır. Etmen, kendine verilen yetkiye göre ya ürünleri doğrudan satın alır, ya da ürünlerin adres ve fiyat bilgilerini toparlayarak müşterinin bilgisine değerlendirilmek üzere sunar.

- *Paralel Çalışım:* Etmenler kendi kopyalarını oluşturarak bunları sistem üzerindeki boş durumdaki diğer makinelere kopyalayarak, üzerlerine yüklenmiş olan görevleri alt parçalara bölerek çok sayıda makinenin dağıtılmış ve paralel olarak çalışmasına olanak sağlayabilirler.

- *Bekçi Köpeği (Watchdog) Uygulamaları:* Etmen bir sistem üzerindeki bileşeni takip ederek onun daha önce belirlenen modele uygun şekilde işlemesini sağlar. Örneğin bir ağ yönetim etmeni, ağdaki trafiği kontrol ederek, trafik yoğunluğunun belirli ölçütlerin üstüne çıkması durumunda, veya ağ tıkanıklığının olması durumunda, sistem yöneticisine cep telefonu mesajı olarak veya elektronik posta ile bilgi vererek bu problemin ortadan kaldırılmasını sağlayabilir.

- *Yazılım deamonları* (çoğu işletim sistemlerinde arka planda çalışan görevcikler gibi). Bunlar yazılım ortamının izleyerek gereksinim duydukları düzenlemeleri yaparlar. Örneğin, Outlook Express programı bir mail geldiğinde görsel veya işitsel olarak kullanıcının uyarılması görevini arka planda sağlamaktadır.

- *Acenteler.* Bir seyahat planlaması yapılması durumunda seyahat yapmak istediğiniz zamanda gidilebilecek en uygun yere, uygun fiyat ve şartlarda gitmenize olanak sağlamaktadır. Bunu başarabilmek içinde otellerin fiyatlarını, özelliklerini, seyahat için gerekli uçak, tren, demiryolu sefer tarifelerini, toplu taşıma aracı imkanlarını, bölgesel hava durumları gibi bir

çok etken faktörü takip ederek müşteri için en uygun seçimin yapılmasına olanak sağlamaktadır.

Bunların haricinde etmenler, telekomünikasyon, üretim, finans, eğitim, hava trafik yönetimi, savunma simülasyonları ve karar destek sistemleri gibi çok sayıda farklı alanda günümüzde halen aktif olarak kullanılmaktadır.

3.2. Zeki Etmenler

"Zeki etmen nedir?" sorusu aslında biraz da "Zeka nedir?" sorusu ile yakından alakalıdır. Buna cevap vermek pek kolay değildir. Etmenler üzerine ciddi çalışmaları olan Woolridge ve Jennings zeki etmen kavramını şu şekilde izah etmiştir [10]:

> ***Zeki Etmen;*** *tasarım amaçlarına uygun olarak esnek özerk eylemleri olan etmenlerdir.*

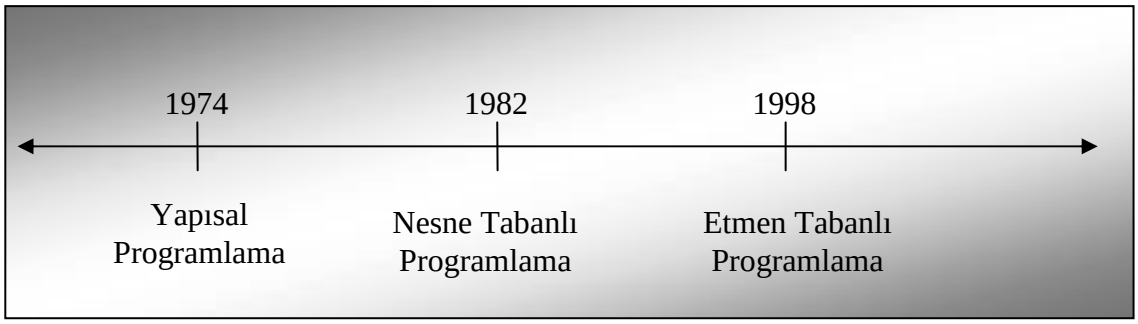
Esnek eylemler kavramı ile de, bir zeki etmende özerkliğin haricinde şu üç özelliğinin olması gerekmektedir:

- ***Tepkisellik:*** Zeki etmenler mevcut çalışma ortamını alıcıları vasıtası ile algılayarak tasarım amacına uygun olarak bu ortamı değiştirmek için meydana gelen olaylara tepkide bulunmaktadırlar.
- ***Aktif Olma (amaca yönelik davranma) :*** Zeki etmenler tasarım amacına uygun olarak, çalışma inisiyatifini ellerinde tutarak, amaca yönelik davranışlarda bulunurlar. Amaca yönelik davranış gösterecek bir sistem geliştirmek fazla karmaşık değildir. Basit anlamda Pascal'da bir prosedür, C de bir fonksiyon veya Java'da bir metot yazmak bu işlevi yerine getirebilmektedir. Böyle bir yordam geliştirilirken, bu yordam bazı ön koşullara bağlanarak bunların gerçeklenmesi durumunda çalışması, bazı son koşullara bağlanarak, çalışma sonucunda bu hedeflere (belirli komutların çalıştırılması) ulaşılması amaçlanır. Bunların yerine getirilmesi sırasında yapılan değişiklikler/hedefler yordamın amacı olarak görülebilir.
- ***Sosyal Yetenek:*** Zeki etmenler, diğer etmenlerle (veya insanlarla) tasarım amaçları yerine getirmek için iletişimde bulunurlar. Günümüzde milyonlarca bilgisayar diğer bilgisayarlarla veya insanlara veri alışverişinde bulunuyorlar. Fakat bu varlıklar arasında sekizli dizileri ile yapılan veri alışverişi aslında sosyal bir yetenek değildir. Gerçek hayat ile karşılaştırıldığında, bir amaca ulaşmak için değişik kişilerle veya kuruluşlarla beraber çalışılması gerekebilir (her ne kadar farklı amaçlara sahip olunsa da). Bunun için bu kişilere işin yaptırılması için onlarında amacını karşılayacak şekilde müzakere yapılması ve bir anlaşmaya ulaşılması (para karşılığı iş yaptırılması) gerekebilir. Benzer

şekilde etmenlerin de kendi amaçlarına ulaşabilmeleri için başka etmenlerle uygun bir etmen haberleşme dili[15] ile bağlantı kurarak müzakere etmesi, konuşması sosyal bir yetenek olarak görülmektedir.

3.3. Etmen mi? Nesne mi?

Başlangıçta direk olarak düz kodlamayla başlayan bilgisayar programlama modellerinin tarihsel süreçte gelişimini Şekil 3.2'de gösterildiği gibi izah edilebilir.



Şekil 3.2 : Programlama Modelleri

Günümüzde, etmenlerle ilgili çıkan yeni fikirlerin, Nesne Tabanlı Programlama yaklaşımıyla çözülmeye çalışmanın başarıya ulaşmadığı görülmektedir. Nesneler bazı durum değişkenlerini kapsayan, bazı eylemleri yerine getiren ve mesajlaşma ile haberleşen hesaplama birimleri olarak görülebilir. Bazı araştırmacılar (özellikle nesneye yönelik programcılar) etmen kavramının nesneye yönelik programlama kavramının ötesinde bir şey getirmediğini düşünmektedirler. Nesne kavramının genel bir tanımını şu şekilde yapılabilir:

> ***Nesne,*** *kendi durum değişkenlerini kapsayan ve bu değişkenlere bağımlı olarak bazı eylemleri yapma yeteneği olan (davranışları yapan, metotları çalıştıran) ve mesaj transferi ile haberleşebilen hesaplama varlıklarıdır.*

Etmenler ve nesneler incelendiğinde benzerlikleri olmasına rağmen, yakından bakıp, detaya inildiğinde ciddi farklılıkları göze çarpmaktadır. Bu farklılıkları James Odell üç sınıf altında incelemiştir [48].

1. En temel fark olarak etmenlerin ve nesnelerin özerklik dereceleri görülebilir. Bir nesneye bakıldığında onun kapsama özelliğinden dolayı nesne kendi içindeki durum değişkenlerine erişim yetkisine sahiptir ve bu değişkenler harici nesneler tarafında görülemez. Her ne kadar bu değişkenler *genel değişkenler* olarak ta tanımlanıp, diğer nesneler tarafından da erişilebilmelerine olanak sağlansa da, bu kullanım zayıf programlama stili

[15] Etmen haberleşme dili, farklı sistemler tarafından geliştirilmiş etmenler arasındaki haberleşme için kullanılır.

olarak görülmektedir. İlk bahsedilen şekildeki kullanımda nesnenin kendi durum değişkenleri üzerindeki özerkliğinden bahsedilebilir, ancak bu değişkenlere erişim sağlamak veya belirli davranışları yerine getirmek için nesnenin *genel* tanımlanmış olan metotları (davranışları) vardır ve bu davranışlar üzerinde tam bir özerkliği yoktur.

Etmenlerde ise olay tamamıyla farklıdır. Bir E_1 etmeni başka bir E_2 etmeninin bir şeyler yapmasını isteyecekse bunu mesajla talep olarak kendine bildirir. E_2 etmeni bu talebi yerine getirmek zorunda değildir. E_2 etmeni bu talep kendi amacına uygun ise yerine getirir, değil ise yapmaz. E_1 etmeni bu eylemin yapılmasını cazip hale getirmek, yani bu görevin aynı zamanda E_2'nin de bir amacı olmasını müzakereler ile sağlayabilir (verilen hizmetin karşılığı olarak kendisi de onun için bir eylem yapabilir, örneği para transferi).

Nesnelerde karar mekanizması, metodu çağıran harici nesnede iken (yani özerklik dışarıda), etmenlerde ise özerklik etmenin içindedir. Bu nedenle bazı araştırmacılar, biraz da slogansal olan şu tabiri kullanmaktadırlar:

Nesne beleş yapar, etmen para için yapar.

2. Bir etmenin esnek özerk eylemleri olan, tepkisellik, amaca yönelik olması ve sosyal yeteneğin kullanılmasını da temel bir fark olarak gösterilebilir. Standart nesne tasarımda bu tip davranışların tasarlanmasına gerek yoktur (nesnenin kullanım amacına göre geliştirilebilir olmasına rağmen). Ancak bu eylemler, özellikle zeki etmenler için mutlak olması gereken özelliklerdir.
3. Çalışma kontrolü açısından da etmenler ve nesneler birbirlerinden ayrılmaktadırlar. Standart nesneye yönelik programlamada tekli bir çalışma kontrolü[16] varken, etmenlerde (özellikle çoklu etmenlerde) çoklu çalışma kontrolü[17] vardır Her ne kadar nesneye yönelik programlamada eşzamanlı çalışım özellikleri varsa da, bunlar yine etmen kavramındaki çoklu çalışma kontrolünü karşılamamaktadır. Çalışma kontrolü açısından etmen kavramına en yakın olabilecek özerkliğin biraz daha artırıldığı aktif nesnelerdir [49].

3.4. Gezgin Etmenler

Bir yazılım etmeni, bir kullanıcı adına çalışarak, belirli işlevleri yerine getiren bir programdır. Bunu gerçek hayattaki bir örneği olarak ta bir acentenin, müşteri adına belirli araştırmaları, rezervasyonları, bilet alım işlerini halletmesine benzetilebilir. Bu temel kavramı referans alındığında;

[16] single thread of excution
[17] multi-thread of execution

Gezgin Etmen; *bir ağ üzerinde (Internet, yerel alan ağı, geniş alan ağı vs), kullanıcı adına belirli işlevleri yerine getirirken, özerk olarak kendisini bir makineden, başka bir makineye taşıyarak (göç ettirerek) görevine /araştırmalarına yeni yerleşkesinde devam edebilen etmendir,*

Şeklinde tanımlanabilir. Gezgin etmenin hangi görevleri yerine getireceği, hangi makineler üzerinde dolaşacağı, nasıl bir karar verme stratejisi uygulayacağı, ilk üretim zamanında gezgin etmene, kullanıcının talepleri doğrultusunda etmen sistemi tarafından yüklenir. Bu veriler, etmen farklı uçlar arasında dolaşırken edindiği bilgilere göre yeniden düzenlenebilir. Bu aşamadan sonra etmenin hareketi amacına göre özerk olarak devam eder.

3.4.1. Tarihsel Süreç

Taşınabilirliğin evrimsel gelişme sürecinin ilk adımında dosyaların bir makineden diğer bir makineye değişik protokollerle (örneğin FTP) transferi amaçlanmıştır. Bu protokollerde karşı makineye bağlanarak gerekli güvenlik kontrollerinden sonra istenilen dosyanın yerel makineye indirilmesi sağlanmaktadır. Ancak genel kabul gören hiyerarşik ilerlemede şu dört ana yapı kullanılmaktadır, *Uzaktan Yordam Çağırma, Uzaktan Değerlendirme, Talepte Kod Alma ve Gezgin Etmenler.* Bir hesaplamanın yapılması için:

- görevi tanımlayan *yapabilme bilgisinin*[18],
- hesaplama için kullanılacak *kaynakların*,
- ve görevi yerine getirecek *çalışan birimin*,

aynı hesaplama ortamı üzerinde bulunmaları gerekmektedir.

Aşağıda detaylandırılacak olan taşınabilir yapıları incelemeden önce temel bir problemi gösterilerek, bu problemle ilgili yapıların nasıl çözüm getirdikleri incelenmektedir. Temel problem şudur. "Aslı ve Bilge, birlikte çalışma ve etkileşimle bir çikolatalı kek yapmak istemektedirler. Bir kekin yapılabilmesi için (hizmetin sonucunu almak):

- bir tarife ihtiyaç vardır (hizmet ile ilgili yapabilme bilgisi),
- kek için gerekli malzemeye (taşınabilir kaynaklar),
- keki pişirmek için bir fırın (taşınması güç bir kaynak),
- ve malzemeleri, tarife göre birleştirecek bir kişiye (kodu çalıştırmaktan sorumlu hesaplama bileşenine) ihtiyaç vardır.

[18] know-how

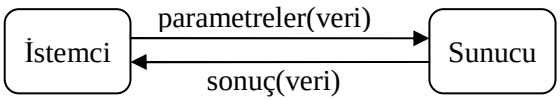
Keki hazırlamak için (hizmeti vermek /görevi yerine getirmek için) bütün bu elemanların aynı yerde toplanmaları gerekmektedir.” Verilen örneklerde A, Aslı bileşenin rolünü yerine getirmektedir. Aslı etkileşimi başlatan ve hazırlanan çikolatalı keke (hizmet sonucuna) ulaşmak isteyen kişidir. Tablo 3.1’de bu bileşenlerin ilgili yapılarda çalışmadan önce ve sonra nerede bulundukları gösterilmiştir. Bu yapıların detayları devam eden kısımlarda açıklanmaktadır.

Tablo 3.1. Taşınabilir Kod Paradigmaları

Paradigma	**Önce**		**Sonra**	
	S_A	**S_B**	**S_A**	**S_B**
İstemci Sunucu	A	Yapabilme bilgisi Kaynak B	A	Yapabilme bilgisi Kaynak B
REV	Yapabilme bilgisi A	Kaynak B	A	Yapabilme bilgisi Kaynak B
Talepte Kod	Kaynak	Yapabilme bilgisi B	Kaynak Yapabilme bilgisi A	B
Gezgin Etmen	Yapabilme bilgisi A	Kaynak	---	Yapabilme bilgisi Kaynak B

3.4.1.1. Uzaktan Yordam Çağırma

Uzaktan Yordam Çağırma (Remote Procedure Call-RPC) işlemiyle, basit dosya transferinden öte çalışan bir programın, çalışma akışında kontrolün sunucu makineye geçmesi sağlanmıştır. Bu modelde istemci, sunucu üzerindeki bir metodu çağırırken, o metodun ihtiyaç duyduğu parametreleri de karşı tarafa göndermektedir (Şekil 3.3). Sunucu bu parametreleri aldığında artık çalışan programın kontrolü kendi elindedir. Diğer taraf sunucudan gelecek olan sonuç değerini beklemektedir.



Şekil 3.3 : Uzaktan Yordam Çağırma

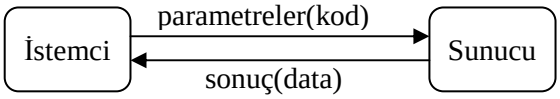
Bu paradigma gerçek hayatta şu şekilde örneklenebilir.

“Aslı çikolatalı kek yapmak istemektedir, fakat elinde kek tarifi olmadığı için nasıl yapıldığını bilmemektedir. Aynı zamanda evinde ne gerekli malzeme ne de fırın bulunmamaktadır. Ancak bilmektedir ki arkadaşı Bilge, çikolatalı kekin nasıl yapıldığını bilmekte ve evde iyi hazırlanmış bir mutfağı bulunmaktadır. Bilge’nin talep gelmesi durumunda kek hazırlamaktan memnun olacağını bildiği için, Aslı Bilge’ye telefon açmakta ve ona “kendisi için bir kek yapıp, yapamayacağını”

sormaktadır. Bilge keki hazırlamakta ve daha sonra bunu Aslı'ya göndermektedir."

3.4.1.2. Uzaktan Değerlendirme

Her ne kadar uzaktan yordam çağırma modelinde "uzak bir makinedeki metot çağrısının yerel makinedeki metot çağrısı gibi yapmak" amacı gerçeklenmesi basit bir amaç olsa da bu model bilgisayar biliminde yeni yaklaşımlar için güçlü bir etkiye sahip olmuştur. Bunun üzerine bir sonraki adım olarak ta basit bir veri (parametre) transferi yerine, çalışacak programın karşı tarafa gönderilmesi fikri ortaya çıkmıştır. Şekil 3.4'te görüldüğü gibi, sunucu makinede çalıştırılması istenen program parçası, parametre olarak gönderilerek bu program parçasının çalıştırılması ve sonucunun istemciye gönderilmesi beklenir. Bu model Uzaktan Değerlendirme (Remote Evaluation-REV)[50] olarak adlandırılmaktadır.



Şekil 3.4 : Uzaktan Değerlendirme

Uzaktan değerlendirme paradigmasında programı çalıştıran bileşen, *istemci*, görevin yapabilme bilgisine (know-how) sahiptir, fakat bunu yerine getirilmesi için ihtiyaç duyulan kaynak uzaktaki bir makinededir (*sunucu*). Bu nedenle istemci, kendisindeki yapabilme bilgisini sunucuya göndererek, bu program parçasının karşı tarafta çalıştırılıp sonucun kendine gönderilmesini bekler. Sunucu gönderilen kodu çalıştırarak kendi yerel verilerine daha hızlı erişim ile istenen sonuca ulaşarak, bunu istemciye gönderir.

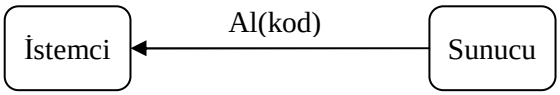
Bu paradigma gerçek hayatta şu şekilde örneklenebilir.

"Aslı çikolatalı kek yapmak istemektedir. Elinde kek tarifi olmasına rağmen, ne malzemesi ne de fırını evinde bulunmamaktadır. Arkadaşı Bilge ise evinde bu ikisine sahip bulunmakta, ancak kendisinde kek tarifi bulunmamaktadır. Aslı Bilgenin yeni tarifleri denemekten mutlu olacağını bilmekte ve telefon ederek ona "Benim için bir çikolatalı kek yapar mısın? Tarifi şu şekildedir: 3 tane yumurta al...." şeklinde yol göstermektedir. Bilge bu tarife göre keki hazırlamakta ve daha sonra bunu Aslı'ya göndermektedir."

3.4.1.3. Talepte Kod Alma

Talepte Kod Alma (Code on Demand-COD) paradigmasında, istemci kendi kaynaklarına yerel erişim hakkına sahiptir. Ancak görevi yerine getirmesi için gereken yapabilme bilgisine sahip değildir. Bu nedenle istemci bu bilgiye sahip olan

sunucudan, ihtiyaç duyduğu yapabilme bilgisine Şekil 3.5'te görüldüğü gibi erişir ve kendi görevini sonlandırır.



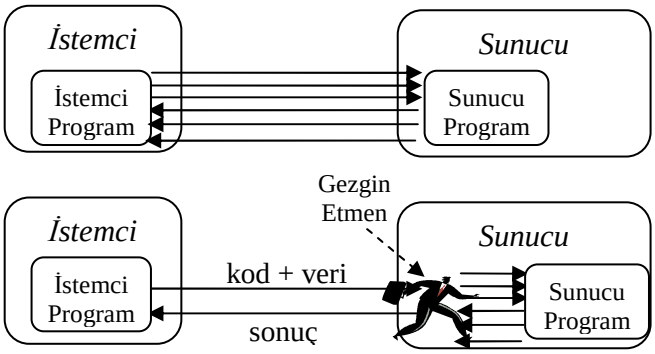
Şekil 3.5 : Talepte Kod Alma

Bu paradigma gerçek hayatta şu şekilde örneklenebilir.

"Aslı çikolatalı kek yapmak istemektedir. Evinde hem kek için gerekli malzeme hem de fırın bulunmakta, ancak kek yapmak için gerekli tarif bulunmamaktadır. Aslı bilmektedir ki, arkadaşı Bilge bu tarife sahiptir ve ihtiyaç duyulması durumunda arkadaşlarına bunu (ödünç) vermektedir. Aslı, Bilge'ye telefon açarak ondan kek tarifini ister. Bilge tarifi verir ve Aslı'da bu tarife göre çikolatalı keki hazırlar."

3.4.1.4. Gezgin Etmen

Taşınabilir kodun bir örneği olarak gezgin etmen [51] kavramı kullanılmıştır. Bir etmen, görevi ile ilgili karar alınmasında ve görevin idamesinde kullanıcı ile her zaman doğrudan bir etkileşim göstermeden farklı özerk derecelerinde çalışabilen aktif bir yazılım varlığıdır. Bir gezgin etmen ise ağ üzerinde farklı makinelerde dolaşabilen ve konak makinelerinin özkaynaklarını ve hizmetlerini Şekil 3.6'da gösterildiği gibi kullanabilen etmenlerdir.



Şekil 3.6 : Gezgin Etmen

Bu paradigma gerçek hayatta şu şekilde örneklenebilir.

"Aslı çikolatalı kek yapmak istemektedir. Elinde gerekli malzeme ve kek tarifi bulunmaktadır, ancak temini güç ve pahalı olan fırını bulunmamaktadır. Aslı bilmektedir ki arkadaşı Bilge'nin fırını vardır ve kullanım ücreti düşük olan bu fırını arkadaşlarına kullandırmaktadır. Aslı kekin hamurunu hazırlar ve bununla beraber Bilge'lere giderek çikolatalı kekini orada pişirerek kekiyle beraber tekrar evine döner."

Gezgin etmen taşınabilir, esnek, özerk, dinamik ve verimli bir yapıya sahiptir. Bir görev kodu kendi içine yerleştirildiği zaman, uzaktaki bir makineye kendi ana makinesinden gönderilir ve bu makinede görevini tamamladıktan sonra ulaştığı sonucu asıl makineye bir mesajla gönderirler veya doğrudan kendisi gelerek sonucu iletir. Gezgin etmen kavramı aynı zamanda dağıtılmış ortamlarda ve Internet'te paralel çalışmaya da olanak sağlamaktadır. Görevler ufak ve çok sayıda gezgin etmen içine gömülerek daha sonra çalışmaları için ağ üzerindeki farklı makinelere gönderilirler. Her etmen kendine yüklenen görevi yerine getirmek için bağımsız olarak çalışır ve ulaştıkları sonuç bir ana sunucu üzerinde veya yönetici etmen üzerinde toplanır. Bu sayede paralelleştirilebilen görevlerde hızlı sonuç elde edilebilmesine olanak sağlanmaktadır [46].

Bu şekildeki kompleks aktif varlıkların aşağıdaki adımları izlemeleri gerekmektedir:

1. Program akışı durdurulur.
2. Göç ettirilecek varlığı durum değişkenleri toplanır.
3. Program kodu ve durum değişkenleri gidilecek uca gönderilir.
4. Kod ve değişkenler yeni makinede eski hallerine getirilir.
5. Program çalışmaya yeniden başlatılır.

Taşınabilir sistemler aslında homojen yapıda sistemler için ortaya çıkmışlardır. Bunlarda verilerin ikili (binary) gösterimleri ve sistem özellikleri aynı olmasına rağmen yine de bazı problemlerle karşılaşılmaktadır. Günümüzdeki heterojen ortamlara doğru olan genişleme göz önüne alındığında, taşınabilirlik daha da zor bir problem olarak karşımıza çıkmaktadır. Buna cevap olarak bulunan bir yol ise Java programlama dilidir. Java nesneye yönelik programlama stilini, bir ara kod dönüşümü ile sekizli kodlarına (byte code) çevirmekte ve bu kod JVM e sahip her platformda aynı şekilde yorumlanmaktadır.

3.4.2. Gezgin Etmen Kullanmanın Faydaları

Gezgin etmen kullanımının faydaları sorulunca herkesin aklına hemen gelen bir iki özellik vardır. Danny Lange yaptığı çalışmalar neticesinde gezgin etmen kullanımı için en az yedi güzel sebep olduğunu söylemektedir [52].

- *Ağ yükünü azaltır.* Dağıtılmış sistemler genellikle verilen bir görevi tamamlamak için karşılıklı etkileşim içeren haberleşme protokollerinden faydalanırlar. Bunun sonucu olarak ağ üzerinde ciddi bir veri trafiğine sebep olmaktadırlar. Gezgin etmenler kullanıcıların bu karşılıklı konuşmalarını (protokollerini) birer paket haline getirerek etkileşimin olacağı yerel makineye gönderilmesine olanak sağlamaktadırlar. Gezgin etmenler aynı

zamanda ağ üzerinden ham veri akışını da ciddi ölçüde azaltabilmektedir. Büyük ölçeklerdeki bir veri bloğunun uzaktaki bir makinede saklanması durumunda bu verinin işlenmesi için veri bloğunun yerel makineye alınması yerine gönderilecek bir gezgin etmen ile gerekli işlemin uzak makinede, yerel olarak halledilebilmesi de mümkündür. Bu nedenle etmen tabanlı veri işlemede kullanılan slogan şöyledir:

Veriyi işleme getirmektense, işlemi veriye gönder[19]

- *Ağ gecikmelerini engeller.* Üretim süreçlerindeki robotlar gibi, kritik gerçek zamanlı sistemler çalıştıkları ortamdaki değişikliklerden hızlı şekilde haberdar olmak isterler. Bu gibi veri iletim işlemlerinin merkezi bir şekilde yapılması, yoğun veri akışlı sistemlerdeki uygulamalarda ciddi gecikmelere ve merkezde tıkanıklığa yol açmaktadır. Gezgin etmen kullanımı sayesinde merkezi kontrolden kurtularak, bu etmenler alıcıların olduğu yere yerleştirilmekte, yönetici etmenin direktiflerine göre hareket etmekte ve bu sayede gecikmeler azaltılmaktadır.
- *Protokolleri içerir.* Bir veri dağıtılmış sistem üzerindeki uçlar arasında el değiştirirken her uç giden veriyi kodlamak ve gelen veriyi yorumlamak için gerekli protokollere (yorumlayacak program parçacıklarına) sahiptir. Bununla birlikte protokoller etkin kulanım ve/veya güvenlik gibi yeni gereksinimleri karşılamak için gelişirken, uçlardaki bu program parçacıklarının da güncellenerek yeni ortama uyum sağlaması ve hantal yapıdan kurtarılması gerekmektedir. Bu da zor ve masraflı bir işlemdir. Bu nedenle protokoller, kalıtımsal problem olarak ortaya çıkmaktadır. Gezgin etmenlerin kullanımı sayesinde bu protokoller etmenin içine gömülerek sisteme dinamik ve güncel bir yapı kazandırılabilmektedir.
- *Asenkron ve özerk olarak çalışabilmektedir*. Günümüzde cep telefonu ve PDA gibi gezgin aygıtlar genellikle pahalı ve kırılgan ağ bağlantıları üzerinde çalışmaktadırlar. Sabit ağ ile, gezgin aygıt arasındaki hattın daima açık olmasını gerektiren görevlerin, bu gibi sistemlerde kullanılması teknik olarak uygun olmadığı gibi, ekonomik te değildir. Bu problemi çözmek için gerekli görev bir gezgin etmen içine gömülerek ağ üzerinden ilgili aygıta gönderilebilir. Bu dağıtım işleminden sonra etmen kendini üreten etmenlerden bağımsız, asenkron ve özerk olarak çalışabilir ve böylece bağlantının/hattın daima açık kalmasına gerek kalmaz. Gezgin aygıt daha sonra uygun olduğu bir zaman sisteme bağlanarak işleminin sonucuna ulaşabilir/gönderebilir.

[19] Move the computation to the data rather than the data to the computation.

- *Ortama dinamik olarak uyum sağlayabilir*. Gezgin etmenler, çalıştıkları ortamı algılayarak, çevrede olan değişikliklere özerk olarak cevap verebilmektedirler. Çok sayıda ve farklı tiplerdeki gezgin etmelerin ağ üzerinde uygun şekilde yayılarak belirli bir problemin çözümü sağlanabilir.
- *Doğal olarak heterojendir.* Ağ işlemleri/hesaplamaları gerek yazılımsal, gerekse donanımsal olarak bakıldığında genel olarak heterojendir. Gezgin etmenlerde genelde bilgisayar ve ulaştırma katmanından bağımsız olup (sadece çalışma ortamına bağımlı), bağlantısız sistem bütünleştirmesi için en iyi koşulları sağlamaktadır.
- *Gürbüz ve hata hoşgörülüdür.* Gezgin etmenlerin beklenmeyen durumlara karşı dinamik olarak karşılık verme yeteneklerinden ötürü gürbüz ve hata hoşgörülü dağıtılmış sistem tasarımına olanak sağlamaktadır. Bir uçtaki makinenin kapatılması gerekince bu uçta çalışan etmenler kendilerini bu kapatma sürecinde ağ üzerindeki başka bir makineye naklederek çalışmalarına orada devam edebilirler.

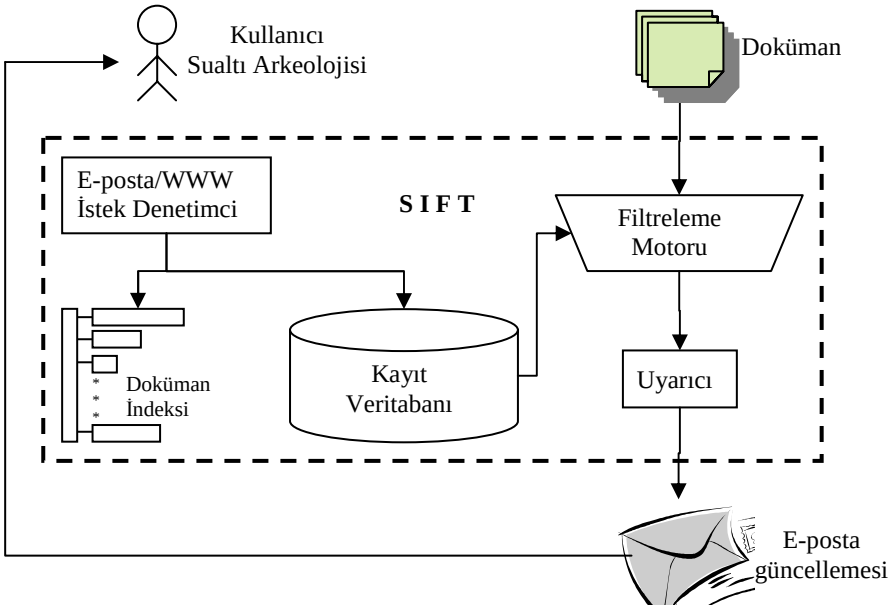
4. KONUYLA İLGİLİ ÇALIŞMALAR

İlk elektronik olay bildirim sistemi olarak seçici bilgi dağıtımı sistemi (Selective Dissemination of Information-SDI) [16] gösterilebilir. SDI bilim adamlarının, kendi uzmanlaştıkları alanlarda yapılan yayınlardan otomatik olarak haberdar edilmesi amacıyla geliştirilmiş bir sistemdir. Bir bilim adamı kendi profilini, ilgi duyduğu anahtar kelimeleri belirterek oluşturmakta ve bu profil bilgileri zamanla değişen ihtiyaçlar doğrultusunda değiştirebilmektedir. Sistem kendine ulaşan olay verilerini bu profillere göre karşılaştırarak ileteceği kişileri/adresleri belirlemekte ve buralara göndermektedir.

Günümüze kadar çok sayıda Olay Bildirim Sistemi geliştirilmiştir. Bu sistemlerden ELVIN [43], CORBA Olay Bildirim Sistemi [28], SIENA [22], Keryx [53], ve LESUBSCRIBE [25] gibileri kullanılacakları ortamdan/uygulamadan bağımsız olarak geliştirilmekle birlikte bazı olay sistemleri ise kullanılacakları uygulamaya yönelik olarak geliştirilmiştir. Örneğin Hermes [54] Sayısal Kütüphanelerde, DAOS [55] hava trafik kontrollünde, GRYPHON [24] ve REBECA [44] Elektronik Ticarette ve Genesis[56] ise seyahat planlama uygulamalarında kullanılmak üzere geliştirilmiştir. Bu sistemlerden, konu üzerinde önemli kilometre taşı olarak görülenler aşağıda detaylandırılmıştır.

4.1. SIFT

Geliştirilen ilk olay bildirim sistemlerinden olan Stanford Information Filtering Tool (SIFT) [21], geniş alanda bilgi dağıtımına olanak sağlayan doküman tabanlı bir merkezi sistemdir. Günümüzde bu sistemin daha da geliştirilmiş bir formu olan InReference [57] ticari amaçlarla da kullanılmaktadır. SIFT aslen metin dosyalarının dağıtılması hususuna yoğunlaşmış bir sistemdir. Müşteriler kendi profilleri ile merkezi sistem üzerine kayıt olarak, ilgi duyduğu dokümanın gelmesi durumunda haberdar edilmeyi beklerler. Şekil 4.1'den de görüleceği üzere SIFT'in merkezi bir yapısı vardır ve olaylar yayımcı tarafından üretilen ve sisteme gönderilen dokümanlardır. Filtreleme motoru gelen bu dokümanları kendi profil havuzuyla karşılaştırarak seçilen dokümanları ilgili müşteriye yönlendirir, diğer dokümanlar ise gözardı edilmektedir.



Şekil 4.1 : SIFT Mimarisi

Bir kullanıcı su altı arkeolojisi ile ilgilendiğini SIFT sunucusuna Internet üzerinden bir grafiksel kullanıcı ara yüzü vasıtasıyla bildirmekte, böylece sisteme kayıt olmakta ve ilgilendiği belgelerin kendine e-posta ile ulaşılmasını istemektedir. Göndermiş olduğu, kendi profilini tanımlayan kayıt mesajı, kayıt veri tabanında tutulmakta ve SIFT sunucusuna bir dosya ulaştığı zaman filtreleme motoru gerekli kontrolleri yapmakta ve uygun dosyaları kullanıcıya göndermektedir.

4.2. ELVIN

ELVIN [43, 58] dağıtılmış bir sistemdeki uygulamaları kontrol etme ve gözlemleme amacıyla Distributed Systems and Technology Centre (DSTC), Queensland, Avustralya tarafından geliştirilmiştir. Sistemin geliştirilmesi sürekli devam etmekte olup günümüzde en son hali ELVIN4 olarak anılmakta ve dağıtılmış istemci sunucu modelini kullanmaktadır. Olay verilerin gözlenmesi sistemin bir görevi olmayıp gelen olay verilerinin yayımı yapılmakta olan ELVIN sisteminde, olay verileri olarak basit olay verilerini kullanılmaktadır.

Bir ELVIN bildirimi isim-değer ikilisinden oluşmaktadır. Temel ilkeller olarak 32 ve 64 bitlik sayısal değerler, 64 bitlik reel sayılar, sekizli (byte) dizileri ve katarlar kullanılmaktadır. Kayıt ifadeleri C benzeri koşul ifadelerine benzemektedir. “stock == ‘abc’ && value > 80” gibi tanımlanan deyimler Lukasiewicz’nin üç durumlu mantığı (doğru, yanlış, belirsiz) ile değerlendirilmektedir.

Evlin sistemi C, C++, Emacs Lisp, Java, Python, Smalltalk, and Tcl dillerini desteklemekte olup içerik tabanlı bir yapıya sahiptir. ELVIN sistemi ayrı bir

güvenlik modeli uygulamakta olup bu model ile kayıtçılar ve yayımcılar kendi aralarındaki haberleşmeyi anahtarlar ile şifrelemektedir.

ELVIN 4.1 19 Mart 2003'te tamamlanmış olup bu sürümü web tabanlı sunucu yönlendirmesi, düzenlenebilir servis kalitesi, sunucuların çökmesine karşı otomatik hata kurtarma ve yeni ölçeklenebilirlik destekleri içermektedir

4.3. Yeast/Ready/OmniNotify.

Merkezi olarak çalışmak üzere geliştirilen Yeast [15] (Event Action) ve onun bir devamı olarak dağıtılmış ortamlar için geliştirilen (*REliable, Available, Distributed Yeast*) Ready [59] bildirim sistemi AT&T tarafından geliştirilmiştir. Ready, SIENA'ya benzer yapıya sahip dağıtılmış bir sistem olup farklı hiyerarşilerde birleştirilmiş bir kısım olay sunucunun birleşmesiyle oluşturulmuş bir sistemdir. Basit ve bileşik olayların yakalanmasına olanak sağlayan dağıtım mekanizmasının yanı sıra, kullanıcı profilleri üzerinde gruplama yapması sistemin ayırt edici özelliklerindendir. Olay tanımlamasında "{event_var[j..k] : event_type | expression }" yapısına uygun olarak olay tanımlamaları yapılmaktadır. Örnek bir olay mesajı ise şu şekilde gösterilebilir, "*Ev1: MyAlarm | $Priority > 5"*.

Sistem CORBA'nın bildirim standartları desteklemek için yeniden yapılandırılmış olup yeni hali OmniNotify [60] olarak adlandırılmıştır. OmniNotify çok görevcikli olarak çalışan CORBA Bildirim Servisini kullanan, dağıtılmış uygulamalar için olay tabanlı ve asenkron haberleşme mekanizması sağlayan bir sistem olup, C++ ile gerçeklenmiştir. Sistem içinde ölçeklenebilir bir filtreleme mekanizması ve olay dağıtım sistemi bulunmakta olup, İtme ve Çekme modelli olay üreticileri ve tüketicileri desteklemektedir.

4.4. Java AWT: Delegasyon Olay Modeli.

Java programlama dili, tek bilgisayarda olan çalışmasında olay modeli olarak delegasyon (delegation) olay modelini [35] kullanmaktadır. Delegasyon olay modeli özellikle grafiksel kullanıcı arabirimi (GUI) merkezli uygulamalarda olay haberleşmelerini tek bir JVM'de merkezi olarak halletmekte iken, Java nın dağıtılmış olay modelinde [61] gerek sanal anlamda gerekse fiziksel anlamda uzak olarak yerleştirilmiş makinelerdeki nesneler arasındaki olay haberleşmesine olanak sağlamaktadır.

Java uygulamalarında grafiksel kullanıcı arabirimlerinin tasarlanmasında gerekli olan sınıflar Abstract Window Toolkit (AWT) kütüphanesi altında toplanmıştır. Delegasyon olay modeli de ilk kez Java 1.1 in AWT kütüphanesinde tanımlanarak,

GUI'lerde ortaya çıkan olayların yakalanması içen ve bu olayların neticesinde programcı tarafından tanımlanan kodların çalıştırılması amacıyla kullanılmaktadır.

Delegasyon olay modeli *merkezi* bir olay modelidir. Yayımcılar, *olay kaynakları* ve kayıtçılar da *olay dinleyicileri* olarak ifade edilirler. Bir olay kaynağı genellikle bir grafiksel kullanıcı arabirimi bileşenidir (button, menu item, list .. vb). Java uygulamaları basit bir olay tabanlı haberleşmeyi JDK da tanımlı iki ara yüz kullanımı ile de bahsedilen şekilde gerçekleyebilirler. Olay tabanlı bir haberleşme kurabilmek için dinleyiciler ilgilendikleri belirli bir olay türü üzerinden kayıt olurlar.

4.5. CORBA Olay Servisi/CORBA Bildirim Servisi

CORBA Olay Servisi [36] uygulama bileşenlerinin olayların üzerinden haberleşmelerine olanak sağlamaktadır. Mevcut sistem ile ilgili gerek filtreleme, gerekse servis kaliteleri (QoS) ile ilgili 1996 yılında öneri sunulmuş olup, 1998 yılının sonunda yeni güncellemeler yapılarak CORBA Bildirim Servisi olarak kabul edilmiştir.

CORBA Olay Servisi, veri alma modeli olarak hem itme modelini hem de çekme modelini desteklemektedir. Tüketici nesneler genellikle çekme modelini kullansalar da gerçekte iki farklı mekanizmayı da kullanabilirler. Çekme modelinde iki farklı mekanizma kullanılır (pull, try_pull). Bloke eden birinci mekanizmada, tüketici nesne, olay verisi çekme talebini *pull* komutuyla ilettikten sonra olay verisi gelene kadar beklerken, tüketici nesneyi bloke etmeyen ikinci mekanizmada *try_pull* komutuyla olay verisi alma talebinde bulunan nesne bloke olmayıp kendi normal çalışmasına devam etmektedir.

Olay kanalı üretici ve tüketici nesnelerin arasına girerek, aralarındaki haberleşmeyi bölmeleyerek çok sayıda üreticinin ve çok sayıda tüketicinin anonim olarak (birbirlerinin adreslerini bilmeden), haberleşmelerine olanak sağlamaktadır. Üretici, tüketici ile karma bir veri alma modeli vasıtasıyla da haberleşebilirler. Örneğin, bazı tüketiciler çekme modelini kullanırken, bazı tüketiciler itme modelini tercih edebilir. Grup yönetim ara yüzleri olan *Üretici Yöneticisi* ve *Tüketici Yöneticisi* ara yüzleri birer fabrika nesne[20] olarak görev yaparak, yeni üreticiler ve tüketiciler eklenmesine olanak sağlamaktadırlar. Olay kanalında aynı zamanda filtreleme kapasitesinin yanı sıra, servis kalitelerinden olan *olay önceliği* ve *olay mesajının ulaşım garantisi* servisleri de sağlanabilir. Bunlar olay servisi tanımlamalarında olmayan, sistemi kullanacak ve geliştirecek olan kullanıcıların geliştirmesi gereken servislerdir.

[20] Fabrika nesne; başka nesneleri oluşturabilen nesnelerdir.

Sistem farklı olay kanallarının birleşmelerine olanak sağlamaktadır. Bu sayede bir kanalda üretilen bir olay verisi başka bir kanaldaki tüketiciye ulaştırılabilmektedir. Bu şekilde olan birleştirmelerde gecikme zamanının arttığı, ortak çalışımda bazı sınırlamalar ortaya çıktığı görülmüştür. CORBA olay servisi tanımlamaları, olay kanalını bulmaları için bir plan tanımlamamışlardır. Bunun yerine bir adlandırma servisinin kullanılması veya yerel makinede saklanmış bir dosyadan veya tablodan ilgili kanalın adresinin alınması önerilmiştir.

1996 yılında OMG CORBA Bildirim Sistemini tanımlayan bir öneri teklif etmiştir. CORBA Bildirim Sistemi, bir önceki bölümde tanımlanmış olan CORBA Olay Servisine [36] filtreleme yeni kapasiteler ve bazı servis kaliteleri eklemiştir (Quality of Services-QoS). Sistemin en son hali tekrar gözden geçirilerek tekrar yayınlanmış 1998 yılı sonunda OMG tarafından kabul edilmiştir [62]. CORBA Bildirim Sistemini bildirim kanalının bu simetrik yapısı yönetim, filtreleme, servis kaliteleri ve yapısal olay haberleşmesi gibi kavramları da desteklemektedir. Bildirim kanalının oluşturulması aşamasında kendi bellek birimleri ilgili bazı yönetimsel özelliklere sahip olması sağlanabilmektedir. Bu özellikler sayesinde bağlı olabilecek en fazla sayıdaki üretici ve tüketicilerin tanımlanması yapılabilmekte, olay kuyruğunda bekleyebilecek olay miktarının üst sınırı tanımlanabilmektedir.

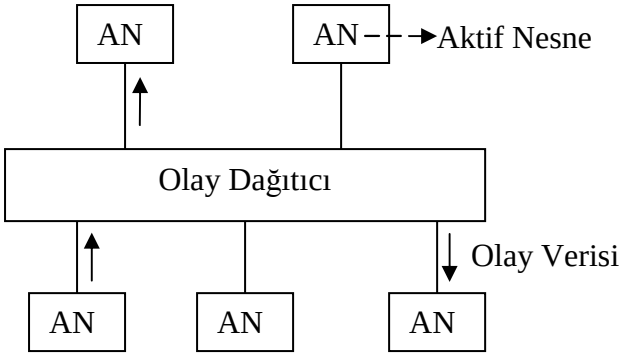
CORBA Bildirim Sistemi kendi başına bir olay bildirim sistemi olmasıyla birlikte, aynı zamanda olay bildirim sistemi geliştirilmesinde de bir araç olarak kullanılmıştır. Örneğin OmniNotify [60] ve COBEA [63] sistemleri CORBA Bildirim Sistemi tabanlı olarak geliştirilmiş olay bildirim sistemleridir.

4.6. JEDI

Java Event Based Distributed Infrastructure (JEDI) [38, 39], olay tabanlı nesneye yönelik bir uygulama altyapısı geliştirme amacıyla CEFRIEL- Politecnico di Milano tarafından Java programlama dili ile geliştirilmiştir. Şekil 4.2'de altyapısı gösterilen JEDI, iş akışı yönetim sistemlerinin geliştirilmesinde kullanılmıştır. JEDI olay sisteminde, *aktif nesne* kavramı üzerine kurulmuştur. Aktif nesneler uygulama platformuna özel görevleri olan, kayıtçı ve/veya yayımcı olabilecek özerk varlıklardır. Sistemde gönderilen olay verisinin alıcıya (alıcılara) ulaştırılması olay dağıtıcının görevidir. Olay dağıtıcısı, gönderilen olay verilerinin dağıtımının yanı sıra, aktif nesnelerin, *"kayıt_olma"* ve *"kayıt_silme"* işlemlerini de yerine getirmektedir.

JEDI'de olay verileri sıralı katar dizileri olarak tanımlanmışlardır. İlk katar olayın ismini diğer katarlar ise olay parametrelerini belirtir. Bu yapı geleneksel programlama yapısındaki fonksiyon çağrısına benzer bir şekildedir.

print(tez.doc, myprinter) // olayın ismi(parametreler)



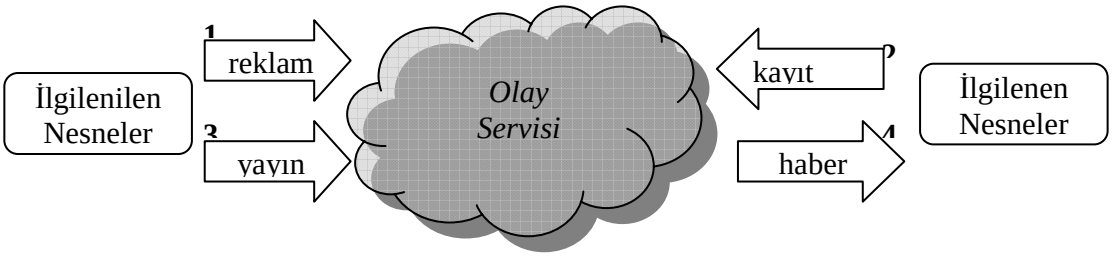
Şekil 4.2 : JEDI Olay Sistemi

JEDI olay sisteminde bir aktif nesne ya belirli bir olay üzerine yada bir olay düzeni (event pattern), üzerine kayıt olabilmektedir. Olay düzeni bir ifadeyi gösteren düzenli bir katar kümesidir. Aynı olay verisinde olduğu gibi, ilk katar olay düzeninin adını verirken, diğer katarlar ise parametreleri göstermektedir. Her katar bir yıldız işareti (asteriks) ile bitebilir. Örneğin "foo(aa*,bb)" olay düzeni foo isimli olay verileri arasında ilk parametresi "aa" ile başlayan ve ikinci parametresi "bb" ye eşit olanları ifade etmektedir. "*(*,*,*) olay düzeni ise üç parametresi olan bütün olay verilerini ifade etmektedir.

4.7. SIENA Olay Sistemi

SIENA (Scalable Internet Event Notification Architecture) [19] olay modeli Politecnico di Milano tarafından geliştirilmiştir. Temel olarak aynı kurum tarafından daha önce geliştirilen JEDI [38] yi temel almış ancak fonksiyonelliği daha da artırılmıştır. JEDI'den farklı olarak SIENA özellikle Internet gibi geniş alan ağlarındaki olay tabanlı haberleşmeyi amaçlayan içerik tabanlı bir kayıt/yayım sistemidir.

SIENA olay sistemi dağıtılmış yapı üzerindeki olay sunucularının oluşturduğu, ölçeklenebilir, olay filtrelerinin kolay yapılabildiği, içerik tabanlı kayıt/yayım sistemi modeline uygun açık kaynaklı (kaynak kodlara ulaşılabilir) bir olay sistemidir. SIENA'nın terminolojisinde yayımcılar *İlgilenilen Nesne (Object of Interest)* kayıtçılar ise *İlgilenen Nesne (Interested Party)* olarak adlandırılmışlardır. Olayların dağıtılmasında ise üç temel ilkel (duyuru, kayıt ve yayım) kullanılmıştır.



Şekil 4.3 : SIENA Olay Modelinin Yapısı

SIENA olay sisteminin temel yapısı Şekil 4.3'te gösterilmektedir. Yapıda da görüleceği üzere *kayıt* ve *yayım* ilkellerinin kullanımı bir çok olay modelinde de standart olmakla birlikte *duyuru (advertise)* ilkeli, modeli ise SIENA'ya özeldir. Bu ilkel sayesinde *İlgilenilen Nesne* üreteceği olay tipini ilan ederek (duyurarak) bu olaylara talep eden istemcilerin kayıt olmalarını sağlar, daha sonrada bu olayları üreterek *İlgilenen Nesnelere* ulaşması için olay servisine yayım yapar. Böylece olay sunucuları yayımı yapılmayacak özellikteki olay verilerinin üzerine kayıt kabul etmezler. Benzer şekilde *duyuru_sil (unadvertise)* ilkeli ile de bahsi geçen olay verisinden artık yayım yapılacağı beyan edilerek olay sunucularının buna göre kayıt kabul etmelerinin (veya etmemelerini) bildirir. Bu üç ana mekanizmanın ara yüzleri şu şekilde tanımlanmıştır.

```
Subscribe(Ilgilenen Nesne, Şablon)
Unsubscribe(Ilgilenen Nesne, Şablon)
Publish(olay)
Advertise(Ilgilenilen Nesne, Filtre)
Unadvertise(Ilgilenilen Nesne, Filtre)
```

SIENA da ilgilenilen nesnelerin, ilgilenen nesnelerin ve olay sunucularının tanımlanması için Uniform Resource Identification (URI) adlandırma şeması kullanılmıştır. Bu sayede her nesnenin kendisiyle alakalı olarak "mailto:ozgur@hho.edu.tr" gibi URI ismi olacaktır.

SIENA olay sistemi tipli olay veri modelini desteklemektedir. Olay verisinde her nitelik bir "isim,tip ve değer" şeklinde bir üçlü oluşturduğu form yapısında ifade edilmektedir. Nitelikler tekil olarak tanımlanmış bir isme sahiptir. Örnek bir olay verisi yapısı Şekil 4.4.a'da gösterilmiştir.

string olay = hesap/borc time tarih = 15.11.2002 int rakam = 12345 float miktar = 215.31	string olay == hesap/* time tarih >= 01.01.2000 float miktar > 100.000	String event > finans/hisse/ string symbol = SAHOL float change < 0 and then string event > finans/hisse/ string symbol = DOHOL float change > 0
a. Olay Verisi örneği	b. Filtre örneği	c. Bileşik Olay Örneği

Şekil 4.4 : SIENA Olay Siteminde Olay Verisi ve Filtre Tanımlaması

Bu şekilde ifade edilen olay verilerinde kayıt için tanımlanan ölçütler (filtreler), olay verisindeki her nitelik için ayrı ayrı ifade edilebilmektedir. Kullanılan niteliksel olay verisinin kullanımı sayesinde, olayların nitelikleri ile ilgili kısıtlamaları yapmak, ve olay filtrelemek kolay bir hal almıştır. Örnek bir olay filtresi Şekil 4.4.b'de gösterilmiştir. Örnekteki olay filtresinde, 2000 yılından itibaren yapılan hesap hareketleri içinde miktarı yüz binden fazla olanlarla ilgili bir filtreleme yapılabilmektedir. SIENA aynı zamanda bileşik olay algılama yapısı da geliştirmiş ve buna uygun kayıt mekanizmalarına işlerlik kazandırmıştır. Örneğin bir kayıtçı Şekil 4.4.c'de görülen bir bileşik olay üzerine kayıt olarak SAHOL hisselerinde bir azalma olduktan sonra DOHOL hisselerinde olacak bir artıştan haberdar edilmek istediğini bildirmektedir.

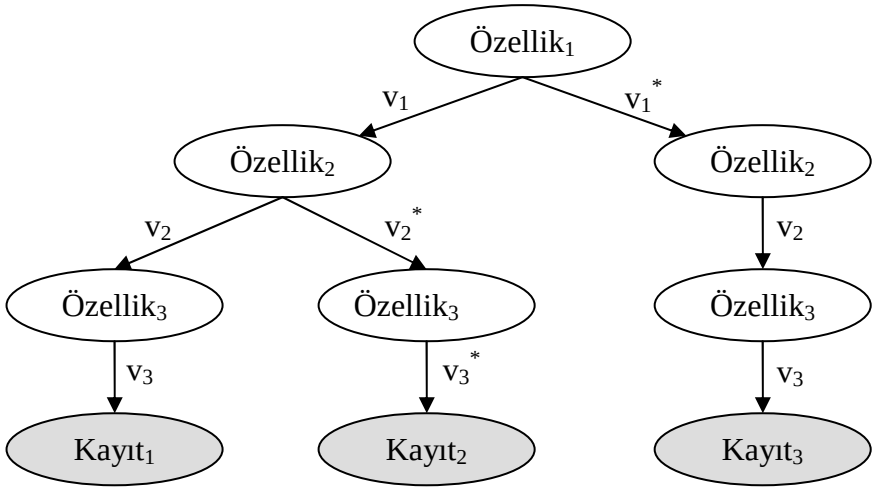
4.8. GRYPHON

GRYPHON [24, 59], IBM tarafından geliştirilmiş olan dağıtılmış içerik tabanlı bir mesaj dağıtım sistemidir. Her ne kadar GRYPHON'un yapısı SIENA'nın yapısına çok benzese de filtreleme tekniği SIENA'da kullanılan modelin tamamlayıcısı olarak görülmektedir. GRYPHON, büyük boyuttaki bir profil kümesi içinde eşleştirilen bir bildirimi oluşturabilmek için dağıtılmış bir filtreleme mekanizması kullanmaktadır. SIENA ile GRYPHON'un temel farkı GRYPHON sisteme gönderilen tüm profilleri sistem üzerinde yayımlarken, SIENA ise daha genel olan profilin sistem üzerinde dağıtılmasına olanak sağlamaktadır (Kapsama İlişkisi yüzünden).

GRYPHON büyük boyutlu verilerin, gerçek zamanlı olarak binlerce müşterisi/kullanıcısı olan bir genel ağda dağıtımını sağlamakta olup sistem içerik tabanlı kayıt/yayım sistemi destekleyen, istemci-sunucu modelinde bir olay sistemidir. Olaylar daha önceden oluşturulmuş bir *karşılaştırma ağacına* göre dağıtılırlar. Şekil 4.5'te görüldüğü gibi bir *karşılaştırma ağacı*

- özellikler üzerine tanımlanmış olan deyimleri içeren **uçlardan,**
- ve bu deyimin değerlendirilmesi sonucu geçilen **kenarlardan** oluşur.

Ağacın her alt seviyesi, deyimlerin değerlendirilmesi/arıtılması sonucu ulaşılmış olup, ağacın yapraklarında ise tekil kayıtlar vardır. Hata hoşgörüsü için sunucu bağlantılarında hücresel yapı kullanılmış olup dağıtım sistemi her biri kendi içinde tam bağlı hücrelerden oluşan alt ağlar olarak, birbirleri ile bağlanmışlardır.



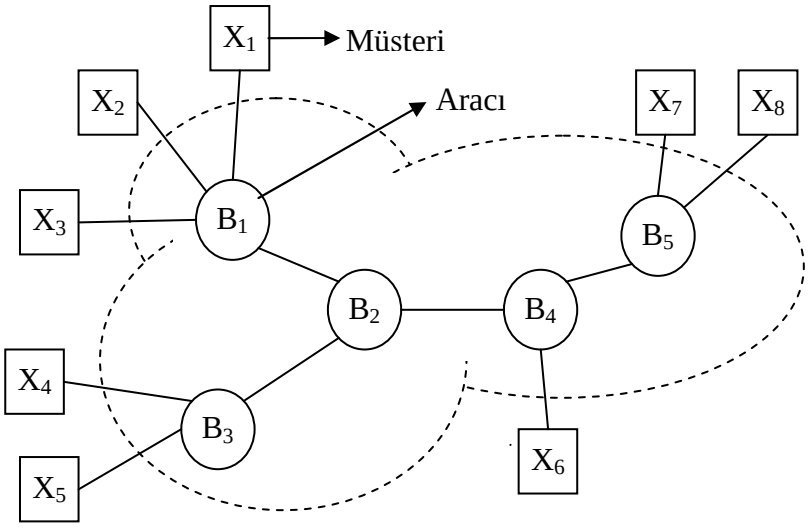
Şekil 4.5 : GRYPHONda Olayların Karşılaştırma ağacından filtrelenmesi

Şekil 4.5'ten de görüleceği üzere sisteme yapılan kayıtlar bir ağaç yapısı üzerinde biçimlendirilmiş olup bu kayıt mesajlarına bir olay verisinin gönderilmesi için aşağıdaki koşulların sağlanması gerekmektedir.

$Kayıt_1 := (Özellik_1 = v_1)$ ve $(Özellik_2 = v_2)$ ve $(Özellik_3 = v_3)$
$Kayıt_2 := (Özellik_1 = v_1)$ ve $(Özellik_2 = v_2^*)$ ve $(Özellik_3 = v_3^*)$
$Kayıt_3 := (Özellik_1 = v_1^*)$ ve $(Özellik_2 = v_2)$ ve $(Özellik_3 = v_3)$

4.9. REBECA

REBECA [44, 60], Darmstadt Teknik Üniversitesinde, olay tabanlı elektronik ticaret sistemi oluşturmak için geliştirilmiş bir projedir. Proje dağıtılmış ortamda olay filtreleme üzerine odaklanmaktadır. Kullanıcı profilleri sistem üzerinde bazı en iyileme teknikleri kullanılarak (yinelemeli profillerin dağıtımlarının engellenmesi gibi) dağıtımı yapılmakta olup, olay mesajlarının bu profillerden oluşturulan olay filtrelerine göre dağıtımı yapılmaktadır. Yapısına bakıldığında REBECA, SIENA'da uygulanan yaklaşımın bir uzantısı olarak görülebilir. Günümüzde borsa veri akış işlemlerinde ve açık artırma uygulamalarında da kullanılan prototipi geliştirilmiş olup heterojen ortamları da desteklemektedir.



Şekil 4.6 : REBECA Sisteminin Topolojisi

Şekil 4.6'da görüldüğü gibi, REBECA sistemi çevrimsel olmayan ve her aracı ile bağlantısı bulunan bir topoloji içindeki aracıların kurduğu dağıtım sistemi üzerinde çalışmaktadır. Olay verilerinin dağıtılması için tek bir ağaç yapısı bulunmakta olup, merkezi bir aracının çökmesi durumunda sistemim büyük bölümü çalışmamaktadır.

4.10. LESUBSCRIBE

LESUBSCRIBE [64, 25], Caraval projesi kapsamında Inria/Paris tarafından geliştirilmiş olan içerik tabanlı bir olay sistemidir. Web üzerinde veri yayımlamaya uygun yarı yapısal bir olay modeli kullanarak yayımcılarla kolaylıkla bütünleşebilen esnek bir yapı geliştirmeye çalışılmıştır. Olay bir (özellik, değer) ikililerinden oluşmaktadır.

Örnek: "{(category, book), (title, Hamlet), (price, $20), (author, Sheakespeare)}"

Sıklıkla kullanılan, olay bildirim sorgulamalarını destekleyen bir kayıt dili kullanılan sistemde kayıt mesajları ise basit ilkellerin birleştirilmesinden oluşmaktadır.

Örnek: "(category = book) & (title = Hamlet)

Etkili bir karşılaştırma algoritması kullanarak gerçek zamanda çok sayıda kayıtlı müşteri profili içinden olay ile ilgilenenlerin en kısa sürede seçilmesi amaçlanmakta olup sistem içinde birkaç algoritma birden kullanılmaktadır. Gerek HTTP, gerekse Java RMI üzerinden kolaylıkla sisteme kayıt olmak ve olay yayımlamak için uygun ara yüzleri bulunmaktadır.

Internet üzerinde çalışan örneği merkezi bir sistem olup, açık artırma sistemi gibi aktif ve pasif olay gözlemeye olanak sağlamaktadır. ELVIN sisteminde olduğu gibi sadece basit olay verileri desteklenmektedir. Sistemin filtreleme mekanizmasında

sürdürülebilirlilik ve performans ölçütleri arasında bir denge kurmayı amaçlamıştır. XML işlemeye olanak sağlayan WebFilter [65] prototipi geliştirilmiştir.

4.11. ECho/JECho

ECho[66], Georgian Tech tarafından geliştirilmiş ve olay kanalları üzerine kurulmuş olan bir yüksek performanslı veri transfer mekanizmasıdır. Aynı CORBA Olay Servisinde olduğu gibi kanal tabanlı kayıt mekanizmasını desteklemektedir. Özellikle sürekli akış gösteren verilenin dağıtımına uygun bir alt yapısı bulunmaktadır. Bu sistemin daha geliştirilmiş hali olan JECho [67] olay kanalları üzerinde gezginlik desteği sağlanmaktadır. Temel problem ise dinamik olarak veri dağıtım ağacının kurulması olup kullanılan modülatörler sayesinde filtreleme işlemleri zorlaşmakta ve üretilen olayların hepsi tüketiciye ulaştırılmaktadır. Gezgin bir üretici bağlı bulunduğu sunucu üzerindeki tüm modülatörleri indirmek durumunda kalmaktadır. Sistem gezginlik desteği sağlayan dağıtılmış olay sistemlerinde yük dengeleme sağlaması ve kaynak görüntüleme işlemleri yapmasından dolayı yeni bir yaklaşım getirmiştir.

4.12. STEAM

STEAM (Scalable Timed Events and Mobility) kablosuz ad-hoc ağlar için tasarlanmış ve özellikle trafik yönetimini amaçlamış olan bir olay sistemidir [68]. Sistem dinamik olarak ağ yapısının yeniden yapılandırılmasından dolayı üç farklı filtre kullanmaktadır. Bunlar *konu, yakınlık ve içerik* filtrelemeleri olarak gruplandırılmıştır. Olay üreticiler üretecekleri olay tiplerini belirli bir coğrafi alan içinde sisteme yayınlamakta ve bu alan yakınlık olarak adlandırılmaktadır. Yakınlık işlemi sadece fiziksel mesafeyi ifade etmemekte olup bir grup mekanizması olarakta düşünülmektedir. STEAM , Proximity-based Group Communication Service (PGCS) yapısını kullanarak bir grup haberleşmesini desteklemektedir. Bir düğüm bu gruba girmek için o grubun coğrafi alanında bulunması yani gerekli kriterleri sağlaması gerekmektedir.

4.13. DADI

DADI (Discovery, Analysis and Dissemination of Information) [69-71] Princeton üniversitesinde geliştirilen bir araştırma projesidir. Sistem bilginin Internet üzerinde keşfedilmesi, analizi ve dağıtımı işlemlerini yerine getirmektedir. Olay tabanlı haberleşmeyi geniş alan ağı üzerinde kayıt/yayım haberleşme modeli ile desteklemekte olup sistem, *algoritma ve uygulama* katmanları gibi farklı

katmanlardaki işlemleri yerine getiren alt projelerden oluşmaktadır. , Sistem katmanı konu tabanlı kayıt yayım modeli için gerekli sistem altyapısını sağlamakta olup algoritma katmanı içerik tabanlı yönlendirme ve karşılaştırma işlemlerini gerçeklemektedir. Uygulama katmanı Internet gibi geniş bir ağ üzerinde kalıcı arama arama yapmaya olanak sağlamaktadır.

4.14. Hermes

Hermes [72, 73] bir eşler arası (peer to peer) olay sistemi olup Pan olarak adlandırılan farklı bir model duyuru mekanizması üzerine inşa edilmiştir. Sistemin mesaj yönlendirme algoritması, geliştirilmiş hata hoşgörü mekanizması ve ölçeklenebilir yapısıyla diğer sistemlere üstünlükleri bulunmaktadır. Hermes basit kayıt yayım işlevlerini desteklemekte olup kullandığı *randevu noktaları* ile de duyuru ve kayıt mesajlarının dağıtımını koordine etmektedir. Kullandığı tip tabanlı yönlendirme modeli sayesinde üretilen kayıt mesajları duyuru mesajlarının aksi yönde hareket ederek dağıtım ağaçlarını belirlemektedir. Diğer duyuru mekanizmasını kullanan sistemler olan Siena ve Rebeca'dan temel farkı ise yönlendirme işlemlerinde öncelikli amacın randevu noktalarına doğru yönlendirme olmasıdır.

4.15. Fuego Olay Servisi

Fuego olay servisi Fuego Core tarafından 2002-2004 yılları arasında Helsinki Institute for Information Technology (HIIT)'da bir proje olarak geliştirilmiştir. Sistem gezgin hesaplama ortamlarını, asenkron-içerik tabanlı kayıt yayım sistemi altyapısını kullanarak desteklemekte ve bu sayede tüketicilerin gezginliğine olanak sağlamaktadır [74]. Sistemin temel elemanı olay yönlendiricidir. Olay yönlendirici üretilen olay verilerinin üretici ile tüketici arasında hızlı şekilde gönderilmesini sağlamaktadır. Fuego altyapısı Java API'leri ile Host Identity Protocolünü (HIP) desteklemekte olup HIP özelliklerini bulunduran uygulamaların kullanılmasına olanak sağlamaktadır [75].

4.16. OBVENT

Dağıtılmış olay sistemlerinde, olay verilerinin niteliği artırılarak, sistemde daha önemli bir varlık olarak algılanmasının amaçlandığı OBVENT (**Ob**ject E**vent**) sisteminde, olaylar birer nesne olarak tanımlanmakta ve sisteme kayıt olacak sistem varlıklarının bu olay tipleri üzerinden kayıt olmalarına olanak sağlanmaktadır.

Sistemin geliştirilmesinde kendi modeline uygun, Java programlama dili ile kolaylıkla uyumunun sağlanabileceği, özel bir dil yapısı geliştirilmiş olup, geliştirilen sistem varlıklarının bir önderleyiciden derlenmesi gerekmektedir. Bu önderleyici Java RMI in derleyicisinin benzeri bir yapıda olup sistemin etkin çalışması için gerekli uyarlayıcıları[21] oluşturmaktadır.

Sistem olay verisi olarak nesneleri kullanmasından dolayı kullanılan bu nesnelerin bazı özellikleri olması gerekmektedir. Bu özelliklerden en kritik olanı nesnenin bağlantısız olmasıdır. Yani o nesnenin bulunduğu bilgisayara bağımlı değerleri, başka nesne bağlantıları olmamalı, farklı bilgisayarlar arasında dolaştığında bundan dolayı değerlerden hata çıkmamalıdır.

Sistem bünyesinde kayıt/yayım modelini gerçeklemek için iki temel ilkel geliştirilmiştir. Bunlardan birincisi yayım işlemini sağlayan *publish* ilkelidir. Bu ilkelin kullanımı

```
publish o;      // "o" nesnesini yayımla (olay sistemine gönder)
```

şeklindedir. Basit bir sentaksı olan bu ilkel önceden belirlenen kurallar çerçevesinde "o" nesnesin bir kopyasının sisteme kayıtlı bütün kayıtçılara ulaşmasını sağlayacak işlem dizisinin tetiklenmesini sağlar. Bu hali ile *publish* ilkeli, Java'da kullanılan *new* ilkelinin dağıtılmış ortamda nesne yaratmaya olanak sağlayan bir varyantı olarak görülebilir. *New* ilkeli ile yerel makinede yeni bir nesne oluştururken, *publish* ilkeli sayesinde uzaktaki makinelerde yeni nesneler oluşturulabilmektedir.

Kayıt işleminde temel ölçüt ilgi duyulan olay nesnesinin tipidir. Kayıt "subscribe" ilkeli kullanılarak bu tip üzerine yapılır. Kayıt olurken aynı zamanda aranan özellikler bir yüklem yapısında, başka bir deyişle bir filtreleme yapısında gösterilebilir. Bunun için bir **T** tipindeki **t** nesnesine kayıt olmak için kullanılan *subscribe* komut dizini şu şekildedir.

```
Subscription s = subscribe (T t) {...Blok 1...} {...Blok 2….};
```

Birinci blokta, ilgilenilen olay nesnesinden hangi şartlar oluştuğu takdirde mesaj alınmasının beklendiği, bir önerme veya filtreler ile belirlenir (sonucu "boolean" olarak dönecek bir metot çağrısı). İkinci blokta ise olay nesnesinin ulaşması ve filtrelemeyle ilgili ölçütler de başarılı şekilde neticelenmesi durumunda yürütülecek işlemler dizisi ifade edilir.

Bu ilkellerin somut bir uygulama üzerinde kullanımını şu örnek üzerinden incelenebilir. Bir borsa olay servisini göz önüne alındığında yayımı yapılacak olan bir Hisse_Fiyat nesnesi SAHOL hisse senedinden 4500 TL fiyat ile 10 Lotluk işlem

[21] adapter

yapıldığının gösterildiği bir olay nesnesi oluşturulmuş ve daha sonra da yayım işlemi yapılmış olsun;

```
Hisse_Fiyat q = new Hisse_Fiyat ("SAHOL", 4500, 10);
publish q;
```

5000 TL fiyatın altındaki Sabancı Holding hisse senetleri ile ilgilenen bir kayıtçıda ilgi duyduğu kaydı ve ölçütleri de şu şekilde belitmiş olsun

```
Subscription s = subscribe (Hisse_Fiyat q)
{ return (q.getPrice() < 5000 && q.getCompany().indexOf("SAHOL") != -1);
  }
{  System.out.print("Hisse Al: ");
   System.out.println(q.getPrice());       };
```

Sistemin “Publish” komutu yürütüldüğü zaman, yukarıda belirtilen şekilde kayıt olan bir kayıtçıya olay nesnesinin ulaştırması gerekmektedir.

Java dilinde bir çözüm getiren tip tabanlı olay haberleşme modelinin çalıştırılabilmesi için, geliştirilen kodun bir ön derleyiciden geçirilerek sistemin çalışması için gerekli uyarlayıcıların oluşturulması gerekmektedir. Bu önderleme işleminin sonunda oluşan sınıf yapıları kullanılarak etkili olarak çalışabilen bir sistem geliştirimi yapılmıştır.

5. ABDES SİSTEMİ

ABDES Sistemi, kayıt/yayım haberleşmeli dağıtılmış olay sistemlerinin ve gezgin etmenlerin özelliklerini birleştirerek geliştirilmiş bir Etmen Tabanlı Dağıtılmış Olay Sistemidir. Günümüze kadar geliştirilmiş olan ve Bölüm 4'te detaylı olarak anlatılan dağıtılmış olay sistemlerinin hemen hepsi, olay kavramını *"bir veri kaynağı / veri üreticisi tarafından üretilen basit formatta bir veri"* olarak algılamışlar ve çalışmalarını üretilen bu verinin, uygun formatta gösterilimi, işlenmesi ve hızlı şekilde dağıtılması üzerine yoğunlaştırmışlardır. Bu olay sistemlerinden sadece Tip Tabanlı Olay Sisteminde (Obvent Sisteminde) olay verisi bir nesne olarak kabul edilmiş olup daha da nitelikli bir veri haline getirilmiş olmasına rağmen, olay verisinin sadece yapısı değiştirilip, kendi karar mekanizması/özerkliği olmayan bir nesne formunda kullanılmıştır.

Bu tez çalışmasının gelişimi sırasında dağıtılmış olay sistemleri basit formatta geliştirilmeye başlanmış olup öncelikle merkezi yapıya sahip olan kural tabanli bir dağıtılmış olay sistemi olan RUBCES [76] geliştirilmiştir. Sistem geliştiriliminde karşılaşılan zorluklar çözümlendikten sonra olay sistemi kural tabanlı kayıt mekanizması korunarak dağıtılmış bir yapıya kavuşturulmuş ve yeni geliştirilen sistem RUBDES [77] olarak adlandırılmıştır.

Bu tez çalışmanın amacı ise olay kavramının üretim niteliğini geliştirerek, alışılagelen *sadece ham veri üretiminin* yanı sıra, eylemlerin (action) de üretildiği ileri bir düzeye yerleştirmektir. Bu düzeydeki bir olay, işlenebilir veriler üretebileceği gibi, kendine özgü bir kontrol akışına sahip olan bir eylemler dizisinin tetikleyicisi de olabilir. Söz konusu eylemler belirli bir amacı gerçeklemeye yönelik işlem dizisine, karar mekanizmalarına ve çoğunlukla birlikte üretildikleri bir veri kümesine sahip olacaklardır.

ABDES Sisteminde olay üreticiler, basit olay verisi yerine gezgin ve zeki bir etmenle (agvent) ifade edilen eylemler üretirler. Her agventin iki farkı davranış kümesi bulunur:

1. *Yaratılmasına neden olan olayın kendine yüklediği amaç ve davranışlar.* Agvent, üreticisinin belirlediği kontrol akışı içinde amacını gerçekleştirmek üzere etkin olur. Amaç, en basit durumda, olayın ürettiği ham verilerin tüketicilere ulaştırılması olabileceği gibi, tüketici makinelerinde belirli

işlem/hesaplamaların yürütülmesini öngören daha karmaşık adımları da içerebilir. Doğal olarak her olay üreticisinin kendi gereksinimlerine yanıt verecek özel agvent tipleri yaratması beklenir.

2. *Tüm agventlerin paylaştıkları ortak davranışlar.* Gezgin etmen niteliğindeki agventler olay sisteminde yer alan Agvent Sunucular üzerinde gezinip, kendi kendilerini belirledikleri hedeflere yönlendirme yeteneğine sahiptirler. Her agvent, tüketici ölçütlerini değerlendirerek hedef tüketicileri seçme, bu tüketicilerin yer aldıkları düğümlere ulaşmak için izlenmesi gereken yolu belirleme, kendi kopyalarını yaratarak bu hedeflere yönlendirme gibi olay sistemine doğrudan destek sağlayan yönetimsel işlevleri yerine getirir. Sunucu bilgi tabanından edindiği bilgileri kendi amaçlarına göre yorumlayarak karar alır ve uygular. Agventlerin bu davranış özellikleri sisteme esnek bir yapı kazandırdığı gibi, olay sunucuların tasarım ve geliştirme karmaşıklığını da önemli ölçüde azaltır

ABDES Sistemi, hem zeki ve gezgin etmen yapısının, hem de kayıt/yayım modeliyle haberleşen olay tabanlı sistemlerin özellikleri birleştiren bir Etmen Tabanlı Dağıtılmış Olay Sistemidir. ABDES Sisteminin Dördüncü kısımda bahsedilen Dağıtılmış Olay Sistemlerinden farklı olan özellikler bir sonraki bölümde detaylandırılmıştır.

5.1. ABDES Sisteminin Sahip Olduğu Özellikler

ABDES Sistemi, hem gezgin-zeki etmen yapısının *amaca yönelik davranan, özerk ve taşınabilirlik* özelliklerinden, hem de kayıt/yayım modeliyle haberleşen olay tabanlı sistemlerin temel özellikleri olan *Bir uçtan çok uca haberleşme (point-to-multipoint), asenkron haberleşme ve gevşek bağlı sistemlerde kullanıma uygunluk* özelliklerini birleştiren bir Etmen Tabanlı Dağıtılmış Olay Sistemidir. Özellikle bilgi saklamanın ön planda olduğu, kıymetli ve büyük verilerin kullanıldığı dağıtılmış olay sistemi uygulamalarında kullanılan ABDES Sisteminin temel özelliklerini şu ana başlıklar altında toplanabilir.

5.1.1. Özerklik

Daha önceleri geliştirilen olay sistemlerinde olay verileri basit formatta düşünülmüş olup bu verilerin sistem üzerinde hızlı ve etkin şekilde yayılmaları amaçlanmıştır. Bu nedenle olay verileri çoğunlukla yalın anlamda işlenebilen ham veri olarak düşünülmüş, Şekil 5.1.a'da görüldüğü gibi katarlardan oluşan mesajlar veya tuple tabanlı mesajlar olarak geliştirilmiştir.

```
string event = account/debit
time date = 15.01.2000
int number = 12345
float amount = 215.31
```

a) Yalın Veri Örneği

```
public class Hisse_Fiyat {
private String company;
private float value;
private int amount;

public String getCompany()
        { return company; }
public float getValue() {...}
public int getAmount() {...}
public StockQuote(String c, float v, int a) {
        company = c;
        value = v;
        amount = a;
        }
}
```

b) Tip Tabanlı Olay Verisi Örneği

Şekil 5.1 : Olay Verisi Örneği

Tip tabanlı olay sistemlerinde ise olay verisine Şekil 5.1.b'de görüldüğü gibi biraz daha fazla yapısal özellik kazandırılmış olmasına karşın nesne yapısında geliştirilen bu olay verisinde basit manada veri alıcı (getter), veri atayıcı (setter) metotları ve yapılandırıcılar tanımlanmış olup özerklik verilmemiştir.

ABDES Sisteminde olaylar basit bir olay verisi olarak görülmez ve sistemin temel elemanı, birinci sınıf üyesi olarak üretilirler. Üretilme aşamasında agventler birer gezgin etmen olarak üretilir ve bu aşamada, içine kendi amaçları, inançları (çevre bilgisi) ve davranışları yüklenir. Bu gezgin etmen, bir agvent sunucusuna ulaştığı zaman, kontrolü yine AS'suna devretmemekte, AS'nun kayıt tablosunu inceleyerek kendi hedef makinelerini özerk olarak seçmekte ve kendisini bu makinelere göç ettirmekte/kopyalamaktadır. Bu yaklaşım sayesinde olay sunucusu geliştirmenin zorluğunu ve karmaşıklığını da ciddi ölçüde azaltılır ve yönlendirme işlemi agvent tarafından yapılması sağlanmış olur. AS'nun yönlendirme işlemlerindeki etkinliğinin azalması, agventlerin içerdikleri bilgilerin saklanmasına da olanak sağlamaktadır.

5.1.2. Tip/Agvent Tabanlı Kayıt

İkinci kısımda detaylı olarak bahsedildiği üzere, günümüze kadar geliştirilmiş olan olay sistemleri kayıt mekanizmasına göre dört ana sınıfta gruplandırılmıştır. Bunlardan *"Kanal Tabanlı"* yapıda bir olay tüketici sisteme kayıt olurken ilgi duyduğu kanala kayıt olur ve o kanala gönderilen tüm olay verileri kendine ulaştırılır. *"Konu Tabanlı"* yapı ise, tek kanallı bir yapıya benzemekte olup mesajlar aynı kanala konulur. Olay tüketicisinin ilgi duyduğu olay verileri özel tanımlı bir alan (konu alanı) üzerinden ifade edilir ve bu konu ölçütüne uyan olay verileri kendine ulaştırılır. *"İçerik Tabanlı"* yapıda ise olay verisi tamamı ile yapısal bir şekilde biçimlendirilmekte olup kayıt aşamasında olay verisinin bütün alanlarından

faydalanılabilir ve değişik ölçütler ifade edilebilir (bakınız Şekil 5.2.a). *"Tip Tabanlı"* kayıt yapısında, nesne olarak gönderilen olay verilerinin, tipi üzerinden olay üreticilerinin bazı ölçütler belirleyerek kayıt yapmasına olanak sağlanır. Burada kullanılan ölçüt alanlarının (*event, number, amount*) önceden sisteme tanımlanmış olması gerektiğinden bu statik bir yapı sağlamaktadır. Şekil 5.2.b'de gösterilen örnekte de olduğu gibi, Hisse_Fiyat obventi üzerine yapılan kayıt ile aynı zamanda ilgili obvent ulaştığı zaman test edilecek filtreler birinci blokta, yürütülecek işlemler de ikinci blokta gösterilmektedir.

```
string event == account/*
int number == 12345
float amount >= 100000.00

and then

string event == account/*
int number == 56789
float amount < 250.50
```

a) Kayıt Filtresi Örneği

```
Subscription s = subscribe (Hisse_Fiyat q)
{
return (q.getPrice() < 5000 &&
q.getCompany().indexOf("SAHOL") != -1);
}
{
System.out.print("Hisse Al: ");
System.out.println(q.getPrice());
}
```

Birinci Blok

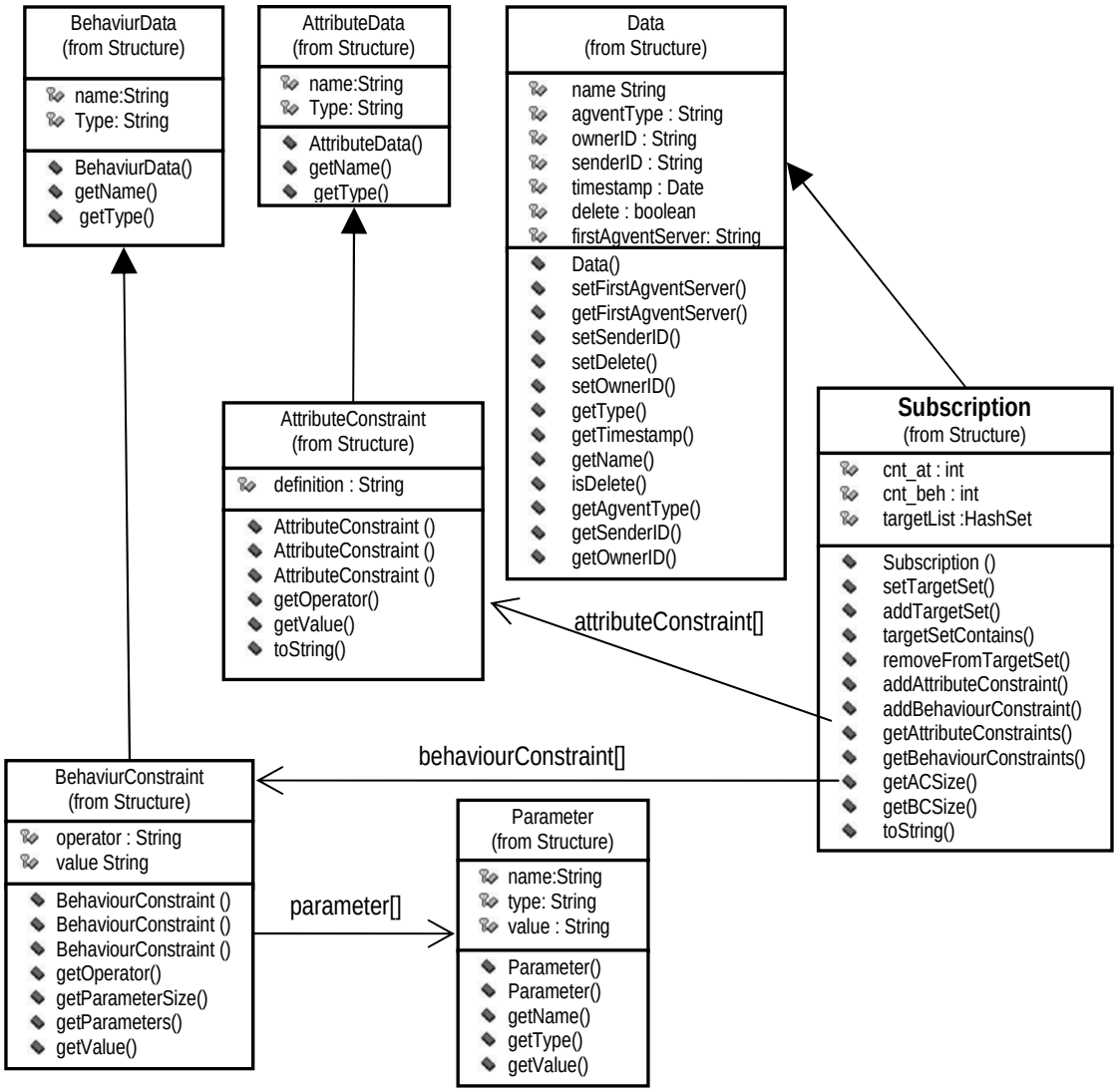
İkinci Blok

b) ;Tip Tabanlı Sistemlerde Kayıt Filtresi Örneği

Şekil 5.2 : Kayıt Filtresi Örnekleri

Tip Tabanlı Sistemlerin kayıt mekanizmasından da görüleceği üzere, kayıt komut dizini standart bir program koduna benzememektedir. Sistem kendisi için özel bir dil yapısı oluşturmuştur. Onun için üretilen bu kayıt komut dizini önce bir önderleyiciden geçirilerek standart Java programı haline getirilir ve obventler için gerekli olan adaptörler(Tip Tabanlı sitemle bağlantıyı sağlayan sınıf yapıları) üretilir. Tip Tabanlı Kayıt modelinde olduğu gibi, ABDES Sisteminde de sisteme dahil olan kayıtçılar ilgi duydukları olaylara, ilgili agventin tipini bildirerek kayıt olacaklardır. Olay bildirimlerinde aranan ölçütler kayıtçının sisteme göndereceği kayıt (*Kayit_Ol*) çağrısında ayrıntılı olarak yer alacaktır.

Kayıtçılar sisteme ilgi duydukları agvent ile ilgili ölçütleri gönderecekleri bir Kayıt nesnesi ile ifade ederler. Şekil 5.3'te UML diyagramı gösterilen bu kayıt nesnesinde ilgi duydukları agvent tipi *agventType* alanı ile ifade edilmekte olup, kayıt aşamasında hem agventin özelliklerini ifade eden değişkenlerin (attributes) üzerine, hem de davranışlarını ifade eden metotların (behaviours) üzerine ölçütler belirlemeye olanak sağlanmaktadır. Bu sayede gerek aramada gerek filtrelemede esnek ve hızlı bir çalışma sağlanmakta olup dinamik yapısı sayesinde yeni agventlerin sisteme tanımlanmasına ve her agvent tipi için farklı özelikler ve davranışlar üzerinden ölçütler oluşturulmasına olanak sağlanmaktadır.



Şekil 5.3 : Kayıt Sınıfının UML Diyagramı

5.1.3. Bilgi Saklama

Günümüze kadar geliştirilmiş olay sistemlerinin hemen hemen hepsinde, sisteme dışarıdan girebilecek herhangi bir kullanıcının olay verisine ulaşma imkanı vardır. Kullanıcı sisteme kayıt olurken tüm olay verilerini almak istediğini gösteren bir filtre ile sistemde dolaşan olay verilerinin hepsini dinleyebilir. Bunu engellemek için kayıtçıların sisteme katılma aşamasında bir kullanıcı adı ve şifre kontrolü yapılabilir, ancak bu da bilgi güvenliğinin sağlanmasında yeterli olmayabilir. Güvenlik aşamasını sadece olay tüketicileri seviyesinde değil aynı zamanda olay sunucuları seviyesinde de incelemekte fayda vardır. Günümüze kadar geliştirilmiş olan olay sistemlerinde olay sunucuları gerekli dağıtımı yapabilmeleri için olay verisinin tamamının veya belirli kısımlarını (konu alanı gibi) görmeleri gerekir. Böyle bir durum ise çok gizli olan bilgilerin transfer edilmesinde ciddi güvenlik problemleri çıkarabilir. Üst düzeyde bilgi saklama işleminin yapıldığı “tip tabanlı kayıt/yayım

(Obvent)" sisteminde dahi bir nesne içindeki veriye ulaşmak için ilgili metotları çağırmak yeterli olmaktadır. Hatta nesne değişkenlerini değiştirmek için gerekli metotlar da bu nesne içinde var ise nesne değişkenlerini kolaylıkla olay sunucusu üzerinden değiştirmek mümkündür.

ABDES Sisteminde ise oluşan olaylar birer gezgin etmen yaratır ve bu etmen AS'lar arasında kendini yönlendirir. Bu sayede kendi gizli ve özel bilgilerine, üzerinde bulunduğu AS'nun veya Kayıtçının ulaşmasına imkan vermez ve erişimleri engellenir. AS'su, ve Kayıtçı olay verisine ulaşamaz, olay verisine ilgili etmenlerin kendi aralarında yürüttükleri kontrol işlemleriyle (mesajlaşmalar ile) erişilmesine olanak sağlanır. Bu sayede veri güvenliği sadece olay/AS'sunun görevi olmaktan çıkar ve gönderilen agventin de güvenlik konusunda etkin bir şekilde rol alması sağlanır.

5.1.4. Kullanıcı Tanımlamalı Agvent Tipleri

Günümüze kadar geliştirilen olay sistemlerinde genellikle sabit olay veri yapıları kullanılmış ve farklı olay verilerinin üretimi/kullanımı kısıtlanmıştır. Sisteme dahil olan bir olay üretici önceden tanımlı olay tiplerinden bir olay verisi (integer, string gibi) gönderir, veya daha ileri seviyeli sistemlerde olay yapısına (record modellemeli veya tuple modellemeli gibi) uygun olarak bir olay verisi oluşturur ve onu gönderir. Eğer olay üretici kendine has bir olay verisi üretecek ve bunun sistem üzerinden yayımlanması gerekecek ise, gerek dağıtım sistemi içinde bulunan olay sunucularında, gerekse olay tüketicilerinde programsal değişiklikler yapmak gerekir. Bu durum ek işgücüne ihtiyaç duyması ve zaman kaybına sebep olmasından dolayı maliyeti artırır.

ABDES Sistemin de ise, olay üreticilerinin, sistem tarafından önceden tanımlanmış olan agvent tiplerini kullanmak zorunda olmayıp, kendi olay modellerine uygun yeni agventler tanımlayıp üretmelerine olanak sağlanır. Önceden tanımlanmış agvent tipleri sistemde mevcut ve kullanıyor olsa bile yeni agvent tiplerinin sisteme duyurulması (*Duyuru_Yap-advertise* çağrısı ile) ve kullanılması mümkündür. Sisteme özellikleri ve yapısı duyurulan bir agvent üzerine artık Kayıtçıların kayıt olma imkanları vardır. Bu da sisteme esneklik kazandırmaktadır.

5.1.5. Kendi Kendini Yönlendiren Agventler

Geçmişte geliştirilen olay sistemlerinde olay verilerinin dağıtımı doğrudan olay sunucularının görevidir. Sisteme gönderilen olay verisi, dağıtım sistemi tarafından sistemde kayıtlı bulunan, ilgili olay tüketicilerine ulaştırılır. Olay sunucularının oluşturduğu ağ içinde, olay verilerinin izleyeceği yol sabit olarak tanımlanabileceği gibi, hattın yoğunluğuna göre dinamik olarak düzenlenebilecek bir yapıda da olabilir.

Bu yönlendirme yapısına/modeline olay verisinin üzerinde bulunduğu olay sunucusu karar verir.

Agventin gezgin etmen yapısından dolayı her agvent, ilgili olay tüketicilerine ulaşması için izlemesi gereken yolu, AS'sunun imkanlarından en alt seviyede faydalanacak şekilde bulur. Sunucuya ulaşan agvent, o sunucunun bilgi tabanını inceleyerek kendi hedef makinelerini ve bu makinelere nasıl ulaşacağını/rotasını belirler. Bu rotaya uygun olarak agventi yönlendirmek ise AS'nun görevidir. Yönlendirme işlemlerinde minimum etkiye sahip olan AS sadece o düğüme ulaşan agventlerin çalışması için uygun bir platform sağlamaktan, hedef makinelerini ve erişim rotasını belirleyen agventi, rotaya uygun olarak göndermekten ve kendine gelen kayıt bilgilerini saklayarak sunucu üzerinde çalışan agventlerin bu bilgi tabanından etkin şekilde faydalandırmaktan sorumlu olur.

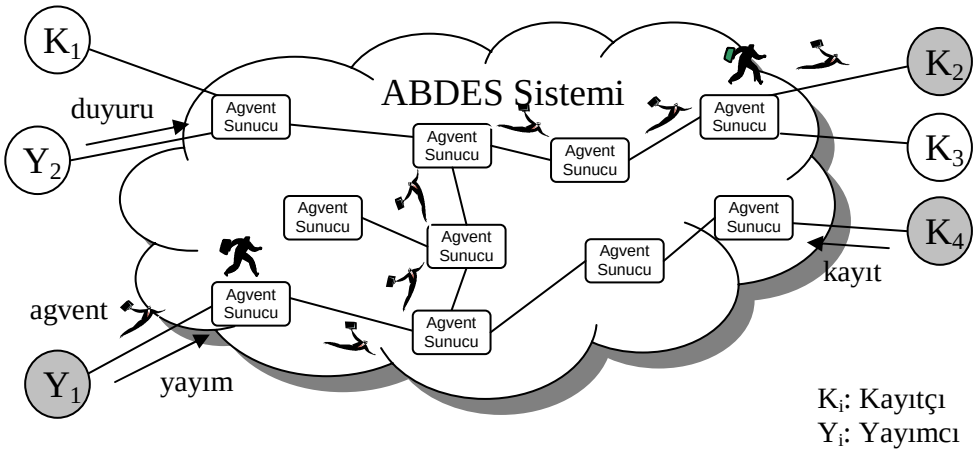
5.1.6. Çift Taraflı Filtreleme

Dağıtılmış olay sistemlerinin yapıları gereği Kayıtçı tarafından filtreleme yapılması gereklidir. Bu filtre sayesinde gereksiz verilerin/mesajların sistem üzerinde dağıtılması engellenerek hat yoğunluğu azaltılabilmektedir. İleride *"ABDES Sisteminde Yönlendirme"* kısmında da detaylandırıldığı üzere, olay verilerinin hızlı şekilde yayılmalarının ve gereksiz yere olay verilerinin ilgisiz sunuculara ve Kayıtçılara gönderilmesinin engellenmesi açısından günümüzde geliştirilen çoğu dağıtılmış olay sisteminde kayıt/yayım haberleşme modeli kullanılmakta ve dağıtım sistemi üzerinde uygun filtreleme mekanizmaları çalıştırılmaktadır. Son yıllarda geliştirilen dağıtılmış olay sistemlerinde,olay verilerinin etkin dağıtımı için *kayıt mesajlarının* kullanılması gibi, kayıt mesajların dağıtımında da benzer bir filtreleme için *duyuru mesajları* kullanılmaktadır. Burada bahsedilen her iki filtrelemede de nihai amaç sistemdeki mesajların dağıtımında, bu mesajla ilgisi olmayan bileşenlere gereksiz veri gönderilmesini engellemektir.

ABDES Sistemi de Kayıtçı tarafından yapılan filtrelemeleri desteklemekle beraber, Yayımcının ürettiği agventinde kendi filtrelemesini yapabilmesine olanak sağlamaktadır. Bu sayede agvent üretici, ürettiği agventin belirli ölçütleri sağlamayan Kayıtçılara (olay tüketicilere) gönderilmesini engelleyebilmektedir. Burada yönlendirme işlemi agventin kendisi tarafından yapıldığı için bu filtreleme işlemi de agvent tarafından kolaylıkla gerçeklenebilmektedir. Bu filtrelemenin yapılabilmesi içinde yine AS üzerindeki bilgi tabanından faydalanılmakta olup sistemin filtreleme mekanizması ileriki konularda detaylandırılmıştır.

5.2. ABDES Sisteminin Altyapısı

Bütün itme tabanlı kayıt/yayım sistemlerinde olduğu gibi, ABDES Sisteminde de bilgi akışı Yayımcıdan Kayıtçıya doğru olmaktadır. Gönderilen olay mesajının/agventin izleyeceği yol ise Kayıtçının gönderdiği, kayıt ölçütlerini içeren kayıt mesajlarıyla düzenlenmektedir. Kayıtçılar, sisteme ilgi duydukları olay tiplerini gönderdikleri kayıt mesajları[22] ile belirtirler. AS kendine ulaşan kayıt mesajlarını ilgili AS'larına doğru yönlendirir ve kendi bilgi tabanında saklar. Bir agvent AS'suna ulaştığı zaman onun bilgi tabanına erişerek kendi yönlendirmesini yapar [78].



Şekil 5.4 : ABDES Sisteminin Altyapısı

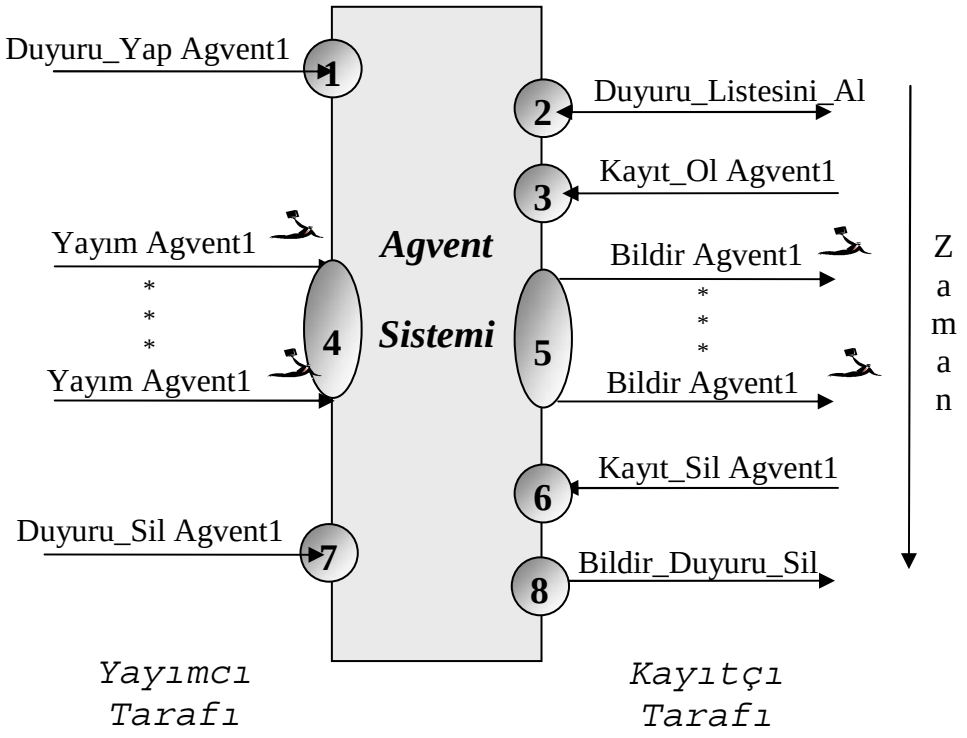
ABDES Sisteminin dağıtımdan sorumlu olan merkez kısmı, bir topoloji ile birbirine bağlanmış AS'ları oluşturmaktadır (bakınız Şekil 5.4). Sistemin diğer iki bileşeni olan Kayıtçı (olay tüketicisi) ve Yayımcı (olay üreticisi) bir AS'su üzerinden sisteme dahil olurlar ve kendi mesajlarını bu sunucu vasıtası ile sisteme iletirler. Her AS kendine gelen kayıt ve duyuru mesajlarını bir protokole göre alır (flooding veya tabloya göre yönlendirme) ve ilgili/gerekli komşu sunuculara ve Kayıtçılara iletir. Gönderilen agventlerde benzer şekilde bir rota izleyerek Kayıtçıya ulaşır. Sistemde dolaşan duyuruların, kayıtların ve agventlerin yönlendirme mekanizması biraz geniş bir konu olması sebebi ile yedinci bölümde anlatılmıştır.

ABDES Sisteminde bir agventin nasıl yayımlandığını anlamak için, agventin sisteme gönderilmeden önce ve gönderildikten sonra yapılan mesajlaşmaların neler olduğunu incelemek gerekmektedir. Bu konu, bir sonraki bölümde detaylandırılmıştır.

[22] Bazı olay sistemlerinde bu kayıt mesajları kayıtçının profili olarak adlandırılmaktadır.

5.3. Mesaj/Agvent Akışı

ABDES Sisteminin nasıl çalıştığını görebilmek için basit bir Agvent dağıtımı için yapılması gereken işlemlerin bir mesaj akışı şemasında nasıl yapıldığının incelenmesinde fayda vardır. Şekil 5.4'ten de görüleceği üzere ABDES Sisteminde, sistem bileşenleri farklı mesajlaşmalar ile kendi aralarında haberleşme sağlamaktadırlar. Sistemin bileşenleri (AS'lar, Kayıtçılar ve Yayımcılar) arasındaki haberleşme Internet üzerinden Java RMI ile sağlanır. Yapısal tarafsızlık özelliğinden dolayı Java RMI, fiziksel olarak dağıtılmış olan heterojen makinelerde çalışabilmekte ve bileşenlere saydam bir ara yüz sağlayabilirler.



Şekil 5.5 : ABDES Sisteminde Kullanılan Temel Mesajlaşma Modeli

Sabit bir etmen olarak görülebilen bir AS sunun görevi kendine gelen mesajlara göre değişir. Şekil 5.5'te gösterilen ABDES Sisteminde kullanılan temel mesajlar ve bu mesajlara göre AS'sunun yaptığı işlemler şunlardır:

1. *Duyuru_Yap (advertise):* ABDES Sistemi, kullanıcı tanımlamalı agventlerin kullanımına da olanak sağlamaktadır. Yayımcının yeni bir agventin yayımına başlayacağı zamanı, üreteceği agventin özelliklerini (attributes) ve davranışlarını (behaviours) sisteme bildirmesi gerekmektedir. Bu sayede sisteme dahil olan Kayıtçılar, ilgilendikleri agventin hangi özellikleri ve davranışları üzerine nasıl bir filtreleme yapabileceklerine karar verebilirler.

Duyuru_Yap çağrısı, Yayımcının üreteceği agventle ilgili gerekli bilgileri ABDES Sistemine bildirmek için kullanılan çağrıdır. AS, Yayımcının *Duyuru_Yap* çağrısını alarak ilgili agventin bilgilerini, kayıt mesajlarının yönlendirilmelerinde kullanılmak üzere kendi duyuru tablosuna işler ve kendine doğrudan bağlı olan sunuculara da kendi duyuru tablolarını güncellemeleri için gönderir. Birden fazla Yayımcının aynı agvent tipinde yayımı mümkündür.

2. *Duyuru_Listesini_Al (getAdvertisementList):* Kayıtçı sisteme kayıt olurken hangi agventlerin yayımın yapıldığını öğrenmek için bu çağrıyı kullanır. *Duyuru_Listesini_Al* çağrısı ile Kayıtçı kendi belirlediği ölçütlere uygun olan duyuruları alır ve bu duyurulara göre agventlerin özelliklerini yorumlayabilir ve istediği agvent üzerine kayıt olabilir.
3. *Kayit_Ol (subscribe):* Bir Kayıtçı, bir agvent ile ilgilenmesi durumunda üretilen bu agventin kendine yönlendirilmesini sağlamak için *Kayit_Ol* mesajını ABDES Sistemine gönderir. AS'su, Kayıtçıdan gelen *Kayit_Ol* mesajını alır ve bu mesajı inceleyerek, kayıt olunmak istenen agventin daha önce yayım yapılacağının duyurulup duyurulmadığını (duyuru tablosunda (advertisement table) agvent kaydının var olup olmadığını) kontrol eder. Eğer yayımı duyurulmuş bir agvent ise bu Kayıtçının profilini kendi kayıt tablosuna (subscription table) işler ve kendine bağlı ilgili AS'lara da işlemeleri için iletir. Kayıt mesajı, duyuru mesajının dağıtımı sırasında izlenen yolun tersi istikamette dağıtılır. Bu mesajların dağıtımı *"ABDES Sisteminde Yönlendirme"* kısmında detaylandırılmıştır.
4. *Yayım (publish):* Bir Yayımcı daha önce duyurusunu yapmış olduğu bir agventi ürettiği zaman bunu olay sistemine *Yayım* çağrısı ile yayımlar/gönderir. Yayım çağrısı agventin dağıtımını başlatan ilk çağrıdır. ABDES Sistemi kendine ulaşan agventin, istekçilerine kendisini yönlendirmesi için gerekli altyapıyı sağlamaktan sorumludur.
5. *Bildir (notify):* Agvent, ABDES Sistemi üzerinde hareket ettikten sonra, ulaşacağı Kayıtçıya, Kayıtçının *Bildir* metodu vasıtası ile ulaştırılır ve orada Kayıtçı tarafından aktif hale getirilerek çalışmasına uygun ortam sağlanır.
6. *Kayit_Sil (unsubscribe):* Kayıtçı daha önce ilgi duyduğunu belirttiği bit agvente artık ilgi duymuyor, üretilen bu agventten haberdar edilmek istemiyor ise, AS'suna bir *Kayit_Sil* mesajı gönderir. Sunucu bu mesajı alınca kendi kayıt tablosundan kayıt verisini siler ve kendine bağlı ilgili AS'larından da bu kaydın silinmesini talep eder.

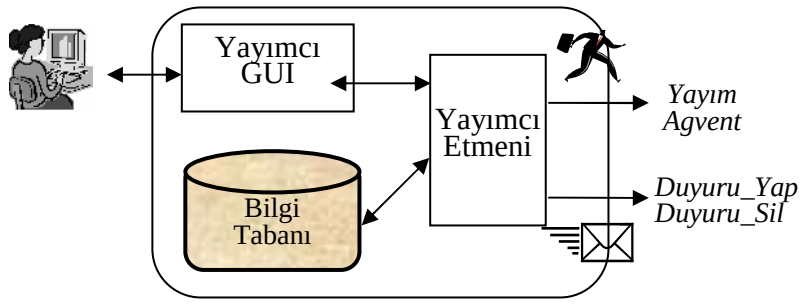
7. *Duyuru_Sil (unadvertise):* Daha önceden yayımı yapılacağı duyurulan bir agventin artık yayımının yapılmayacağının ABDES Sistemine bildirilmesidir. Bu mesajı alan sunucu, ilgili agvent kaydını, kendi duyuru tablosundan çıkarır ve kendisi ile bağlantılı sunuculara da aynı işlemi yapması için gönderir. Böylece yayımı yapılmayacak agventler üzerine Kayıtçıların kayıt olması engellenir.
8. *Bildir_Duyuru_Sil (notifyUnadvertise):* Sisteme yayımlanacağı duyurulan bir agvent tipinin artık yayımının yapılmayacağı *Duyuru_Sil* ile bildirilmiş ve bu agventi yayımlayacağını duyuran başka Yayımcı da kalmamış ise, bu agvent tipine ilgilerini bildirmiş olan Kayıtçıların, bu durumdan haberdar edilmeleri ve kayıtlarının silinmesi gerekir. Kayıtçıya bu bilginin iletilmesi *Bildir_Duyuru_Sil* metodunun çağrılması ile yapılır.

5.4. ABDES Sisteminin Bileşenleri

ABDES Sisteminin bileşenleri üç temel grupta sınıflandırılabilir. Bunlar sisteme olay verilerini sağlayan *Yayımcılar*, bu Yayımcılar tarafından üretilen agventler üzerine kayıt olan *Kayıtçılar* ve bu agventlerin dağıtımına olanak sağlayan AS'*lardır*. Bir sonraki bölümde bu bileşenlerin yapıları, işlevleri ve alt bileşenleri tanımlanmaktadır.

5.4.1. Yayımcı

Yayımcı olayları gözlemleyerek sisteme veri sağlayan birimdir. Aynı zamanda agventin çalışma kurallarını tanımlayan özerk olarak çalışmasına olanak sağlayan davranışlarını belirleyen, agventi oluşturan ve sisteme gönderen sistem bileşenidir. Bir agventi sisteme yayımlayabilmek için Yayımcının Şekil 5.6'da iç yapısı gösterilen "*Yayımcı Altsistemini*" kendi makinesine kurması ve etkin kılması gerekir. Yayımcının ABDES Sistemine bir AS'su üzerinden bağlanması gerekir ve bu bağlantının kurulması içinde Yayımcının bağlanacağı AS'sunun adresini bilmesi ve gerekli ön bilgileri sağlayarak kendisini sisteme tanıtması gerekir.



Şekil 5.6 : Yayımcı Altsistemi Yapısı

Yayımcı altsistemi Şekil 5.6'da görüldüğü gibi, üç temel bileşenden oluşur:

1. *Bilgi Tabanı:* Daha önceki işlemler/yayımlanmış agventler ile ilgili verileri içeren bir veritabanı.
2. *Grafiksel Kullanıcı Ara yüzü (GUI):* Kullanıcılarla iletişimi sağlayan görsel arabirimdir.
3. *Yayımcı Etmeni:* Yayımcı kullanıcısının adına çalışan etmendir. Yayımcı etmeni, Yayımcı sisteminin merkezi birimidir. Hareket etmeyen, sabit bir etmen olarak kurulum zamanında oluştulur, ve üretilecek agventin duyurulması ve yayılması aşamalarında gerekli olan dağıtım desteğini sağlar.

Yayımcının üç temel işlevi vardır. Bunlar:

1. ***Duyuru Yapılması:*** Bir Yayımcı yayımlayacağı agventlerin filtrelenebilir özelliklerini ve davranışlarını[23] izah ettiği agvent tanımlamalarını ABDES Sistemine *Duyuru_Yap* çağrısı ile duyurur. Sisteme bağlı bulunduğu süre içinde de herhangi bir yeni agvent tipini de üreteceği zaman, aynı şekilde bu agventin tanımlamalarını ABDES Sistemine duyurur. Gönderilen bu mesaj AS'ları üzerinde bir duyuru tablosunda saklanır. Kayıtçılar ilgi duyup kayıt olacakları olay/agvent tipleri hakkında bilgi almak için bu duyuru tablosundan faydalanırlar.

   Duyuru mekanizması Bilgi Tabanının fazla büyümemesi ve sistemin hızlı çalışması için önemlidir. Duyuru mesajlarının kullanılması sayesinde kayıt mesajlarının sistem üzerinde dağıtımının yapılması için bir dağıtım ağacı yapısı kolaylıkla oluşturulmakta ve kayıt mesajlarının, ilgisiz sunuculara dağıtılması engellenmektedir. Yayımcı, bir agventi yayımlamadan önce, bu agventi bir *Duyuru_Yap* çağrısı ile sisteme tanıtmakta, bu sayede sisteme katılmak isteyen Kayıtçılar, yayımı yapılabilecek agvent tipleri hakkında bilgi alabilmekte ve buna göre kayıt olabilmektedir. Sistemin dağıtımına destek vermediği bir agvent üzerine kayıt olunması, AS'larının üzerindeki kayıt tablosunun gereksiz yere büyütülmesini sağlayacaktır. Bunun sonucunda da sorgulama işlemleri ve agventin gönderileceği Kayıtçıların ve AS'larının seçimi uzun zaman alacaktır. Duyuru mekanizması sayesinde filtreleme işlemlerinin hızlı ve etkili çalışması sağlamakta ve ağ trafiğini azaltılmaktadır. Duyuru mesajlarının kayıt mesajlarının yönlendirilmesindeki işlevleri 7 nci bölümde izah edilmiştir.

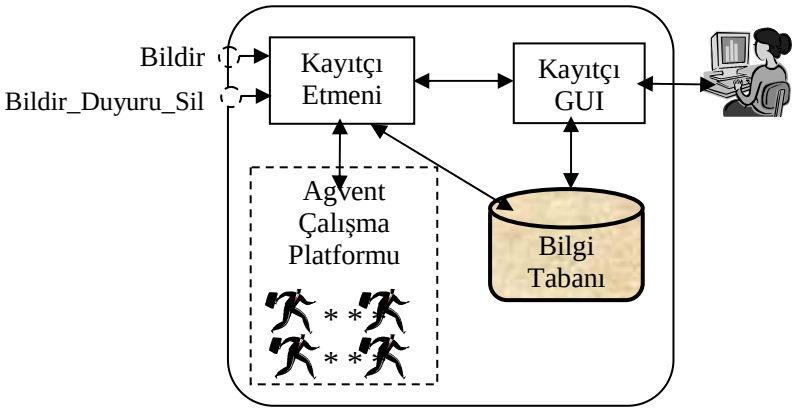
[23] Filtrelenebilecek her durum değişkeni ve agvent davranışları "Duyuru_Yap" metodu ile olay sistemine duyurulur

2. ***Duyuru iptali:*** Eğer Yayımcı belirli bir agvent tipi hakkında yayımını sonlandıracak ise, daha önce yapmış olduğu *Duyuru_Yap* talebinin etkisini geçersiz kılmak üzere, yeni durumu *Duyuru_Sil* çağrısı ile sisteme bildirir. Eğer aynı olayın yayımını yapan başka bir Yayımcı yok ise her sunucu kendi duyuru tablosundaki Kayıtçının profilini siler ve kendine bağlı Kayıtçılara da bu durumu bildirir.

3. ***Yayım:*** Yayımcı hangi olayların gözlenebileceğine, bunların nasıl adlandırılacağına ve tanımlanacağına, ve bir olayın somut bir varlık olarak (agvent) nasıl gösterilebileceğine karar verir. Bir olayın olması durumunda ilgili agventi üretir ve ABDES Sistemine yayım çağrısı ile gönderir. Agventin dağıtılması bu çağrı ile başlatılmış olur ve agvent ulaştığı ilk AS üzerinde aktif hale gelerek kendisini yönlendireceği Kayıtçıları seçer.

5.4.2. Kayıtçı

Kayıtçılar ilgilendikleri olayları sisteme bildirmek için o olaya karşı düşen agventleri belirleyerek ABDES Sistemine kaydolurlar. Her Kayıtçı söz konusu agventin yaratılıp kendine ulaşması durumunda gelen agventin kabul edilerek aktif hale gelmesine olanak sağlayacak uygun bir çalışma platformunu sağlar.



Şekil 5.7 : Kayıtçı Altsisteminin Yapısı

Yayımcıyla benzer şekilde, bir Kayıtçı da sisteme katılmak ve kayıt isteklerini gönderebilmek için *"Kayıtçı Altsistemini"* kendi makinesine kurması ve çalıştırması gerekir. Sistemde yer almak için önceden adresini edindiği AS'suna gerekli önbilgileri, kullanıcı adı ve şifresi ile birlikte sunarak kendini tanıtır. Kullanıcı adı ve şifre denetimi sayesinde istenmeyen Kayıtçıların sistemden bilgi çekmeleri/görmeleri engellenebilir.

Şekil 5.7'de görüldüğü gibi, bir Kayıtçı altsistemi dört ana bileşenden oluşur. Bunlar:

1. *Bilgi Tabanı:* Kayıtçıya önceden gelen tüm agventlerin ve yürütülen işlemler ile ilgili kayıtlarının tutulduğu ve JDBC üzerinden erişimin yapıldığı veri tabanı.
2. *Grafiksel Kullanıcı Ara yüzü (GUI):* Kayıtçı altsisteminin kullanıcı ile bağlantısını ve veri alış verişini sağlayan ara yüz.
3. *Kayıtçı Etmeni:* Kullanıcı adına hareket eden ve gerekli işlemleri yürüten etmendir. Kayıtçı etmeni sabit bir etmen olup, kullanıcının sisteme kendi isteklerini iletme, gelen agventlerin çalışması için uygun ortam sağlama ve onları denetleme işlemlerini yerine getirir.
4. *Agvent Çalışma Platformu:* Kayıtçıya ulaşan agventlerin aktif hale getirilmesine olanak sağlayan ve amacına yönelik olarak güncelleme, dağıtım gibi işlemleri Kayıtçı etmeni ile haberleşerek yürütmesine olanak sağlayan işlem platformudur.

Geliştirilen bir Kayıtçı altsisteminin agvent bildirimlerini kabul etmek *Bildir* ve silinen duyurularla ilgili bilgi almak için *Bildir_Duyuru_Sil* metodunu içeren Int_Subscriber ara yüzünün gerçeklenmesi gerekir

```
public interface Int_Subscriber extends Remote
{       public void notify(Agvent agv);
        public void notifyUnadvertisement (Subscription sub);
}
```

Kayıtçının dört temel işlevi vardır.

1. ***Sisteme Kayıt Olma:*** Kayıtçı, sisteme kayıt olabilmek için *Kayit_Ol* çağrısını kullanarak, kendine bildirim yapılmasını istediği agventin ölçütlerini belirtir. Sistem üzerinde gerekli kayıt ve bildirim işlemlerinin yürütülebilmesi için daha önce bir Yayımcı tarafından böyle bir agventin yayımının yapılacağının duyurulmuş olması, yani sunucu üzerindeki duyuru tablosunda bu agvente ilişkin bir kaydın yer alması gerekir. Kayıtçı birden fazla agvent üzerine kaydolabilir.
2. ***Sistemdeki Kaydını Silme*:** Kayıtçı, bir agventin oluşması durumunda haberdar edilme isteğinden vazgeçerse, mevcut kaydını geçersiz kılmak için, Yayımcıya benzer şekilde, *Kayit_Sil* çağrısını kullanır.
3. ***Bir Agventi Kabul Etme:*** ABDES Sistemi üzerinden kendini yönlendirerek, istekli Kayıtçıya ulaşan agvent, kendini bu Kayıtçıya *Bildir* çağrısıyla iletir. Kayıtçıya ulaşan agvent burada aktif hale geçirilerek çalışması sağlanır.
4. ***Yayımlanmayacak Agventler hakkında bilgi edinme:*** ABDES Sistemine daha önce yayımlanacağı duyurulan bir agventin artık yayımı sonlandırılacak

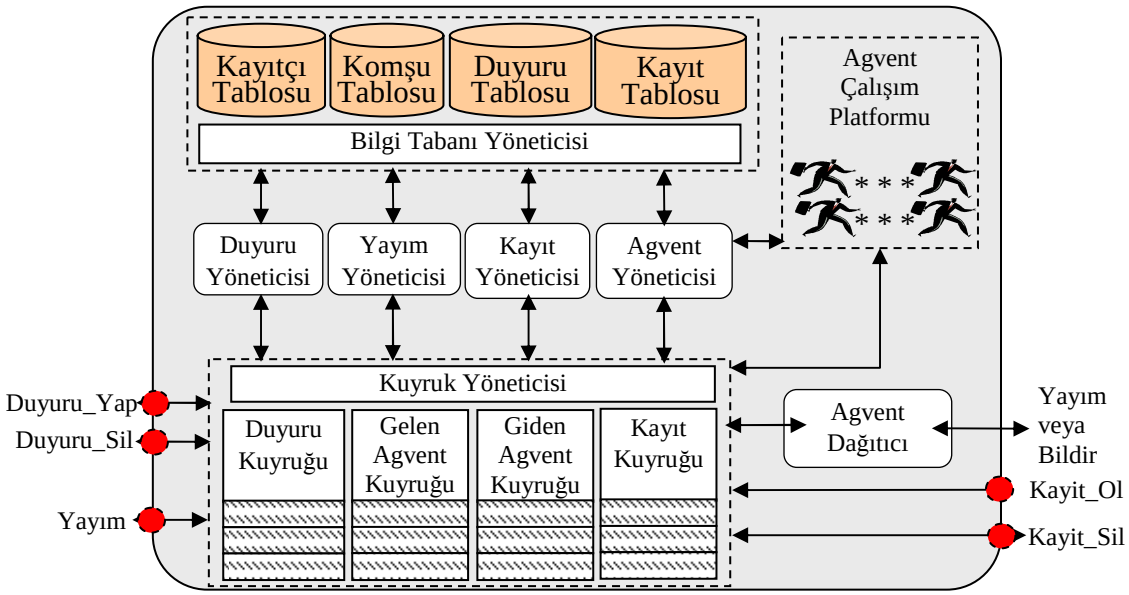
ise bu durum Yayımcı tarafından bir *Duyuru_Sil* çağrısı ile sisteme bildirilir. Bu çağrı yapılana kadar sisteme kayıt olan Kayıtçıların bilgilendirilmeleri için, bu Kayıtçılar sistem tarafından *Bildir_Duyuru_Sil* çağrısı ile yeni durumdan haberdar edilirler.

5.4.3. Agvent Sunucu

ABDES Sistemi Şekil 5.4'te görüldüğü gibi, farklı topolojilerde birleştirilebilen AS'larından oluşur. ABDES Sisteminin temel görevi kendine ulaşan agventin en hızlı şekilde bu agventle ilgilenen Kayıtçılara ulaştırması için uygun çalışma ortamını oluşturmaktır. Bu nedenle AS'larının belirli bir düzen içinde çalışmaları ve kendi aralarında işbirliği yapmaları gerekir. Bir AS'sunun işlevini yerine getirebilmesi için aşağıda gösterilen Int_AgventServer ara yüzünü gerçeklemesi gerekir.

```
public interface Int_AgventServer extends Remote
{       public void Yayım(Agvent agv);
        public void Kayit_Ol(Kayit sub);
        public void Kayit_Sil(Kayit sub);
        public void Duyuru_Yap(Duyuru adv);
        public void Duyuru_Sil(Duyuru adv);
        public Duyuru[] Duyuru_Listesini_Al(String filter)
}
```

Bir Agvent Sunucusunun iç yapısı Şekil 5.8.'de gösterildiği yedi yöneticiden oluşmaktadır. Bu yöneticilerin işlevleri aşağıda sıralanmıştır.



Şekil 5.8 : Bir Agvent Sunucusunun İç Yapısı

Bilgi Tabanı Yöneticisi: Bu birimin görevi AS'sunun çalışması için gerekli olan verilerin saklandığı dört ana tabloya erişimi düzenlemektir[24]. Bu tablolar ve içeriklerini şu şekilde sıralanabilir:

1. *Komşu Tablosu (Neighbour Table);* Bir AS'sunun doğrudan bağlı olduğu diğer komşu AS'suna ait tanımlayıcı, adres gibi bilgilerin yer aldığı veri tabanı tablosudur. Bu tablo özellikle kayıt ve duyuru mesajlarının dağıtımında kullanılır.

2. *Duyuru Tablosu (Advertisement Table);* Bu tablo sistemde yayımlanan agventlere ilişkin tanımlayıcı bilgilerini tutar. Bu bilgiler agventin filtrelenebilir özelliklerinin yanı sıra, filtrelenebilir davranışları da içerir (bakınız Şekil 5.12, Şekil 5.13). Kayıtçı bu bilgilere dayanarak kendi belirlediği koşullarda bildirimlerin ulaşmasını sağlayacak ölçütleri hazırlar.

3. *Kayıtçı Tablosu (Subscriber Table);* Bu tablo Kayıtçılara ait tanımlayıcı, şifre, adres gibi belirleyici bilgilerin yanında agventin de kendi filtreleme ölçütlerine uygun olarak seçim işlemi yapabilmesi için Kayıtçının makinesinin özellikleri (hard disk kapasitesi, işlemci hızı, işlemci sayısı, bellek miktarı, Internet bağlantı hızı, vs) ve bazı şirketsel bilgileri (sermayesi, kar oranı, güvenirliği, vs) tutmaktadır.

4. *Kayıt Tablosu (Subscription Table);* En sık erişilen veri yapısıdır. Tabloda Kayıtçıların bildirdikleri kayıt ölçütleri yer alır. Erişim süresini azaltmak için veri tabanı özellikleri kullanılarak SQL komutları ile erişim sağlanır[25]. AS'suna ulaşan bir agvent, kendini ilgilendiren bilgileri değerlendirerek tanımlı ölçütlerin doğrulandığı takdirde, kendini hedef düğümlere yönlendirme işlemini yürütür.

Kuyruk Yöneticisi: Bu birimin görevi eş zamanlı çalışan diğer birimlerin, kendi denetiminde yer alan dört kuyruk yapısına, karşılıklı dışlama koşullarında erişmelerini sağlamaktır. AS'suna gönderilen bu mesajların geriye bir sonuç döndürmeleri gerekmediği için (return değerleri *void* olan metot çağrıları olduğu için), her gelen mesaja, mesajın gönderildiği çağrı içinde işlem yapmak yerine, gelen mesaj hemen ilgili kuyruğa atanarak, metot çağrısı sonuçlandırılmış olur. O mesajı işlemek ise ilgili yöneticinin (yayım yöneticisi, duyuru yönetici, agvent dağıtıcı, agvent yöneticisi) işidir. Kuyruk Yöneticisinin yönettiği kuyruk yapıları şunlardır:

1. *Duyuru Kuyruğu; "Duyuru_Yap"* ve *"Duyuru_Sil"* metotları ile AS'suna gelen duyuruların işlenmek üzere saklandığı kuyruk yapısıdır.

[24] ABDES Sisteminde Bilgi Tabanı Yönetimi konu başlığında detaylandırılmıştır
[25] Kayıt Tablosu Java Veri Nesnesi olarak tutulduğu için erişimde JDOQL sorgulaması ile yapılmaktadır.

2. *Gelen Agvent Kuyruğu;* "Yayım" metodu ile AS'suna ulaşan agventlerin işlenmek üzere saklandığı kuyruk yapısıdır.
3. *Giden Agvent Kuyruğu;* AS'sundan diğer AS'larına veya Kayıtçılara gönderilecek olan agventlerin saklandığı kuyruk yapısıdır. Bir agvent kendisini başka bir AS'ya kopyalamaya/göndermeye karar verdi ise, hedef düğüm adresi ile birlikte kendisini bu kuyruğa ekler.
4. *Kayıt Kuyruğu; "Kayit_Ol"* ve "*Kayit_Sil*" metotları ile AS'suna gelen kayıt mesajlarının işlenmek üzere saklandığı kuyruk yapısıdır.

Kayıt Yöneticisi: Kayıt kuyruğunda bekleyen kayıt mesajlarını alarak bunları AS'nun bilgi tabanına işler. Yönlendirilmesi gereken AS'ları *duyuru tablosundan* tespit eder ve kayıt mesajlarını buralara yönlendirir.

Agvent Yöneticisi: Gelen Agvent Kuyruğunda bekleyen agventleri inceleyerek bunları aktif hale getirir veya ilgili AS'ya veya Kayıtçıya yönlendirilmek üzere *Giden Agvent Kuyruğuna* ekler.

Duyuru Yöneticisi: *Duyuru Kuyruğunda* bekleyen duyuru mesajlarını alarak bunları AS'nun bilgi tabanındaki duyuru tablosuna işler. Yönlendirilmesi gereken AS'ları komşu tablosundan tespit eder ve kayıt mesajlarını buralara yönlendirir.

Agvent Dağıtıcı: AS'sundan diğer düğümlere gönderilecek olan agventlerin gönderilme işlemlerini yürütür. Giden Agvent Kuyruğunda bekleyen agventler hedef makinelere Agvent Dağıtıcı tarafından gönderilir.

Agvent Yönetici: Gelen Agvent Kuyruğunda bekleyen agventler eğer AS üzerinde aktif hale getirilmesinden ve bu agventlerle ilgili yönetimsel işlerden sorumludur.

5.5. Agventler

Sistemin temel taşı olan *Agvent* kavramının tam olarak ne olduğunu anlamak çok önemlidir. Bir agvent;

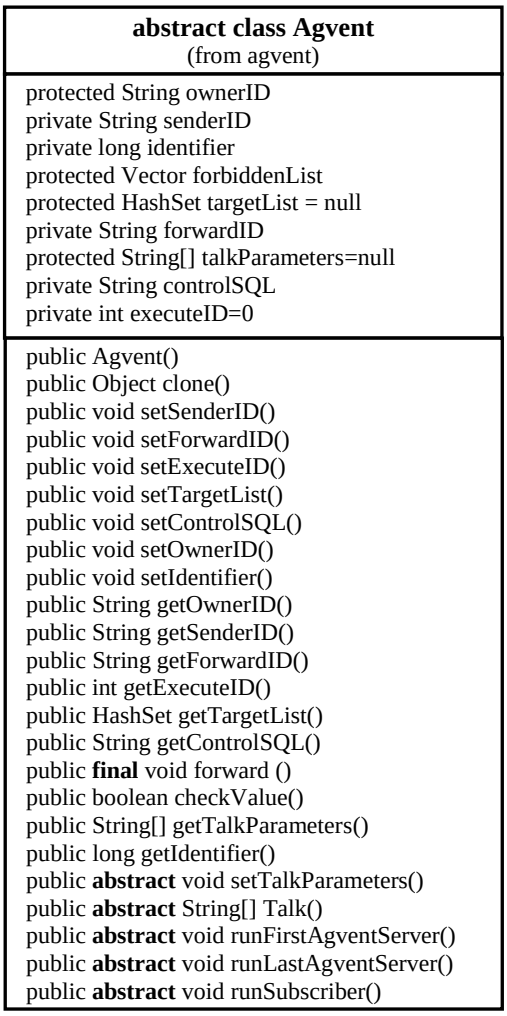
> *Yayımcılar tarafından üretilen olay verisini Kayıtçılara ulaştırmak üzere ABDES Sistemi üzerinde göç ederek hareket eden ve bağlantı düğümlerinde[26] aktif hale getirilerek göç edeceği hedef makineyi/makineleri seçip kendini özerk olarak bu makinelere yönlendiren kod ve veri bütünüdür.*

Agventler Yayımcılar tarafından üretilerek Kayıtçılara ulaşmaları için herhangi bir topoloji ile birleştirilmiş AS'*larından* oluşan *ABDES Sistemine* gönderilirler. Her düğümde yer alan bir AS'su, kendine ulaşan agventlerin çalışması, kendi yaratılma

[26] Bağlantı düğümü Agventin sistem üzerinde hareket ederken ulaştığı ilk ve son Agvent Sunuculardır.

amacına uygun işlevler yapabilmesi ve ihtiyaç duyulması durumunda kendisini (veya kopyasını) hedef bir makineye göndermesi için uygun bir çalışma platformu oluşturur.

5.5.1. Agvent Sınıfının Yapısı



Şekil 5.9 : Agvent Sınıfının UML Diyagramı

UML diyagramı Şekil 5.9'daki gibi olan agvent sınıfının çalışmasını tam anlayabilmek için, bu sınıfın temel bazı özelliklerini ve davranışlarını açıklamakta fayda vardır.

5.5.1.1. Agvent Sınıfının Temel Özellikleri

Agvent Sınıfının geliştirilmesinde kullanılan ve bütün agvent örneklerinde bulunması istenen özellikler (attributes) ve bu özelliklerin işlevleri şunlardır:

- *protected String ownerID;* ile Agventin üreticisinin tanımlayıcısı saklanmaktadır. Bu bilgiden filtreleme ve yönlendirme işlemlerinde faydalanılmaktadır.

- *private String senderID;* Bir agventin, AS'lar arasında yönlendirilmesi sırasında, agventin hangi AS'dan (veya Yayımcıdan) geldiğinin bilinmesi gerekmektedir. Eğer bir agventin senderID si ile ownerID si aynı ise bu agvent sisteme yeni gönderilmiş ve ilk AS üzerinde olduğunu göstermektedir. Bu nedenle aktif hale getirilmeli ve ilk AS'su üzerinde yapması gereken eylemleri yerine getirmelidir.
- *protected Vector forbiddenList*; Yayımcı bir agventi üreterek sisteme gönderdiği zaman, geçmiş tecrübelerinden de faydalanarak bazı kullanıcılara bu agventin gönderilmemesini isteyebilir. Bu durumda gönderilmemesini istediği Kayıtçıların adreslerini *forbiddenList* e kaydederek agventin bu kullanıcılara gönderilmesini engellemektedir.
- *protected HashSet targetList;* Üretilen agventin sistemde yönlendirilmesi sırasında, hangi hedef makinelere gönderileceği bir liste yapısında tutulmakta ve yönlendirme işlemlerinde gerek agventin kendisi, gerekse AS'lar bu listeden faydalanmaktadırlar.
- *private String forwardID;* agventlerin yönlendirme işlemi sırasında, agventin gönderileceği AS veya Kayıtçının adresinin tutulduğu kısımdır. Agvent Dağıtıcı, agventi *forwardID* adresine göndermekle yükümlüdür.
- *private String controlSQL;* Agventin sistem üzerinde dağıtımı sırasında kendi filtrelemesi için kullandığı iki farklı kontrol yapısı vardır. Birisi yukarıda bahsedildiği gibi adres/tanımlayıcı üzerinden filtreleme, diğeri de ölçütler üzerinden filtreleme. Ölçütler üzerinden filtrelemenin sağlanabilmesi için Yayımcı agventin içine test edilmesini istediği ölçütü/ölçütleri SQL formatında yükler. Agventte ilgili AS'ya ulaştığında nesne veri tabanında bu kontrol komutunu çalıştırarak, kendisini yönlendireceği Kayıtçıları seçer.
- *private int executeID=0;* Agventin üç farklı çalışma mekanizması vardır. Bunlar ilk AS üzerinde, son AS üzerinde ve Kayıtçı üzerinde. Bir agvent aktif hale gelirken hangi metotları çalıştıracağını belirtmek için *executeID* özelliğinden faydalanılır ve ilgili metotları aktif hale getirilir.
- *protected String[] talkParameters;* Kayıtçı üzerine ulaşan agventin, sahip olduğu nitelikler ve davranışlar neticesinde, agvent ile Kayıtçı arasında haberleşmede kullanılacak parametreleri ve açıklamaları saklar.

5.5.1.2. Agvent Sınıfının Temel Davranışları/Metotları

ABDES Sisteminde yayımlanacak farklı agvent tiplerinin, belirli davranışlarının standart olması gerekmektedir. Bunun için bir üst sınıf olarak tanımlanan *"Agvent"*

sınıfında bazı sabit metotlar tanımlanmış olup, bu metotlardan agventin sistem üzerinde yayımı sırasında değişik aşamalarda faydalanılmaktadır. Bu metotlar ve işlevleri şu şekildedir.

checkValue Metodu: ABDES Sisteminin önemli bir aşaması da değerlerin karşılaştırılmasının sağlandığı aşamadır. Agventin geliştirme amacına uygun olarak gerek özellikler, gerekse davranışlar farklı tiplerde olabilmektedirler. Bir Kayıtçı, bir agventin her hangi bir özelliği veya davranışı üzerinde filtreleme yapılmasını istediğinde bu karşılaştırma işlemlerinin yapılması için adaş metotlar olarak *checkValue* metodu aşağıda gösterildiği gibi geliştirilmiştir.

```
public boolean checkValue(int var1, String str, String op) { }
public boolean checkValue(float var1, String str, String op) { }
public boolean checkValue(double var1, String str, String op) { }
public boolean checkValue(boolean var1, String str, String op) { }
public boolean checkValue(String var1, String str, String op) { }
```

Bu adaş metotların yardımıyla Tablo 5.1'de gösterilen tiplerin karşılaştırılmalarına olanak sağlanmıştır. ABDES Sistemi, ihtiyaç duyulması halinde bunların haricindeki tiplerdeki (hatta bunlar basit veri tipinden olmayıp, nesne tipinden de olabilir) verilerin karşılaştırılmalarına da olanak sağlayacak şekilde geliştirilebilir.

Tablo 5.1. ABDES Sistemi Tarafından Desteklenen Özellik ve Davranış Tipleri

Tipler
int
float
double
boolean
String

Geliştirilen bu adaş metotlar, sistem tarafından tanımlanmış olan operatörleri desteklemektedir. ABDES Sisteminin desteklediği operatörler sabit olarak belirlenmiş olup bunlar Tablo 5.2'de gösterilmektedir.

Tablo 5.2. ABDES Sistemi Tarafından Desteklenen Karşılaştırma Operatörleri

Operatörler
>
>=
<
=<
!=
<>
=

Bunların haricinde agvent sınıfında, sistemin nasıl çalıştığını gösteren davranışlar/metotlar da şunlardır:

- *public final void forward():* Agventin hedef listesindeki makinelere agventin gönderilmesini sağlayan metottur. Sistemin yapısı gereği dağıtım aşamasında bir agventin kopyaları çıkarılarak sistem üzerinde yayımlanmaktadır. Agvent kendi üretilme amacına göre hedef listesini belirler ve bu hedef listesine yönlendirmelerin hangi AS'lar üzerinden olacağını hesaplar. Agvent kaç farklı AS üzerinden dağıtılacak ise o kadar farklı agvent klonlanır (kopyası çıkarılır). Bu iş için yönlendirme aşamasında AS üzerindeki Agvent Dağıtıcıdan faydalanılır.
- *public abstract void runFirstAgventServer():* Yayımcı tarafından sisteme gönderilen bir agventin ulaştığı ilk AS üzerinde nasıl bir davranış sergileyeceğinin tanımlandığı metot çağrısıdır. Agvent ulaştığı ilk AS üzerinde, sisteme daha önce kayıt olan Kayıtçıların kayıt ölçütlerini kontrol ederek, kendisini bu ölçütlere uygun Kayıtçılara doğru yönlendirir. Bu metot agventin yönlendirileceği Kayıtçıların seçildiği ve yönlendirme işleminin başlatıldığı metot çağrısıdır.
- *public abstract void runLastAgventServer():* Bir agventin, ABDES Sistemi üzerinde dolaşarak ilgili Kayıtçıya ulaşması gerekmektedir. Agvent, Kayıtçının bağlı bulunduğu AS'ya ulaştığı zaman, Kayıtçının özelliklerini inceleyerek, bu özelliklere göre kendisini ilgili Kayıtçıya doğru yönlendirip yönlendirmemeye karar verdiği metot çağrısıdır.
- *public abstract void runSubscriber():* Agventin üretim amacına göre, Kayıtçı üzerinde yapması gereken işlemlerin tanımlandığı metot çağrısıdır.
- *public final void run(): Thread* sınıfının bir alt sınıfı olarak geliştirilen *Agvent* sınıfında agventin aktif hale getirilmesi sırasında çalıştırılacak metot çağrısıdır. Agvent aktif hale geldiğinde bulunduğu makinenin özelliğine göre ilgili metotları çağırarak kendine yüklenen görevleri yerine getirir.
- *public String[] getTalkParameters();* Kayıtçı, kendine ulaşan agventle haberleşmesi sırasında, faydalanabileceği bilgileri ve bunları hangi parametrelerle nasıl çağırabileceğini öğrendiği metot çağrısıdır.
- *public abstract void setTalkParameters(); Agvent sınıfının* altsınıfı olarak üretilen her farklı agvent tipi için, haberleşme parametrelerinin tanımlandığı metot çağrısıdır. Her agvent tipinin haberleşme parametreleri farklı olacağı için bu metodun ayrı ayrı gerçeklemesi gerekmektedir. Bu nedenle metot soyut olarak tanımlanmıştır.

- *public abstract String[] Talk(String question, String parameter);* Kayıtçı ile agvent arasında haberleşmenin sağlandığı temel metoddur. Kayıtçı *getTalkParameters* metot çağrısı ile elde ettiği bilgileri kullanarak, kendine ulaşan agventten bazı ek bilgileri isteyerek ona göre belirli işlemleri yapmaya karar verebilir. Agvent kendisinden talep edilen bir isteği amacına uygun olarak değerlendirerek cevap verir. Bu cevabında, Kayıtçının bazı ölçütlerini de dikkate alarak ona göre davranışını düzenleyebilir. Her agventin konuşma işlemi farklı olacağından bu metot soyut olarak tanımlanmış olup, agvente özel olarak gerçeklenmesi sağlanmıştır.

5.5.2. Bir Agvent Örneği

Örnek olarak Bölüm 1.1.4.6 da anlatılan Yayın Dağıtım/Bildirim Sistemlerinin senaryosunu incelenecek olursa; bir yayınevi yeni basılan kitabını tanıtmak ve pazarlamak için üye bulunduğu ABDES Sistemini kullanmak istemektedir. Bunu başarabilmesi için öncelikle kendi ürün tipini (kitap) sisteme duyurması ve daha sonra her yeni ürünü sisteme yayımlaması gerekir. Yayımlayacağı ürün bir kitap olacağı için kendisi Agvent sınıfının bir altsınıfı olan PublicationAgvent sınıfını oluşturarak bunu sisteme duyurur (*Duyuru_Yap*) (bakınız Şekil 5.10).

Oluşturulan bu sınıfta sadece Kayıtçının öğrenmesinin yeterli olacağı ve/veya AS'sunun görmesinde sakınca olmayacağı düşünülen basit bilgilerin görülmesine olanak sağlanabilmekte ve asıl pazarlamak istenen bilgiye (örneğin yayının/kitabın tamamına) erişim ise kısıtlanmaktadır.

```
class PublicationAgvent extends Agvent
{       private String Author;
        private String Name;
        private int             Pages;
        private float[] Dimensions ;
        private String Publisher;
        private Date   Publish ;
        private int             ISBN;
        private float   ListPrice;

        private boolean         ReferenceContains(String AuthorName) {…}
        private boolean         TOCContains(String topic) {…}
        private float   WholesalePrice(int amount, String destination) {…}
        …… }
```

Şekil 5.10 : Bir Agvent Altsınıfı Örneği

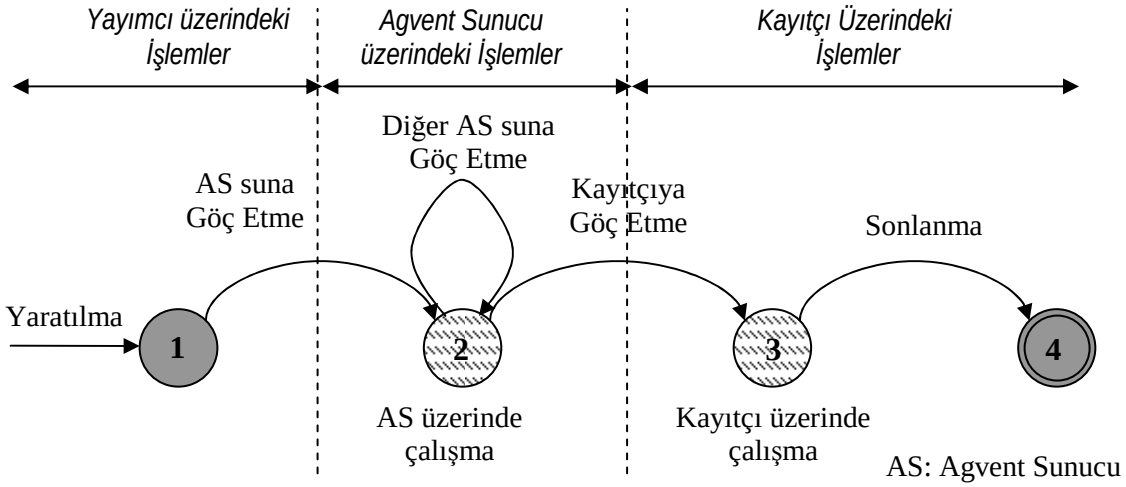
Eğer bu agvent için (PublicationAgvent) duyuru işlemi daha önce yapılmış ise aynı tipteki her ürün için tekrar yapılmasına gerek yoktur. Yapılması durumunda ABDES

Sistemi otomatik olarak bu duyuru talebini dikkate almayacaktır. Duyuru talebi yapıldıktan belirli bir süre sonra yayınevi kitabın detay bilgilerini içeren (gerekiyorsa kitabın hepsi de konulabilir) PublicationAgvent sınıfının örneği olan agventi oluşturup sisteme yayımlayarak dağıtım sürecini başlatır.

5.5.3. Agventin Yaşam Döngüsü

Bir agventin yaşam döngüsü Yayımcı düğümü üzerinde agventin yaratılması ve sisteme gönderilmesi ile başlar ve Kayıtçı düğümünde aktif hale gelerek üzerine yüklenen görevleri tamamlanması ile son bulur. Agventin farklı düğümlerde geçirdiği aşamalar *Agventin Yaşam Döngüsünü* olarak adlandırılır. Şekil 5.11'de nasıl işlediği gösterilen bir agventin yaşam döngüsü dört ana basamaktan oluşur:

1. Yayımcı tarafından yaratılması,
2. ASsu üzerinde sorgulama, denetim ve yönlendirme işlemlerinin yürütülmesi,
3. Kayıtçı üzerinde amaca yönelik çalışma,
4. Agventin kendi amacına ulaşmasından, görevini tamamlamasından sonra kendisini yok etmesi.



Şekil 5.11 : Bir Agventin Yaşam Döngüsü

Hangi agventi/agventleri yayımlayacağına ve bunları nasıl tanımlayacağına Yayımcılar karar verir. Bu nedenle bir agventin yaşam döngüsü bir olayın gözlenmesi ve bu olay sonucunda ilgili agventin yaratılması ile başlar (1nci basamak). Daha sonra Yayımcı oluşturduğu bu agventi doğrudan bağlı olduğu AS'suna gönderir. AS'su üzerinde kendine çalışması için ayrılan platformda aktif hale gelen agvent burada yürüttüğü sorgulama, denetim ve yönlendirme işlemleri (2nci basamak) sonucunda kendisini yönlendireceği düğümü/düğümleri seçer ve buralara göç eder. Hedef düğüm doğrudan Kayıtçı olabileceği gibi, agventin

Kayıtçıya ulaşacağı yol üzerinde yer alan bir AS'da olabilir. Agvent hedefi olan bir Kayıtçıya ulaştıktan sonra, orada kendisi için hazırlanan platformda aktif hale geçerek amacına uygun olarak çalışır (3ncü basamak). Görevi sona eren agvent kendisini yok ederek yaşam döngüsünü sonlandırır (4ncü basamak).

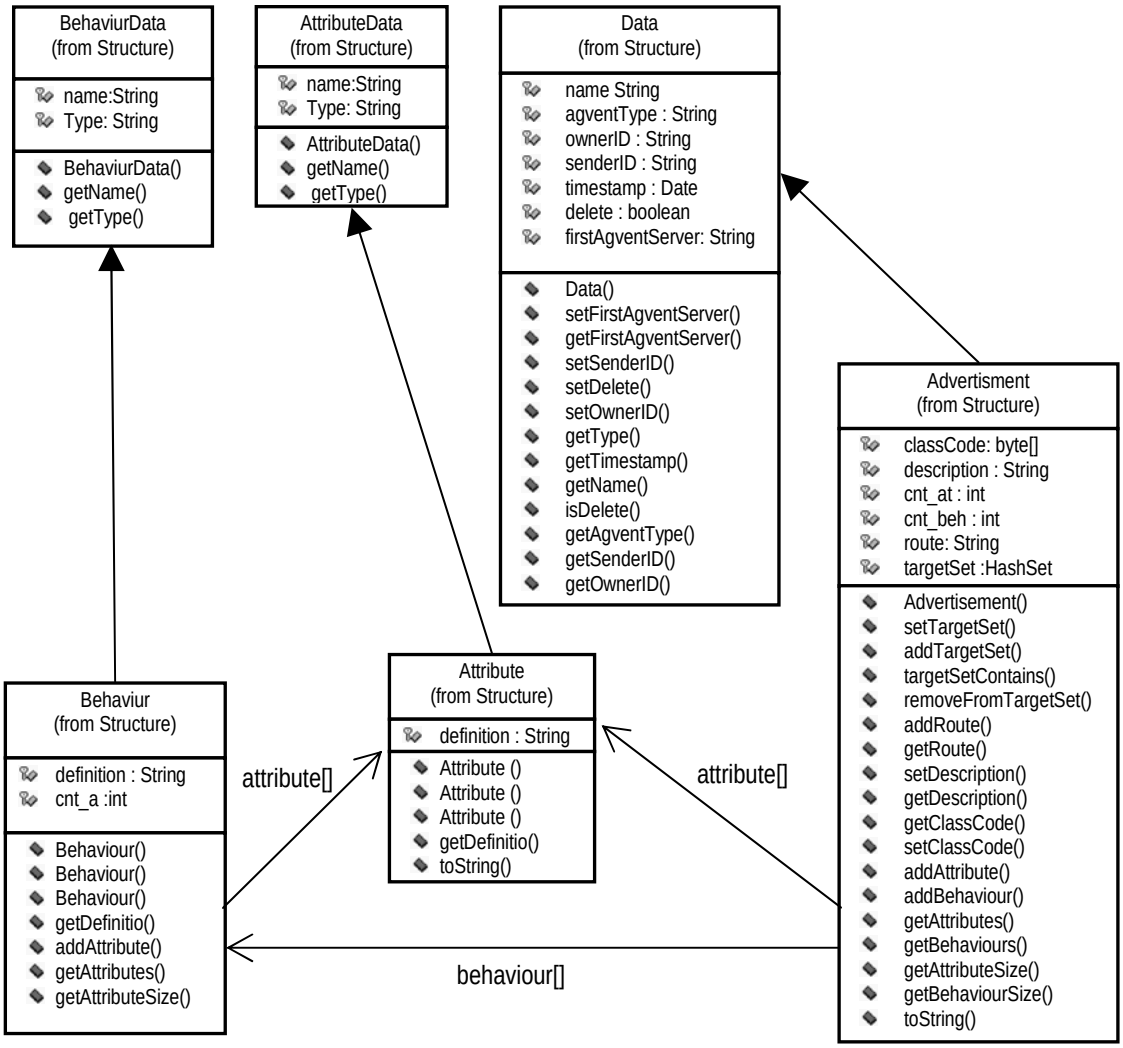
5.6. Duyurular

Duyuru çağrıları, AS ara yüzü üzerinde tanımlanmış olan iki adet metot çağrısıdır; *Duyuru_Yap (advertise)* ve *Duyuru_Sil (unadvertise).* Bir *Duyuru_Yap* çağrısı, yayımı yapılacak olan agvent tipinin sadece özelliklerini değil aynı zamanda da agventin davranışlarını da tanımlanır. Kayıtçının bu bilgiyi aldıktan sonra kayıt olacağı agvent tipi ile ilgili her türlü bilgiye sahip olması ve bu bilgiler ışığında kayıt olması amaçlanır. *Duyuru_Sil* çağrısı ise, yayımı yapılacağı daha önce sisteme ilan edilen bir agventin artık yayımlanmayacağının sisteme bildirilmesi amacı ile kullanılır.

Author	: String	"Tanımlama"	PublicationAgvent in filtrelenebilir özellikleri ve gerekli tanımlamalar
Name	: String	"Tanımlama"	
Pages	: int	"Tanımlama"	
Dimensions	: float[3]	"Tanımlama"	
Publisher	: String	"Tanımlama"	
Publish Date	: Date	"Tanımlama"	
ISBN	: int	"Tanımlama"	
ListPrice	: float	"Tanımlama"	
ReferenceContains(AuthorName)	: boolean (String)	"Tanımlama"	PublicationAgvent in filtrelenebilir davranışları ve gerekli tanımlamalar
TOCContains(topic)	: boolean (String)	"Tanımlama"	
WholesalePrice(amount,destination)	: float (int,String)	"Tanımlama"	

Şekil 5.12 : PublicationAgvent İçin Bir Oluşturulan Duyuru Mesajı Örneği

Bir önceki senaryoya bakılınca "PublicationAgvent" sınıfını geliştirdikten sonra yayınevi bu sınıfın özelliklerini içeren bir duyuru mesajı hazırlayarak bunu ABDES Sistemine duyurur. Her Yayımcı doğrudan bağlı bulunduğu ve oluşturduğu agventi onun üzerinden yayımlayacağı *Agvent Sunucusunun* adresini bilmek zorundadır. Şekil 5.12'de gösterilen örnek duyuru yapısını tanımlayabilmek için geliştirilen Duyuru (Advertisement) sınıfının UML diyagramı Şekil 5.13'te gösterilmektedir. Sisteme yayımlanacak olan bir agvent tipinin duyurusu, şekilde gösterilen sınıfın bir örneği (instance) olarak hazırlanarak sisteme gönderilir. ABDES Sistemi gelen bu duyuru mesajını sistemi oluşturan tüm AS'larına dağıtmaktan sorumludur.

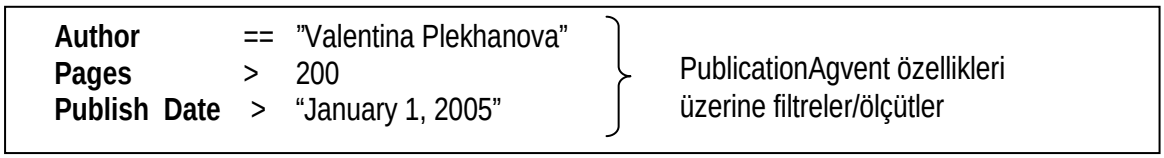


Şekil 5.13 : Duyuru Sınıfının UML Diyagramı

5.7. Kayıtlar

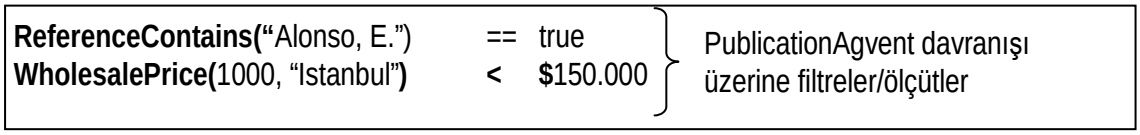
Kayıt isteği (kayıtlar) uygulamanın veya kullanıcının ilgi duyduğu olayları/agventleri sisteme bildirmek için kullandığı bir metot çağrısıdır. Kayıt çağrıları, AS ara yüzü üzerinde tanımlanmış olan iki adet metot çağrısıdır; *Kayıt_Ol (subscribe)* ve *Kayıt_Sil (unsubscribe).* Bir olayın belirlenen ölçütleri doğrulaması durumunda, bu olay neticesinde yaratılan agventin Kayıtçıya iletilmesinin isteği sisteme *Kayıt_Ol* çağrısı ile ulaştırılır. *Kayıt_Sil* çağrısı ise Kayıtçının ilgi duyduğu bir agvent ile ilgili artık haberdar edilmesini istememesi durumunda kullanılır. Bu kayıt ölçütleri, agvent üzerinde tanımlanan filtrelerle ifade edilirler ve bu filtreler iki türde tanımlanabilirler.

1. ***Özellik Filtresi ile Kayıt:*** Agventin özellikleri (attributes) üzerine doğrudan uygulanabilen filtrelerdir. Örneğin PublicationAgvent'in duyurusundan öğrenilen filtrelenebilen özelliklere göre yapılabilecek bir özellik filtresi örneği Şekil 5.14'te gösterilmiştir.



Şekil 5.14 : Özellik Filtresi ile Yapılan Bir Kayıt Örneği

2. ***Davranış Filtresi ile Kayıt:*** Agventin davranışları/metotları üzerine doğrudan uygulanabilen filtrelerdir. Örneğin PublicationAgvent in duyurusundan öğrenilen filtrelenebilen davranışlara göre yapılabilecek bir davranış filtresi örneği Şekil 5.15'te gösterilmiştir.



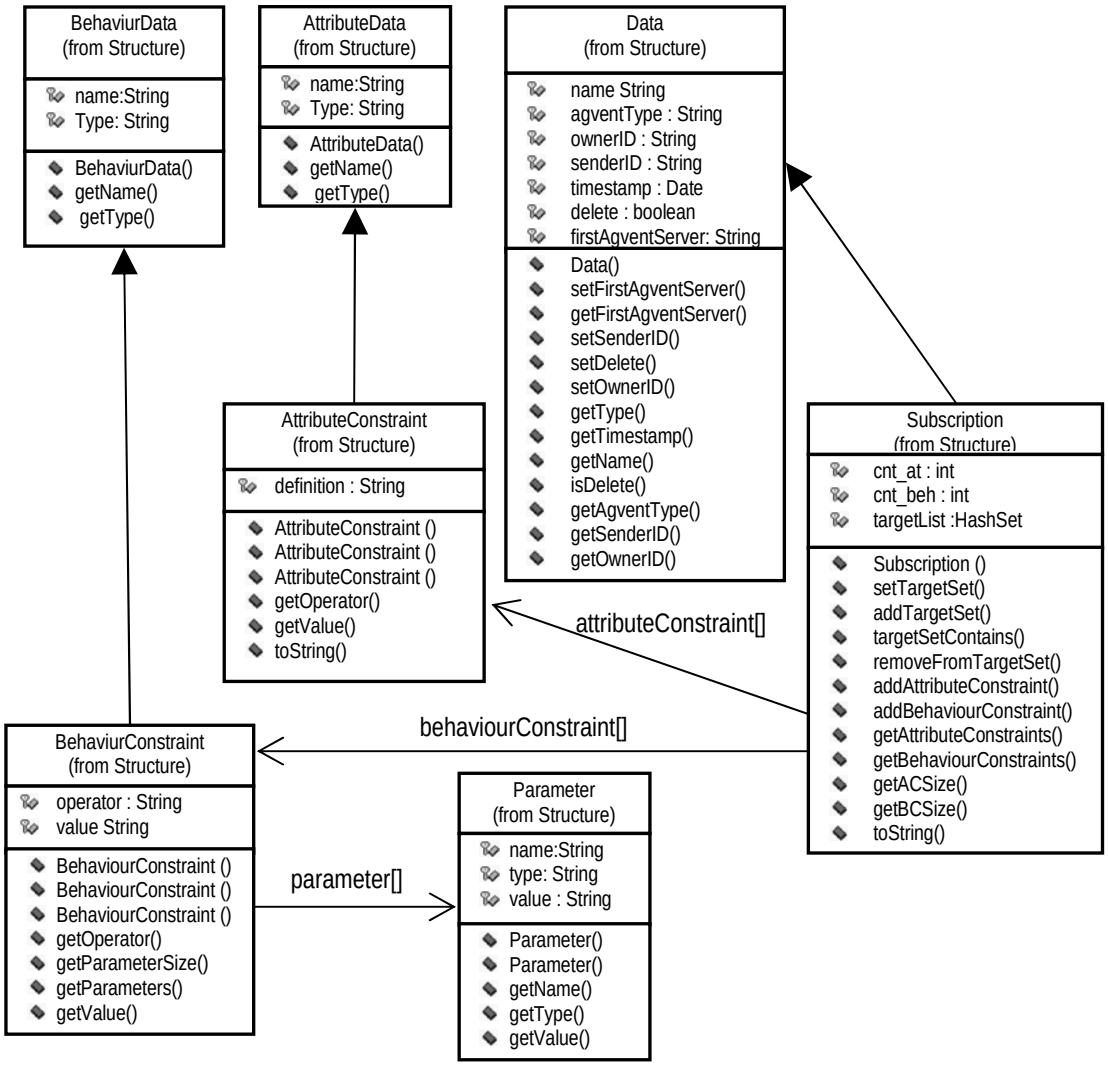
Şekil 5.15 : Davranış Filtresi ile Yapılan Bir Kayıt Örneği

Daha önceden duyurulan bir PublicationAgvent üzerine kayıt olmak için gönderilen bir kayıt mesajı, sistemde tanımlanmış bulunan Kayıt (Subscription) sınıfının bir örneği olması gerekmektedir. Kayıt sınıfının yapısı Şekil 5.16'da gösterilmiştir.

5.8. ABDES Sisteminde Filtreleme

Yukarıda da bahsedildiği gibi, ABDES Sistemini diğer dağıtılmış olay sistemlerinden ayıran temel özelliklerden biri de kullanmış olduğu filtreleme mekanizmasıdır. Dağıtılmış Olay Sistemlerinin bir amacı da, sisteme gönderilen olay verisini, sadece bu olay verisi ile ilgilenen olay tüketicilerine ulaştırılmasını sağlayarak sistemdeki veri akışını en düşük seviyede tutulmasını sağlamaktır. Bunun için de filtrelemeye olanak sağlayan ölçütler/profiller oluşturularak sisteme gönderilmektedir.

Günümüze kadar geliştirilen olay sistemleri, olay verisinin dağıtımında sadece tüketicinin ölçüt belirlemesine olanak sağlamaktadır. ABDES Sisteminde ise, kullanmış olan gezgin etmen yapısından ve agvente yüklenilen özerk özelliklerden dolayı olay verisini (agventi) üreten Yayımcının da agvente belirli ölçütler yüklemesi imkanı vardır. Bu nedenle ABDES Sistemi hem Kayıtçının hem de Yayımcının filtreleme yapmasına olanak sağlamaktadır. Bu filtreleme modelleri aşağıda açıklanmıştır.



Şekil 5.16 : Kayıt Sınıfının UML Diyagramı

5.8.1. Kayıtçının Filtrelemesi

Kayıtçı, bir agvent ile ilgilendiği zaman, ilgilendiği agventin özelliklerini bir kayıt mesajı ile ABDES Sistemine bildirir. ABDES Sistemi gönderilen bu kayıt mesajını, daha önceden sisteme ilan edilen duyuruların oluşturmuş olduğu dağıtım ağaç yapılarına göre AS'larına dağıtılmalarını sağlar.

Bir agvent, sisteme yayımlandığı zaman, ulaşmış olduğu ilk AS'da, kendisi ile ilgilendiğini beyan eden Kayıtçıların bu ölçütlerini inceleyerek, filtreleme ölçütlerine uyan Kayıtçıları belirler ve kendisini bu Kayıtçılara doğru yönlendirir.

5.8.2. Yayımcının Filtrelemesi

Bir Yayımcı ürettiği agvent üzerine, yönlendirileceği Kayıtçıların seçimi için iki şekilde filtre koyabilmektedir.

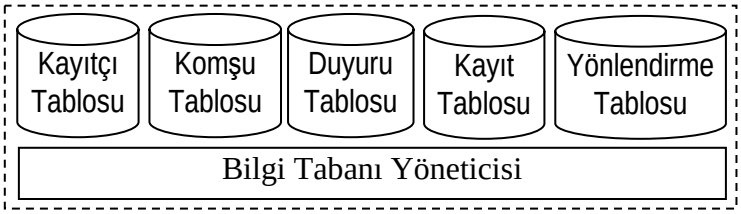
- *Kayıtçının tanımlayıcısı (adı veya adresi) üzerinden filtre:* Yayımcı daha önce dağıtmış olduğu agventlerle ilgili verileri saklamış olduğu geçmiş tecrübelerini içeren bir bilgi tabanı vardır. Yayımcı bu bilgi tabanından faydalanarak belirli Kayıtçılara bu agventin ulaştırılmasını istemiyorsa, bu Kayıtçıları bir yasak listesine koyarak agvente tanımlarlar. Agvent kendi yönlendirmesini yaparken, Kayıtçının ölçütlerini sağlamış olsa dahi kendisini bu listedeki Kayıtçılara doğru yönlendirmez. Agventin yapmış olduğu bu filtreleme, ulaşmış olduğu ilk AS'su üzerinde yapılır.
- *Kayıtçının özellikleri üzerinden yapılan filtre:* Her Kayıtçının bağlı olduğu AS'su üzerinde kendisi ile ilgili "çalışan sayısı, net gelir, kısa vadeli borç miktarı, uzun vadeli borç miktarı, toplam mal varlığı, yatırım miktarı, aktiflerinin değeri, pasiflerini değeri" gibi istatistiki bilgileri tutulmakla birlikte aynı zamanda geliştirilen agventlerin de yayımlanmasında faydalanılmak üzere Kayıtçının bilgisayarının performansını gösteren "CPU hızı, bellek miktarı, hard diskteki boş alan" gibi değişik özellikleri de saklanabilmektedir[27]. Bu bilgiler Kayıtçının özel bilgileri olduğu için Kayıtçının, bir agventle ilgili kayıt mesajının sistemde dağıtıldığı gibi, tüm AS'lara dağıtılmaz, sadece bağlı bulunduğu AS üzerinde tutulur.

  Yayımcı, Kayıtçının bu özelliklerini de kontrol ederek bir filtreleme yapmak istemesi durumunda, kendi profilini tanımlayarak bir katar olarak agventin içine yerleştirir. Agvent Kayıtçının bağlı bulunduğu AS'ya ulaşınca, bu ölçütleri AS'nun bilgi tabanından kontrol eder ve kendisini bu Kayıtçıya gönderip göndermemeye karar verir.

[27] Bu özellikler sistemde yayımlanabilecek agventlerin kullanabilecekleri veriler belirlenerek daha da genişletilebilir.

6. ABDES SİSTEMİNDE BİLGİ TABANI YÖNETİMİ

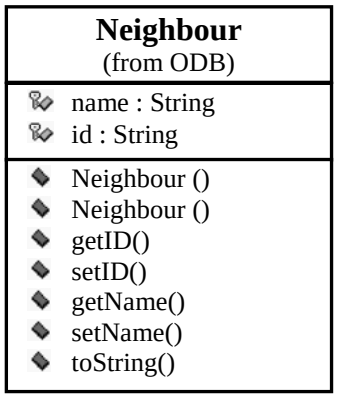
ABDES Sisteminde, Agvent Sunucuları (AS) üzerinde kayıt bilgilerini, duyuru bilgilerini, komşu AS bilgilerini ve Kayıtçı bilgilerini saklamak için Şekil 6.1'de görüldüğü gibi, beş ana tablonun bulunduğu bir bilgi tabanı yönetilmektedir. Bu verilere erişimin kontrollü ve koordineli bir şekilde yapılabilmesi, sistemindeki veri bütünlüğünün korunabilmesi için bu veri tabanlarına erişim bir bilgi tabanı yöneticisi üzerinden sağlanmaktadır.



Şekil 6.1 : Agvent Sunucusu Üzerinde Veri Tabanları

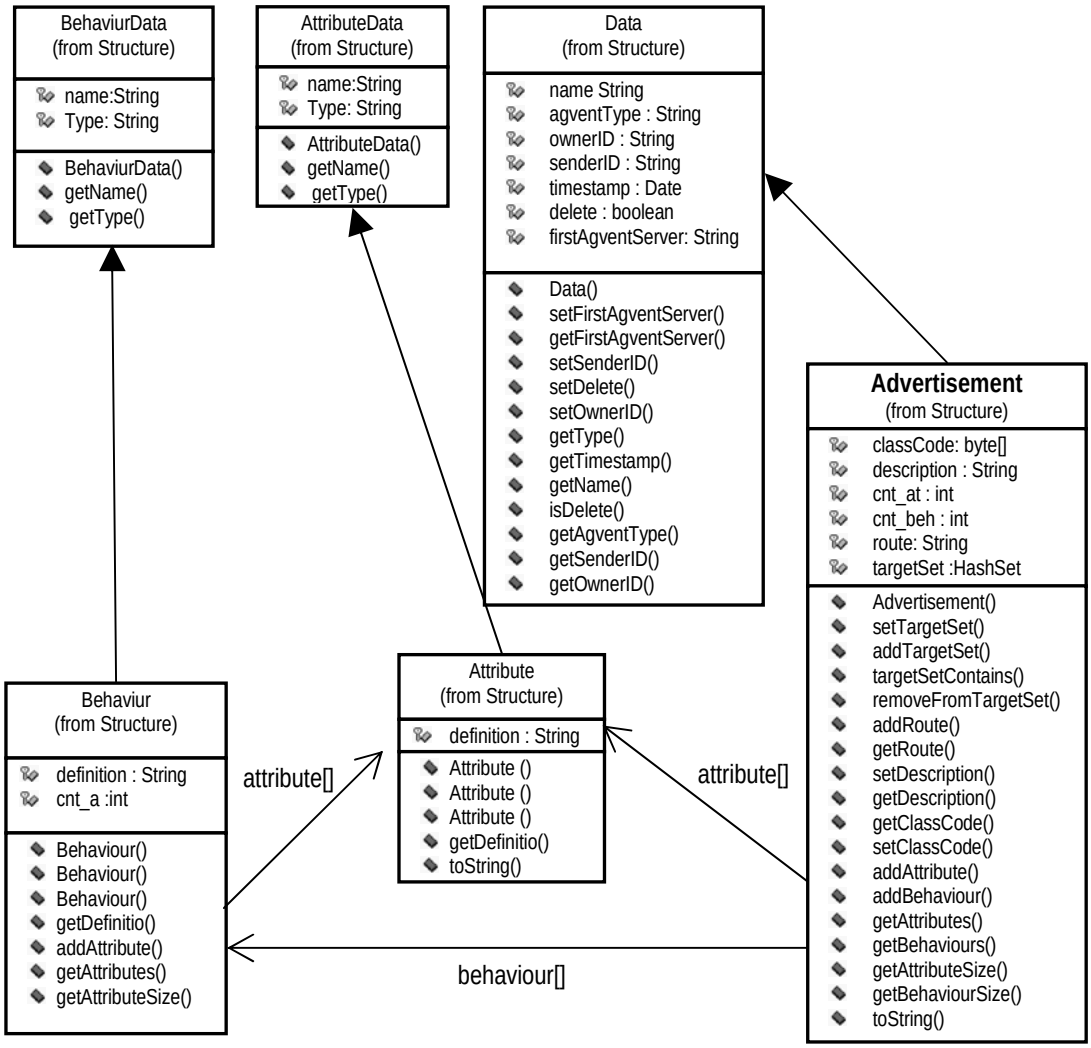
Kullanılan tabloların özellikleri ve içerdikleri nesnelerin yapıları şu şekilde tanımlanabilir:

1. ***Komşu Tablosu:*** Bir AS'sunun doğrudan bağlı olduğu diğer komşu AS'larına ait ad, adres, vs, gibi bilgilerin yer aldığı veri tabanı tablosudur. Bu tablo özellikle kayıt ve duyuru mesajlarının dağıtımında kullanılır. Bu tablonun içerdiği Komşu (Neighbour) sınıfının UML diyagramı Şekil 6.2'de gösterilmiştir.



Şekil 6.2 : Komşu Sınıfının UML Diyagramı

2. ***Duyuru Tablosu:*** Sistemde yayımlanan agventlere ilişkin tanımlayıcı bilgilerini tutan tablodur. Bu bilgiler agventin filtrelenebilir özelliklerinin yanı sıra, filtrelenebilir davranışları da içerir. Kayıtçı bu bilgilere dayanarak kendi belirlediği koşullarda agventlerin ulaşmasını sağlayacak ölçütleri hazırlar. Bu tablonun içerdiği Duyuru (Advertisement) sınıfının UML diyagramı Şekil 6.3'te gösterilmiştir.



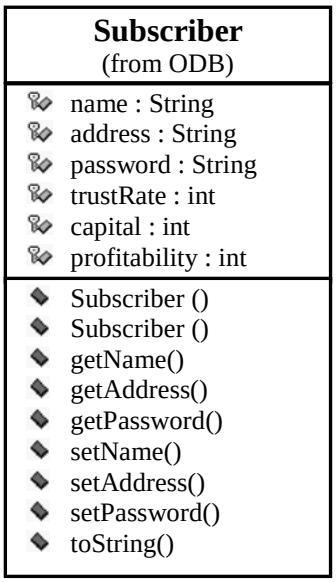
Şekil 6.3 : Duyuru Sınıfının UML Diyagramı

3. ***Kayıtçı Tablosu:*** Bu tablo Kayıtçılara ait tanımlayıcı, şifre, adres vs. gibi belirleyici bilgilerin haricindeki

    - bir şirkete ait "çalışan sayısı, net gelir, kısa vadeli borç miktarı, uzun vadeli borç miktarı, toplam mal varlığı, yatırım miktarı, aktiflerinin değeri, pasiflerini değeri" bilgileri ile,

- Kayıtçının bilgisayarının performansını gösteren CPU hızı, bellek miktarı, hard diskteki boş alan gibi bilgiler tutulmaktadır.

Bu tablonun[28] içerdiği Kayıtçı (Subscriber) sınıfının UML diyagramı Şekil 6.4'te gösterilmiştir.



Şekil 6.4 : Kayıtçı Sınıfının UML Diyagramı

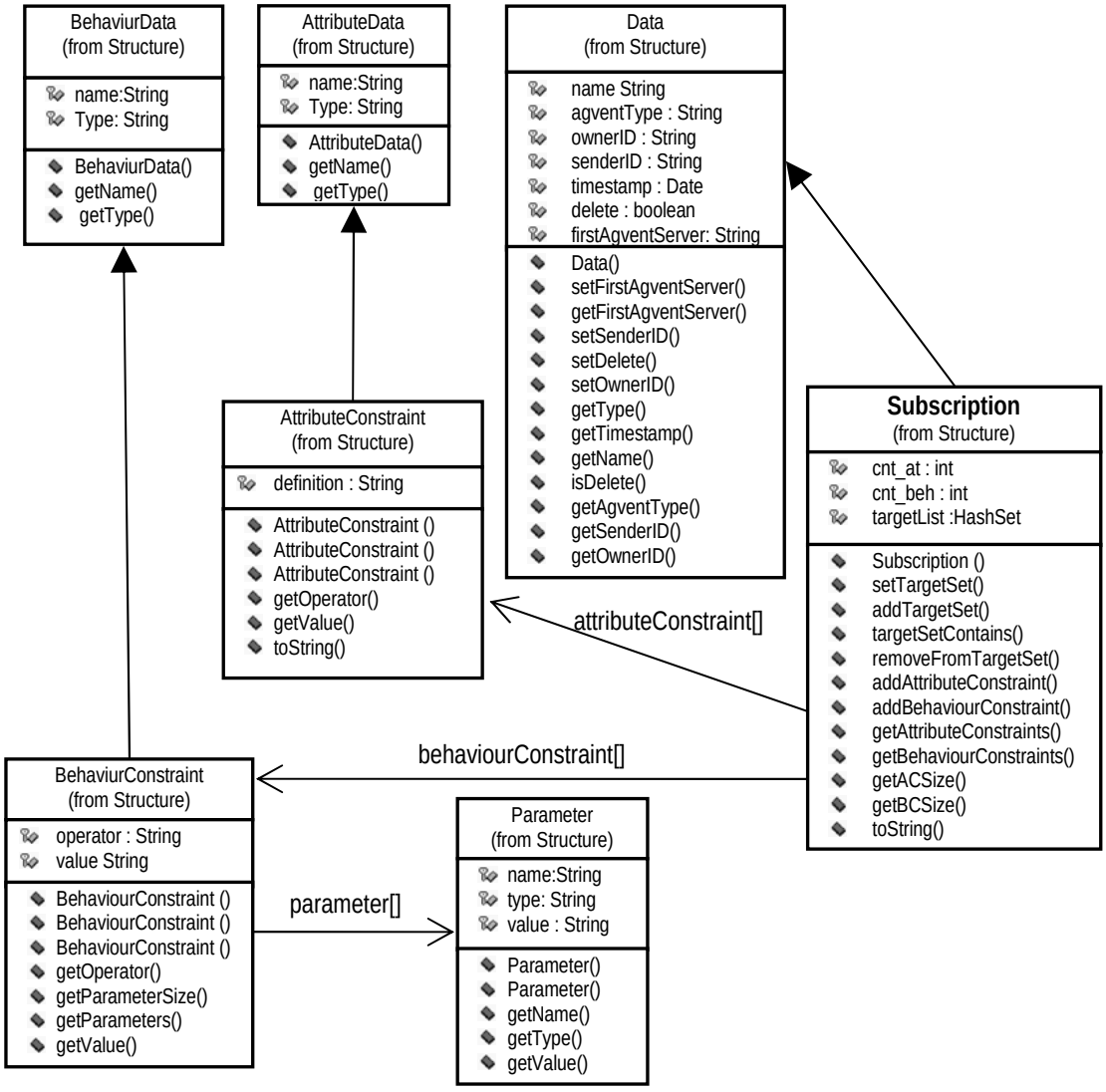
4. ***Kayıt Tablosu:*** En sık erişilen veri yapısıdır. Tabloda Kayıtçıların sisteme kayıt olurken bildirdikleri ölçütler yer alır. AS'suna ulaşan bir agvent, kendisini ilgilendiren bilgileri değerlendirerek tanımlı ölçütlerin doğrulandığı takdirde, kendini hedef düğümlere (ilgili AS'na ve Kayıtçılara) yönlendirme işlemini yürütür. Bu tablonun içerdiği Kayıt (Subscription) sınıfının UML diyagramı Şekil 6.5'te gösterilmiştir.
5. ***Yönlendirme Tablosu:*** Duyuru-Kayıt mesajları ile agventlerin yönlendirilmelerinde bir hata oluşması durumunda alternatif bir rota üzerinden yönlendirme işleminin sonuçlandırılmasını sağlamak için kullanılır.

Bu veri tabanlarına kayıt ekleme, silme ve sorgulama işlemlerinin hızlı bir şekilde yapılabilmesi, sistemin etkin ve verimli bir şekilde çalışabilmesi için çok önemlidir.

ABDES Sisteminde, sistem bileşenleri arasında veri alışverişleri nesneler vasıtasıyla olmaktadır. Tanımlanan *Duyuru, Kayıt, Kayıtçı ve Komşu* sınıfları bu amaç için tanımlanmış sınıf yapılarıdır. Şekil 6.2, Şekil 6.3, Şekil 6.4 ve Şekil 6.5'ten de görüleceği üzere bu sınıf yapılarının bir kalıcı dosya yapısına saklanması ve buradan

[28] Bu tablonun oluşturulması ve komşu agvent sunucuların verilerinin eklenmesi ABDES Sistem Yöneticisi tarafından sağlanmaktadır.

hızlı bir şekilde erişilmesi bu şekilde saklama işlemlerinde kullanılan genel teknikler ile oldukça zor olacaktır. Çünkü burada kullanılan ana veriler birer nesne olup farklı tasarım metodolojileri içermektedir. Bu nedenle, veri tabanı tasarımı ve geliştirilmesi yapılırken kullanılan nesnelerin yapısının dikkate alınması gerekmektedir. Nesnelerin kalıcı hale getirilmesi ile ilgili yapılan çalışmalar ve tez çalışmasında kullanılan Java Veri Nesnesi (Java Data Object) tekniği konusunu anlatmadan önce nesnelerin bilgisayar üzerinde kalıcılığını sağlamak amacıyla kullanılan diğer mekanizmaları ve bunların avantaj ve dezavantajlarını incelemekte fayda vardır.



Şekil 6.5 : Kayıt Sınıfının UML Diyagramı

6.1. Kalıcı Nesneler

Yıllardır tamamen nesneye dayalı olan Java uygulamalarında, verilerin dosyalara saklanması ve yeniden dosyadan yüklenmesi işlemlerinde ciddi problemlerle karşılaşılmıştır. Bunda ana sebep olarak:

- Saklanacak verilerin nesneler olarak tanımlanmış olması,
- Ara yüzler (interface) ve kalıtımsallıkların (inheritance) kullanılması sebebi ile nesne yapısının karmaşıklığı,
- Nesnelerin doğrudan bir veri tabanına transfer edilmesinin mümkün olmaması,
- Özel veri tiplerinin saklanmasında karşılaşılan zorluklar gösterilmektedir.

Kalıcı Nesne (Persistent Object) kavramında ortaya konan yeni standartlar neticesinde bu nesnelerin kalıcı dosyalara saklanması, sorgulanması ve yeniden yüklenmesi işlemleri kolaylaşmıştır.

Bir çok nesneye yönelik programlama dilinde, kendi durumları ve davranışları olan nesneler birer sınıfın örneğidirler. Uygulamayı oluşturan sınıflar farklı fonksiyonellikler göstermekte olup, bu sınıfların tanımlamalarının tamamı, uygulamanın nesne modelini oluşturmaktadır. Fakat, her nesne modelinde, bir dizi nesne doğrudan uygulama verilerinin soyutlamasını içeren bir veri yapısı şeklinde olup, bu nesneler uygulamanın çalışması için gerekli olan öncelikli durum ve davranışları içermektedir[29]. Bu nesneler sistemin çalışmasında temel olan farklı modüller arasında, veri aktarımı için kullanılan bilgileri içermektedirler ve bilgilerin kalıcı olarak bir veri deposunda/tabanında saklanması gerekmektedir.

Son yıllarda nesneye yönelik bir programlama dili olarak yoğunlukla kullanılan Java programlama dilinde de, geliştirilen nesnelerin Java Sanal Makinesinin dışında da saklanabilmesi ve daha sonra bu nesnelere erişilebilmesi önem arz etmektedir. Nesnelerin bu şekilde ikincil bellekte kalıcı olarak saklanabilmelerine nesnelerin kalıcılığını sağlamaktadır. Bu kalıcılığın sağlanması için değişik teknikler kullanılmaktadır.

Kalıcılık, bir nesnenin durum değişkenlerinin daha sonra kullanılmak üzere saklanmasını gerektirmektedir. Piyasada bu işlevi gerçekleyen farklı mekanizmalar kullanılmaktadır. Java programlama dilinde bu mekanizmalardan en yaygını olarak kullanılan teknik olarak Java Database Connectivity (JDBC) ve Structured Query

[29] Örneğin bir Tedarik Zinciri Yönetiminde (Supply Chain Management) *Müşteri*, *Sipariş* ve *Ürün Nesneleri* veya bir Finanssal uygulamadaki *Müşteri*, *Hesap*, *Kredi Bilgisi* ve *Borç Bilgisi* nesneleri gibi.

Language (SQL) bileşimi bir kombinasyonla erişilen ve İlişkisel Veri Tabanı Yönetim Sistemleri (Relational Database Management System-RDBMS) gösterilebilir.

6.1.1. İlişkisel Veritabanları

RDBMS teknolojisi, sahip olduğu verilerin satırlar ve sütunlar şeklinde serbest tanımlanmasına olanak sağlayan yapısı, özel sorgulama dili ve *begin*, *rollback*, ve *commit* gibi işlemler ile sağlanan işlem güvenirliliği sayesinde son 15 yılda hızla yayılmış ve her alanda kullanılmaya başlanmıştır. Aynı zamanda veri tabanı erişiminde standart bir dil kullanılması (SQL) ve veri alışverişlerinin bunun üzerinden yapılması ile bir standartlaşma sağlanmış ve bunun sonucu olarak RDBMS'lerin yayılmasını hızlandırmıştır.

JDBC, Java sınıflarının ilişkisel veritabanları ile haberleşmeleri için kullanılan bir alt seviye teknolojidir. Kalıcı verilerini saklamak için ilişkisel veritabanlarını kullanan Java uygulamaları bir SQL komutunu veri tabanı sunucusuna JDBC API si üzerinden göndermektedir. SQL komutları birer *String* nesnesi olarak oluşturulmakta ve daha sonra derlenmesi ve çalıştırılması için veri tabanı sunucusuna gönderilmektedir. Sorgulama sonucunda elde edilen veri, çok sayıda satır ve sütundan oluşan bir *ResultSet* formunda aşağıdaki kodlamada görüldüğü gibi, programa dönmektedir.

```
String sorgu = "SELECT * FROM OGRENCI WHERE NOT_ORT > 3";
try{
Statement stmt = con.createStatement();
ResultSet rs = stmt.executeQuery( sorgu );
while(rs.next()) {
String str = rs.getString(1) + " " + rs.getString(2) + " " +  rs.getString(3) +
        " " + rs.getInt(4) + " " + rs.getInt(5);
System.out.println(str);  }
stmt.close();  }
catch (java.sql.SQLException exc){
        System.err.println("--------- SQL EXCEPTION ---------");
        exc.printStackTrace(System.out);
        }
```

Her ne kadar nesnelerdeki bilgilerin kalıcı hale getirilmesinde RDBMSlerden faydalanılması sık kullanılan bir durumsa da bazı ciddi zorluklarla karşılaşılmaktadır:

- Sadece JDBC uyumlu veri tabanları ile kullanılabilmektedir (Microsoft® SQL Server, Sybase™, Oracle®, ve MySQL® gibi).
- Projede çalışan en iyi/tecrübeli kişiler zamanlarını saklanacak verinin kalıcı hale getirilmesi çalışmalarında harcamaktadırlar.
- Program geliştirici SQL dilini bilmeli ve veri işleme işlemlerinin hepsinde bunu uygun şekilde kullanmalıdır.

- Program geliştirici, nesne özelliklerini bir veya daha fazla tablo üzerinde haritalandırması gerekmektedir. Bu haritalandırmada ise değişik uyuşmazlıklar çıkabilmektedir.
- Nesne yapısının biraz karmaşık halinde (ara yüz kullanımı, kalıtımsallık) ise bu haritalandırma içinden çıkılmaz haller alabilmektedir.
- Oluşturulan veri tabanının taşınabilirliğinde karşılaşılan ciddi eksilikler bulunmaktadır. Örneğin, Oracle ile hazırladığınız bir veri tabanını başka yere taşıdığınızda yine Oracle üzerine kurmanız gerekmektedir.
- Veri Tabanınızı değiştirmeniz durumunda (Oracle'dan Sybase'e geçilmesi gibi) program içinde kullanılan SQL komutlarının da bazılarının yeniden yazılmaları gerekmektedir.
- SQL komutları bir *String* nesnesi olarak gönderildiği için, program kodunun derlenmesinde bu string ifade derlenmemektedir. SQL komutunun derlenmesi veri tabanının görevi olarak kalır ve bu da hatalar ve kırılmalar çıkarabilmektedir.

6.1.2. Dosya Sistemleri

Dosya sistemleri de kalıcı nesnelerin saklanmasında kullanılan basit çözümlerden biri olan nesnelerin metin tipi ya da rasgele erişimli dosyalar içinde saklanmasını ifade etmektedir. Java'nın nesne serileştirilmesi kolaylıklarından faydalanılmaktadır ve bunun içinde saklanacak olan verinin "*Serializable*" ara yüzünü kullanıyor olması gerekmektedir. Bunun neticesinde sorgulama kolaylıklarından faydalanılamamakta ve veri erişim süresi artmaktadır.

Sistemin temel avantajı ise işletim sisteminin ötesinde ciddi bir destek gerektirmemesidir. Bu sayede kolaylıkla taşınabilmekte ve başka bilgisayarlarda kullanılabilmektedir. Büyük veri saklama işlemlerinde nadiren kullanılmakta olup genellikle veri setinin tamamının bellekte saklanacağı durumlarda aşağıdakine benzer bir kodlama ile faydalanılmaktadır.

```
try {
 File data = jFileChooser1.getSelectedFile();
 FileOutputStream fos = new FileOutputStream(data);
 ObjectOutputStream oos = new ObjectOutputStream(fos);
 Advertisement adv = new Advertisement("test",this.jTextField1.getText());
 for (Enumeration en = attributes.keys(); en.hasMoreElements(); ) {
  Attribute tra = (Attribute) attributes.get(en.nextElement());
  adv.addAttribute(tra);
 }
 for (Enumeration en = behaviours.keys(); en.hasMoreElements(); ) {
  Behaviour trab = (Behaviour) behaviours.get(en.nextElement());
  adv.addBehaviour(trab);
```

```
 }
 oos.writeObject(adv);
 oos.close(); }
catch (Exception ex) { }
```

6.1.3. Nesne Veritabanları

Nesneye Yönelik Veritabanları aynı zamanda Nesne Veri Tabanı Yönetim Sistemleri (Object Database Management Systems-ODBMS) olarak adlandırılırlar. Nesne veritabanları integer, string, real, gibi basit verilerden ziyade nesneleri saklamak için tasarlanmışlardır. Nesneler ise nesneye yönelik diller olan Smalltalk, C++ ve Java gibi diller tarafından kullanılırlar. Nesneler şu iki bileşenden oluşurlar:

- ***Özellikler*** Özellikler nesnenin karakteristikleri belirleyen verilerdir. Bu veriler integer, string, float gibi basit veriler olabilecekleri gibi karmaşık nesne referanslarıda olabilirler.
- ***Davranışlar (Metotlar)*** Metotlar nesnenin prosedürler veya fonksiyonlar olarak ifade edilen davranışlarını tanımlarlar.

Bu nedenle nesneler hem veri içermektedir, hem de çalıştırılabilir program kodu içermektedir. Sınıflar ise nesnenin içereceği bu davranış ve özelliklerin tanımlandığı bir şablon veya kalıp olarak düşünülebilir Sınıflar verileri ve metotları içermez ancak onları tanımlarlar. Sınıflar nesnelerin yaratılmasında kullanılırlar. Sınıflar aynı şekilde ODBMS lerde de nesneyi tanımlamak için kullanılırlar. Veri tabanında nesnenin özellikleri saklanır, metotları ise saklanmaz. Metotların kullanılması gerekince bu sınıf tanımlamasından yeniden yaratılması gerekmektedir.

Nesne Veritabanları karmaşık veriler ve/veya çoklu (n-m relationship) nesne ilişkileri olduğu zamanlarda kullanılması uygundur. Nesne veri tabanları birkaç tablonun ve çok basit tipte verinin olduğu uygulamalarda kullanılması uygun değildir. Elektronik Ticaret uygulamalarında, Çoğulortam uygulamalarında, zamanla değişen nesne yapılarının olduğu projelerde kullanılması uygundur.

- Geçici (ad hoc) sorgulama kapasitesinin eksikliği,
- ODBMS sistemlerin net bir standardının olmaması,
- Oluşturulan veri tabanının, oluşturulduğu ODBMS'e bağımlı olması ve taşınabilirliğinin eksikliği

ciddi bir eksiklik olarak karşımıza çıkmaktadır. Object Data Management Group, nesne veritabanlarında veri işleme için bazı standartlar koysa da bu ODBMS sistemlerinin gelişimini çok az bir şekilde artırmıştır.

6.1.4. Varlık Çekirdekleri

Varlık Çekirdekleri (Entity Beans), Java 2 Platform, Enterprise Edition, Enterprise JavaBeans (J2EE EJB)'nin bir parçası olup çok sayıdaki eş zamanlı uzak/yakın kullanıcının ortak erişebileceği kalıcı verilerin ifade edilmesine olanak sağlayan standartları ve kullanım metotlarını sağlamaktadır.

İki çeşit EJB bulunmaktadır. Bunlar; "Varlık Çekirdekleri " ve "Oturum Çekirdekleri (Session Beans)". (Bir de "Mesaj Tabanlı Çekirdekler (Message-Driven Bean)" ler vardır ancak bu JMS kullanımını gerektirmektedir.) Bunlardan kalıcı olanlar Varlık Çekirdekleri olduğu için bizim için incelenmesi gerekende budur.

Varlık çekirdekler normal nesnelerden farklı özellikleri olan nesnelerdir. Varlık Çekirdeklerinin özelliklerini şu şekilde sıralanabilir:

- ***Kalıcıdır.*** Standart Java nesneleri, program içinde yaratıldıkları zaman var olabilirler. Program sona erdiği zaman ise nesne yok olur. Fakat bir varlık çekirdeği silinene kadar hayatta kalabilir. Bir program bir varlık çekirdeği yaratabilir, durdurabilir ve yeniden başlatabilir. Program kapatıldıktan sonra da varlık çekirdeği var olmaya devam eder. Program yeniden çalıştırıldığında ise beraber çalışmış olduğu varlık çekirdeğini bularak beraber çalışmaya devam eder.
- ***Ağ tabanlıdır.*** Standart Java nesneleri genellikle bir program tarafından kullanılabilirler. Fakat varlık çekirdeği ağ üzerindeki herhangi bir program tarafından kullanılabilirler. Programın bir varlık çekirdeğini kullanabilmesi için onu ağ üzerinde bulabilmesi (adreslemesi) yeterlidir.
- ***Uzaktan çalıştırılabilir.*** Bir varlık çekirdeğinin metodu bir sunucu üzerinde çalıştırılabilir. Kendi programınızda bir varlık çekirdeğinin metodunu çağırdığınız zaman kendi programınızın çalışımı durur ve kontrol sunucuya geçer, sunucudaki metodun çalışımı bitince kontrol tekrar sizin programınıza döner.
- ***Bir tekil anahtar (primary key) tarafından tanımlanır***. Varlık çekirdeğinin birer tekil tanımlayıcıları olmalıdır. Örneğin bir "çalışan" entity beani "TC Kimlik No" yu tekil anahtar olarak kabul edebilir. Varlık çekirdeği sadece böyle bir alanı olduğu zaman kullanabilirsiniz.

Her ne kadar bu erişim metodolojisi yaygın olarak kullanılsa da bazı tasarım kusurları, program geliştiricileri engellemektedir. Bu kusurların büyük bir kısmı EJB 2.0 ile ortadan kaldırılsa da varlık çekirdeğinin dezavantajlarını şöyle sıralanabilir:

- Yerel ve uzak veri erişimlerindeki semantik farklılık bulunmaktadır.

- İnce taneli nesnelerin kullanımı için dahi ağır bir bileşen mekanizmasının kullanılmasını zorlamaktadır.
- Kullanımı nispeten karmaşıktır.
- İyi bir performans sağlamak zordur.
- Kalıtımsallık desteklenmemektedir.
- Uygulama sunucusu (application server) olmayan ortamlarda desteklenmemektedir.
- Dinamik sorgulama mekanizmasını desteklememektedir. Önceden elle yazılmış olan Enterprise JavaBean Query Language (EJBQL) sorgulamaları kullanılmakta ve sorguların derleme zamanında hazırlanmış olmaları gereklidir.
- Sadece ilişkisel veritabanları desteklenmektedir.
- Otomatik birincil anahtar (primary key) üretimi yoktur.
- Altprogram testlerinin yapılması ve bu altprogramlarında uygulama dışında kullanılması zor ve karmaşıktır.

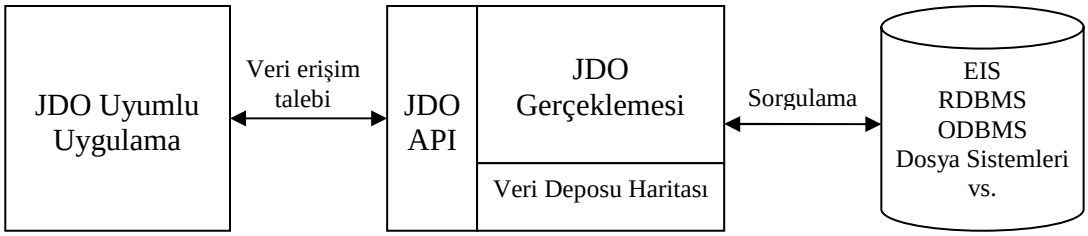
Bu bilinen ve sık kullanılan tekniklerin ötesinde Sun Microsystems tarafından şartnamesi ve ölçütleri belirlenen Java Veri Nesneleri (Java Data Objects-JDO) yeni bir kavram olarak karşımıza çıkmaktadır. Bir sonraki konu JDO'nun özelliklerini içermektedir.

6.2. Java Veri Nesneleri

JDO şartnamesi (specification) Sun Microsistemlerinde çalışan Craig Russell tarafından Java topluluk çalışmaları (Java Community Process) sırasında JSR-000012 kodu ile geliştirilmiştir. Bu şartname üzerine çalışmalar ilk olarak 1999 yılında başlamış ve 1.0 sürümü 2002 mayısında ortaya çıkmıştır. Güncellemesi JDO 1.0.1 olarak 2003 eylülünde yapılmıştır.

JDO Java nesneleri için üzerine kurulduğu veri deposundan bağımsız API si sayesinde uygulama geliştiriciler için saydam bir kalıcılık sağlamaktadır. Kullanılması gereken herhangi bir ara yüz (interface) olmayıp, kullandığımız standart Java sınıfları ile veri saklamaya olanak sağlamaktadır.

Java Veri Nesneleri, Java dili için tanımlanmış olan, saklama, sorgulama, ve nesnelerin veri deposundan alınmasını sağlayan ara yüz tabanlı kalıcı nesne tanımlamasıdır.JDO uyumlu bir uygulamanın JDO API si üzerinden bir veri deposuna erişimi diyagramı Şekil 6.6'da gösterilmiştir.



Şekil 6.6 : JDO Uyumlu Bir Uygulamanın Bir Veri Deposuna Erişimi Diyagramı

JDO'nun kullanıldığı sistemlerde şu saydamlıkları bulunmaktadır:

- JDO saydam bir şekilde üzerinde bulunduğu veri deposundaki JDO örneklerinin haritalarını tutmaktadır.
- JDO kalıcı hale getirilecek olan Java Nesneleri içinde saydamdır. Diğer bir deyişle bu nesnelerin aktif şekilde kullanılabilmesi için özel bir metot veya özellik eklenmesine gerek yoktur.
- JDO ilişkisel veritabanı, nesne veri tabanları, dosya sistemleri ve XML dosyaları gibi farklı veri depolama paradigmalarına karşılık olarak kullanılabilir.
- JDO içerdiği veri deposuna göre de saydamdır. Bu sayede uygulamalar kolaylıkla taşınabilmektedir.

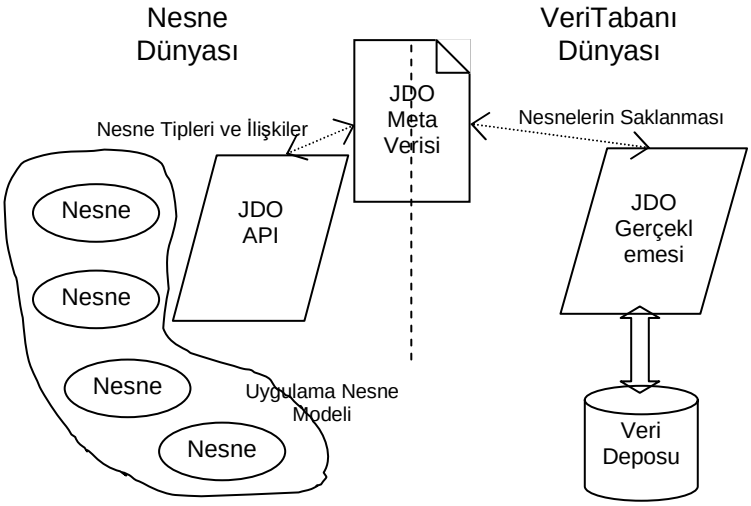
Tablo 6.1. JDO'nun Diğer Tekniklerle Karşılaştırılması

	Serialization	JDBC	ODBMS	EJB	JDO
Hareketsellik (Transactional)	X	√	√	√	√
Sorgulama Kolaylıkları	X	√	√	√	√
Standart API	java.io	JDBC	ODMG	EJB	JDO
Standart Sorgulama Dili	X	SQL	OQL	EJBQL	JDOQL
Desteklenen Veri Depolama Yapısı	Dosya Sistemi	RDBMS	ODBMS	RDBMS EAI	RDBMS ODBMS EAI Dosya Sistemi
Kalıcı Örneklerin Kaplanması (closure)	X	X	√	X	√
Alan Modeline (Domain Model) Saydamlık	X	X	√	X	√
Gerçek Nesne Veritabanı	X	X	√	X	X
Mevcut Tablo Yapılarını destekleme	X	√	X	X	√[30]

JDO'nun daha önce bahsedilen nesnelerin kalıcı hale getirilmesi ile ilgili tekniklerle olan karşılaştırması Tablo 6.1'de gösterilmektedir. JDO kısıtlamaları farklı geliştirme ortamlarını kullanılan genel API'ler sayesinde desteklemektedirler. Aynı zamanda alt sistem olarak bir RDBMS kullanılması gerekmediği için PDA ve Cep telefonları gibi

[30] Her ne kadar bu özellik JDO şartnamesinde (teknik ayrıntılı dokümanında) standartlaştırılmamışsa da JDO satıcıları bunu gerçeklemişlerdir.

kısıtlı kaynağa sahip ortamlar içinde kullanımı uygundur. JDO destekli bir uygulama geliştirebilmek için programcıların Şekil 6.7'ye benzer şekilde bir veri erişim modeli gerçeklemeleri gerekmektedir. Programcılar öncelikle kalıcı servislerin ihtiyaç duyulacağı sınıfları oluştururlar (persistance-capable classes). Bu sınıflarda genellikle Uygulama Nesne Modelini (Domain Object Model) oluşturmaktadır. Sonra bir kalıcılık tanımlayıcısı olan bir XML dosyası oluşturulur. Bu metin dosya en basit manada kalıcı sınıfların isimlerini içermektedir. Bir geliştirme (enhancement) işlemi tetiklenerek XML dosyasında bulunan sınıfların JDO üzerinden erişimine uygun hale getirilmesi sağlanır (yeni metotların eklenmesi)[31]. JDO uygulamalarında, sadece kalıcı sınıfların nesneleri doğrudan veri tabanında saklanabilir.



Şekil 6.7 : JDO Veri Erişim Modeli

6.3. Kullanılan JDO Uygulama Ara yüzü

JDO uyumlu yazılım araçları çok sayıda şirket tarafından geliştirilip standartların haricinde bazı ek kolaylıklarda eklenmiştir. JDO uyumlu yazılım araçları geliştiren bazı şirketler ve bu şirketlerin ürettikleri ürünler Tablo 6.2'de gösterilmiştir.

Tablo 6.2. JDO Uyumlu Yazılım Geliştiren Şirketler

Ürün Adı	Üretici Firma
enJin	Versant Software
FastObjects"	Poet Software

[31] Aslında bütün JDO örneklerinin "PersistanceCapable" ara yüzünü kullanmalarını gerekmektedir. Bu geliştirme sürecinin temelidir. Ancak genellikle kullanılan JDO araçları bu yapıyı kendileri otomatik olarak sağlarlar.

FrontierSuite	ObjectFrontier
JDO Genie	Hemisphere Technologies
Kodo JDO	Solarmetric
ObjectDB	ObjectDB Software
LiDO	LIBeLIS

Tez çalışmasında Agvent Sunucuları üzerinde gelen kayıt ve duyuru mesajları saklamak için ObjectDB'nin kullanılması tercih edilmiştir. Bunun nedenleri şu şekilde sıralanabilir:

- ***ObjectDB bir Nesne Veri Tabanıdır.*** ObjectDB ile nesneye yönelik uygulamalar geliştirilmesi çok kolay ve verimlidir. Çünkü veri tabanının içeriği de birer nesnedir. Halbuki İlişkisel Veri Tabanı Yönetim Sistemleriyle çalışmak, geliştirilen uygulamadaki sınıflar ve nesnelerin yanı sıra tablolarla, kayıtlarla ve SQL ile uğraşmak gerektiği için zordur.
- ***ObjectDB JDO Uyumludur.*** ObjectDB Sun tarafından geliştirilen JDO standartlarıyla tam olarak uyumludur. Bu nedenle ObjectDB kullanan uygulamalar kullandıkları veri tabanına erişim için ObjectDB ye bağımlı değildirler. Başka bir JDO uyumlu API aracılığı ile de bu veri tabanına erişebilir ve gereken işlemleri yapabilirler. Günümüzde JDO API'si Oracle, IBM DB2 ve Microsoft SQL Server gibi bir çok RDBMS sistemlerinde de desteklenmektedir.
- ***ObjectDB nin kullanımı kolaydır.*** ObjectDB kullanarak çok hızlı ve kolay bir şekilde etkin programlar üretilebilir. Aslında en kolay JDO gerçeklemesi olduğu da söylenebilir. Diğer JDO gerçeklemelerinin büyük bir kısmında sınıf ve nesne yapıları RDBMS teki tablo ve kayıt yapılarına haritalandırılmaktadır. Böyle bir haritalandırma Java kodu ile yazılmasına olanak sağlansa da yine veri tabanı bilgisi ve erişimine ihtiyaç duymaktadır. ObjectDB kullanımı sayesinde ilişkisel veritabanı, tablo, kayıt, alan, SQL, JDBC ve sürücü gibi kavramları geri plana iterek sadece geliştirilecek Java uygulamasına yönelinebilir.
- ***ObjectDB Taşınabilirdir.*** ObjectDB tamamıyla Java ile geliştirilmiştir. Bu nedenle Java'yı destekleyen her ortamda çalışabilir. Geliştirilen uygulama kolaylıkla Windows, Unix ve Macintosh gibi başka platformlara da sadece ObjectDB ve JDO'nun jar dosyalarını alarak taşınabilir. Aynı zamanda geliştirilen uygulama ObjectDB nin sadece 300 KB lik bellek ihtiyacı nedeniyle, PocketPC gibi küçük kapasiteli ortamlarda da çalıştırılabilir.
- ***ObjectDB çok hızlıdır.*** ObjectDB nin çok iyi performansı vardır. Bir saniyede binlerce nesneyi işleyebilir. İndeks dosyalarının uygun şekilde

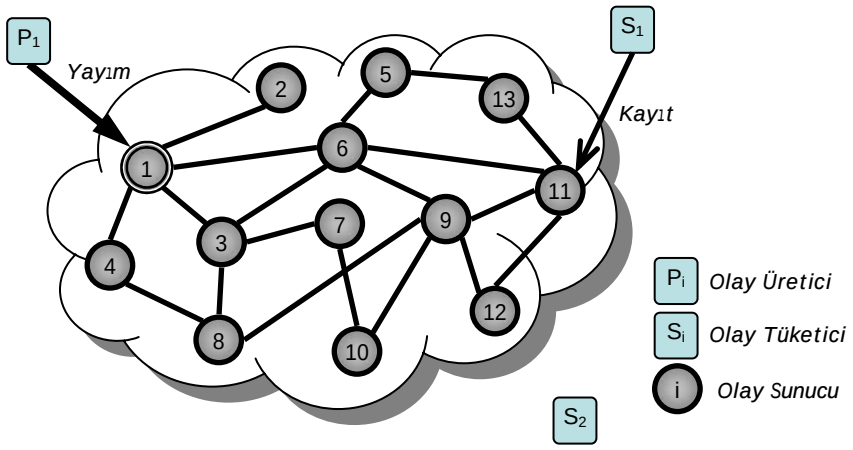
kurulması durumunda içinde on milyonlarca nesne kayıtlı bulunan veri tabanlarında dahi çok hızlı sorgulama sonuçları alınabilir.

- ***ObjectDB nin kendi veri tabanı gezgini vardır.*** ObjectDB JDO veri tabanlarını inceleyecek, içindeki nesneleri görüntüleyebilecek yeni Java nesneleri oluşturup ekleyebilecek, sorgulama geliştirip çalıştırabilecek görsel bir ara yüz sağlamaktadır. Çoğu nesne veritabanının görsel bir gezgini yoktur veya çok kısıtlı kapasitesi vardır. Veri tabanı programında da veri tabanının görüntülenmemesi ve sorgulama yapılamaması, hata ayıklayıcı (debugger) olmadan bir yazılım geliştirmeye benzer.

7. ABDES SİSTEMİNDE YÖNLENDİRME

Kayıt/yayım modeli kullanan dağıtılmış olay sistemlerinde, harici sistem bileşenleri olan Kayıtçılar (olay tüketiciler) ve Yayımcılar (olay üreticiler), olay sistemi ile iki temel mesajla haberleşirler: *kayıt (subscribe) ve yayım (publish)* mesajları.

Sistemin verimli ve etkin bir şekilde çalışabilmesi için bu mesajların Şekil 7.1'dekine benzer şekilde tasarlanmış olay sistemlerinde hızlı bir şekilde alıcısına (olay sunucusu veya olay tüketicisi) ulaştırılması ve bu mesajlara ihtiyaç duymayan alıcılara da bu mesajların ulaşmamasını sağlayacak şekilde olay sistemi tarafından yönlendirilmesi gerekmektedir.



Şekil 7.1 : Bir Olay Sistemi Topolojisi

7.1. Olay Sistemlerinde Yönlendirme

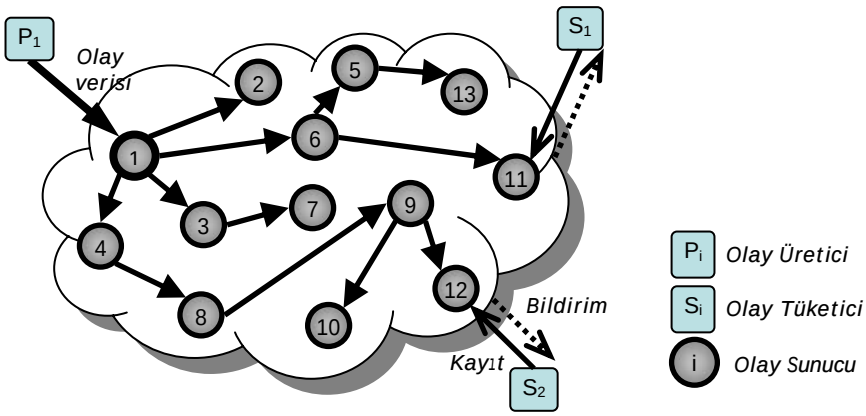
Yönlendirme probleminin çözümü için üç yaklaşım vardır. Bu üç yaklaşımın da ortak özelliği, hepsinde de mutlaka yayın (broadcast) yapısının kullanılmasıdır. Bu yayın yapısının temel özelliği, olay sunucularının önceden sistemdeki sunucuların yerleşkesi hakkında herhangi bir bilgiye ihtiyaç duymamasıdır. Bu yaklaşımları şu üç ana başlık altında incelenebilir;

7.1.1. Olay Verilerinin Yayını

Bu yapıda olay tüketicinin bağlı bulunduğu olay sunucu, gerekli filtrelemeleri yapmak için görevlendirilmiştir. Olay sunucu kendine gelen kayıt mesajlarını kendi bilgi tabanında tutarak, daha sonra gelecek olay verisini ilgili olay tüketicisine

yönlendirip yönlendirilmeyeceğine karar verir. Sistemdeki diğer Olay Sunucularının bu kayıt verisini veya Kayıtçı ile ilgili bilgileri tutmasına gerek yoktur. Sisteme gönderilen Olay Verisi, tüm Olay Sunucularına yayınlanarak, bu veriye ilgi duyan Olay Tüketicilerin bağlı bulunduğu sunuculara ulaşması sağlanır.

Şekil 7.2'de görüldüğü gibi, sisteme gönderilen kayıt mesajları 11, ve 12 numaralı olay sunucuları üzerinde tutulmakta, daha sonra bu sunuculara ilgili olay verisi ulaşınca S_1 ve S_2 Kayıtçılarına bu verinin iletilip iletilmemesine bu sunucular karar vermektedir.



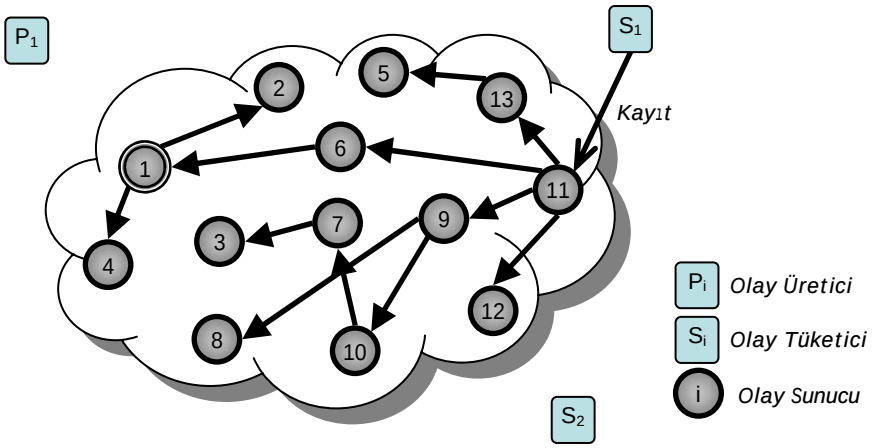
Şekil 7.2 : Olay Verisinin Yayını

Gerçeklemesi en kolay yönlendirme sistemi olmasına rağmen bu çözümün temel dezavantajı, üretilen olay verilerinin tamamının sistem içinde bulunan olay sunuculara dağıtılması zorunluluğudur. Dağıtılan olay verisini her hangi bir şekilde kullanmayacak olan olay sunucularına da bu olay verisi ulaşacak, sistemin yükünü artıracaktır ve verimi azaltacaktır.

7.1.2. Kayıt Mesajlarının Yayını

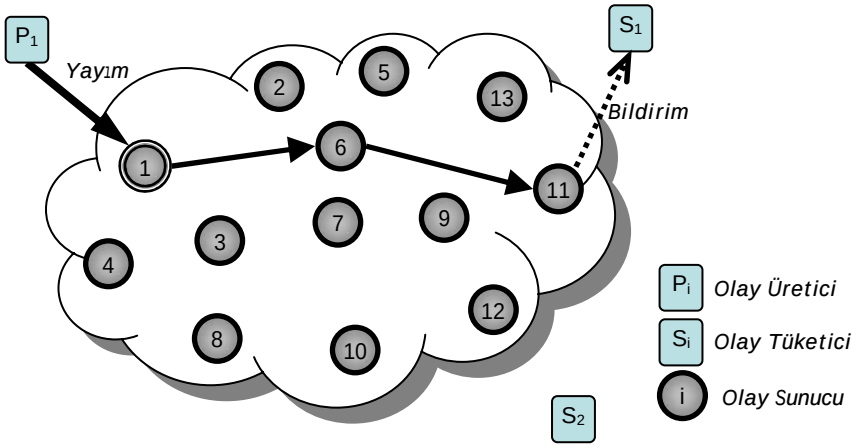
Bu yaklaşımda, olay verilerinin dağıtımında en kısa yol algoritması kullanılarak, sisteme kayıt olan Kayıtçılara ulaştırılması amaçlanmaktadır. Bu en kısa yolun oluşturulmasında ise sisteme gönderilen kayıt mesajlarından faydalanılmaktadır. Sisteme gönderilen kayıt mesajları, sistem üzerinde yayın yapısına uygun bir şekilde dağıtılarak, Yayımcıların (olay üreticilerin) gönderecekleri olay verilerinin izleyecekleri en kısa yolun belirlenmesi sağlanır.

Bu yayına örnek olarak Şekil 7.3'te görüldüğü gibi, S_1 olay tüketicisi tarafından üretilen ve kendine ulaşmasını istediği olay verilerini tanımladığı kayıt mesajını 11 numaralı olay sunucusuna gönderilmekte ve bu kayıt mesajı olay sistemi üzerinde yayınlanmaktadır.



Şekil 7.3 : Kayıt Mesajlarının Yayını

Kayıt mesajının sisteme yayınlanmasından sonra, Yayım mesajı ile sisteme gönderilen bir olay verisi, Şekil 7.4'teki yolu izleyerek ilgili Kayıtçıya ulaştırılır. Bu sayede, bir önceki yaklaşımda olduğu gibi, olay verilerinin gereksiz yere diğer olay sunucularına gönderilmesi engellenmiş olur.



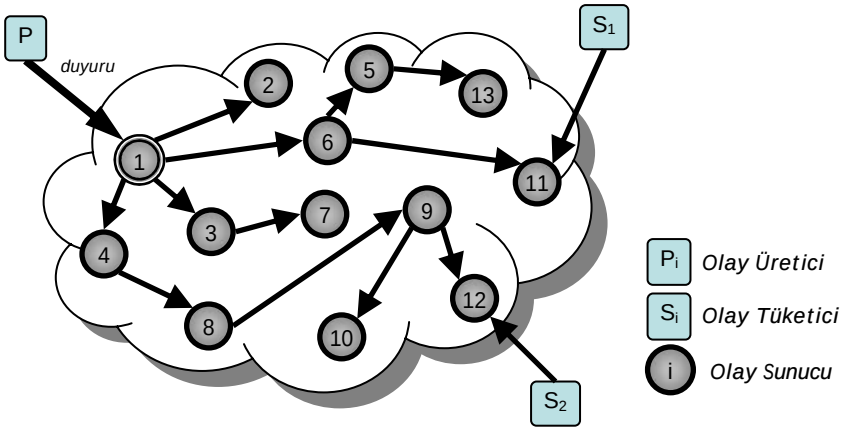
Şekil 7.4 : Olay Verisinin Yönlendirilmesi

Gerçeklenmesi nispeten kolay olmasına ve bir önceki modele göre sisteme gönderilen olay verilerinin dağıtımındaki trafiği ciddi boyutta azaltmasına rağmen, sisteme gönderilen kayıt mesajlarının çok olması durumunda (genelde öyledir) her kayıt mesajının sistem içinde bulunan tüm olay sunucularına yayınlanması gerekmektedir. Bu da gereksiz bir trafiğe yol açmaktadır.

7.1.3. Duyuruların Yayını

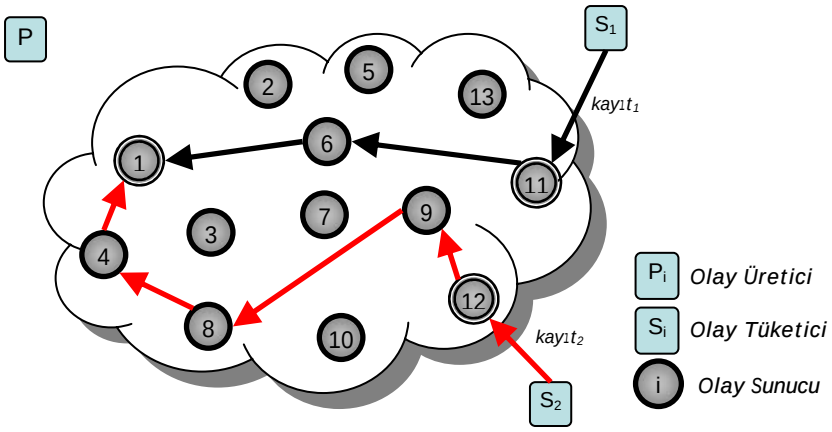
Üçüncü alternatif olarak gösterilen yol ise duyuru mesajlarının kullanılmasıdır. Kayıt mesajları nasıl ki, olay verileri için en kısa yolun oluşturulmasını sağlıyorsa, duyuru mesajlarının kullanılması da, kayıt mesajları için en kısa yolun oluşturulmasını ve bu mesajların ihtiyaç duyulmayan olay sunucularına gönderilmemesini sağlamaktadır.

Bir olay üretici, üretmeyi planladığı olay ile ilgili bir duyuru mesajı üreterek bu mesajı olay sistemine gönderir. Şekil 7.5'te görüldüğü gibi, olay sistemi kendine ulaşan bu duyuru mesajının yayınını yaparak, bunun tüm olay sunucularına ulaşmasını sağlar.



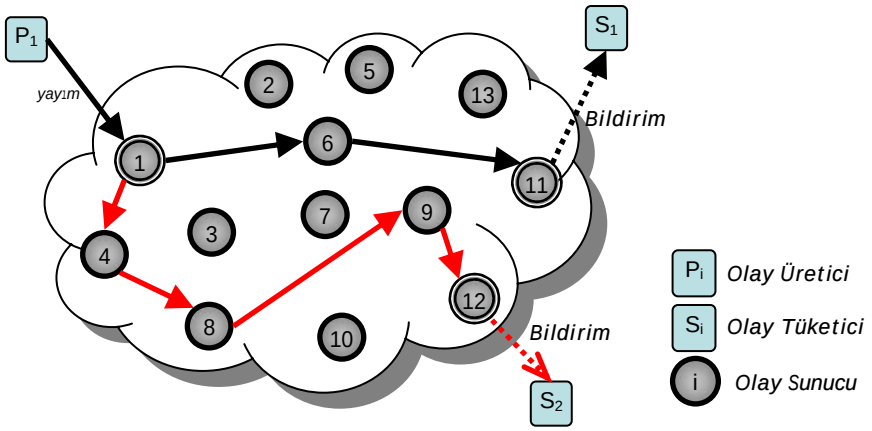
Şekil 7.5 : Duyuru Mesajlarının Yayını

Bir olay tüketici, ilgi duyduğu bir olay verisi için sisteme kayıt olmak istediğinde, bağlı bulunduğu olay sunucusuna bir kayıt mesajı gönderir. Olay sistemi, Şekil 7.6'da görüldüğü gibi, bu kayıt mesajını, olay üreticilerinin bağlı bulunduğu olay sunucularına, daha önce gönderilmiş olan duyuru mesajlarıyla belirlenmiş olan yol üzerinden gönderir (duyuru mesajlarının dağıtıldığı yolun tam tersi istikamette).



Şekil 7.6 : Kayıt Mesajının Yönlendirilmesi

Kayıt mesajının dağıtımının yapılmasından sonra sisteme gönderilen bir olay verisi, bu kayıt mesajının tam tersi istikamette hareket ederek, ilgili olay sunucularına ve olay tüketicilerine Şekil 7.7'deki gibi ulaştırılır. Bu sayede sadece *yayım* işleminin değil, aynı zamanda *kayıt* işleminin de sistemde oluşturmuş olduğu trafik en aza indirilir.



Şekil 7.7 : Olay Verisinin Yönlendirilmesi

Duyuru mesajlarının kullanılması sayesinde;

- Olay sistemi üzerindeki trafik yoğunluğu azaltılmakta,
- Olay verisinin dağıtımında rol oynamayacak, olay sunucularının bilgi tabanlarının büyümemesi sağlamakta,
- Bu sayede bu bilgi tabanından istenilen veriye ulaşma süresini azaltılmakta,
- Olay verisinin, olay tüketiciye ulaşma süresini azaltılarak sistem performansı ve verimi artırılmaktadır.

Bu nedenle son yıllarda geliştirilen kayıt/yayım sistemlerinde, üretilen kayıt mesajlarının yönlendirilmesine yardımcı olacak ve ilgili olay verisinin dağıtımında rol oynamayacak olay sunucularına bu kayıt mesajlarının ulaşmasını engelleyecek bu son yaklaşım ağırlıklı kullanılmaktadır.

7.2. ABDES Sisteminde Yönlendirme

Geliştirilen ABDES Sisteminde, agventlerin hızlı şekilde Kayıtçılara ulaşmasının amaçlandığı için, Duyuru mesajları kullanılmaktadır. ABDES Sisteminde üç temel mesajlaşma kullanılır. Bunlar:

1. Duyuru mesajları (Duyuru_Yap ve Duyuru_Sil mesajları),
2. Kayıt mesajları (Kayıt_Ol ve Kayıt_Sil mesajları),
3. Agventler (Agventlerin yayımlanması).

ABDES Sistemi temelde olay sistemlerinin *Duyuru Yayımı* modelini kullanıyor olsa da modeldeki bazı farklılıkları açısından bu mesajların nasıl yönlendirildiklerini [79] da anlatıldığı gibi biraz daha detaylandırmak gerekmektedir.

7.2.1. Duyuruların Yönlendirilmesi

Duyurular, Yayımcıların yayımlamayı planladıkları agventlerin niteliklerini ifade eden mesajlardır. Diğer olay sistemlerinde duyurular sadece yayımı yapılacak olay verisinin adını içermektedir. Bu mesaj Kayıtçıların, yayımlanabilecek agvent tipleri ve nitelikleri hakkında bilgi sahibi olabilmelerini ve kayıt mesajının en kısa yol üzerinden yönlendirilmesini sağlamaktadır.

ABDES Sisteminde ise Duyuru mesajı biraz daha nitelik kazandırılmış ve sadece yayımı yapılacak agventin adının yanı sıra, bu agventin filtrelenebilecek özelliklerini ve davranışlarını da içeren bir sınıf yapısında geliştirilmiştir.

Her Kayıtçının, sistemde yayımlanan ve/veya yayımlanabilecek agventler üzerine kayıt olabilmeleri için, Yayımcılar tarafından üretilen duyurunun tüm AS'lara yayınlanması gerekmektedir. Bunun için de en temel yol bir AS'ya gelen duyuru mesajının, geldiği AS dışındaki tüm AS'lara gönderilmesi ile olabilir (flooding-sel tekniği) (Şekil 7.5'te olduğu gibi[32]).

Bu yayınlama tekniğinde ise AS'ların oluşturduğu graf üzerinde döngüsellikten dolayı tekrarlanan duyuru mesajlarının AS'ların bilgi tabanlarına yeniden eklenmemesi önemlidir. Bunun içinde bir duyuru mesajının ulaştığı ilk AS'yu (Şekil 7.5'te 1 numaralı AS) kök olarak kabul eden ve sistemde bulunan her AS'ya bu mesajın yayınlanmasını sağlayacak bir yol ağacının oluşturulması gerekmektedir. AS'ya bir duyuru mesajı ulaştığında sunucunun yönlendirme için izleyeceği algoritma şu şekildedir;

```
1 Bilgi Tabanında Duyuru Mesajının var olup olmadığını kontrol et (Duyuru adı ve üretici Yayımcısına göre)
2 Eger varsa
3     Islem yok
4 yoksa
5     Duyuru Mesajını bilgi tabanına ekle
6     Duyurunun geldigi AS dışındaki Komşu AS ların Listesini al
7     Duyuru mesajını bu AS lara yönlendir.
```

Duyuru_Sil mesajlarının yönlendirilmesi aynen *Duyuru_Yap* mesajlarının yönlendirilmesi gibidir. Yukarıdaki algoritmadan tek farkı 6. satırdaki "Duyuru Mesajını bilgi tabanına ekle" yerine "Duyuru Mesajını bilgi tabanından sil" olarak değiştirilecek olmasıdır.

[32] ABDES Sisteminin yönlendirme mekanizmasında daha önce anlatılan Olay Sistemlerindeki Duyuru Yayımı mekanizmasının şekilleri üzerinden anlatılmaktadır. O şekillerde bahsedilen Olay Sunucular,(Agvent Sunucu), Olay Üreticiler, (Yayımcı), ve Olay Tüketicilerde (Kayıtçı) olarak adlandırılmaktadır.

7.2.2. Kayıt Mesajlarının Yönlendirilmesi

Kayıt mesajları, Kayıtçı tarafından üretilen ve kendine ulaşması istenen agventin özelliklerini tanımlayan bir filtre olarak tanımlanabilir. ABDES Sisteminde, diğer sistemlerden farklı olarak, sadece olay verilerinin durum değişkenleri üzerine değil, olay verilerin davranışları üzerine de filtreleme yapılmasına olanak sağlanmaktadır.

Kayıt mesajlarının yönlendirilmeleri, daha önce sistem üzerine dağıtılmış olan duyuru mesajlarına göre yapılır. Kayıt mesajları, duyuru mesajlarının geldikleri yönün tam tersine gidecek şekilde, Şekil 7.6'da gösterildiği gibi yönlendirilirler.

ASya bir kayıt mesajı ulaştığında sunucunun yönlendirme için izleyeceği algoritma şu şekildedir;

```
1   Kullanıcı adı ve şifre kontrolünü yap.
2   Eger dogruysa
3       Kayıtçıya onay mesajı dönder. (Senkron haberleşme)
4       Bilgi Tabanında kayıt olunan Duyuru kayıtlarını al
5       Duyuru kayıtlarının geldikleri Agvent Sunucu Listesini oluştur (Bilgi
            Tabanındaki Duyuru kayıtlarındaki senderID alanından)
6       Kayıt mesajını bu AS lara yönlendir.
7   Degilse
8       Kayıtçıya ret mesajı gönder
```

7.2.3. Agventlerin Yönlendirilmesi

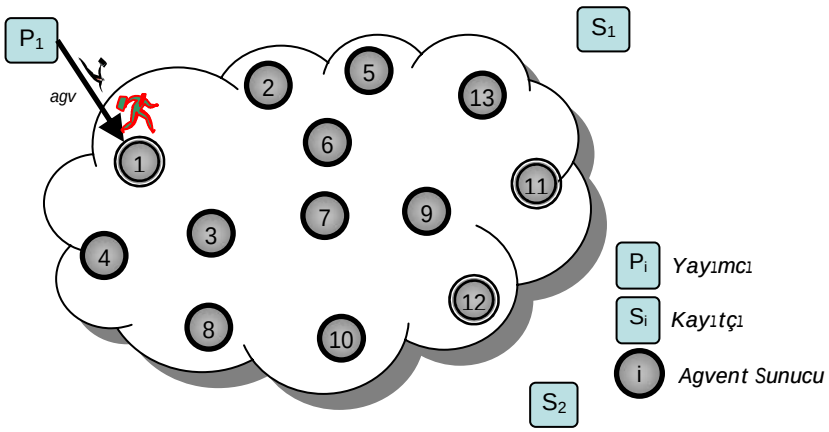
ABDES Sisteminin yönlendirme metodolojisindeki temel farklılık agventlerin yönlendirilme modelinden kaynaklanmaktadır. Daha önce de bahsedildiği gibi, ABDES Sistemini diğer olay sistemlerinden farklı kılan temel özellik, sistemin temel taşı olan agventlerin basit birer veri şeklinde değil de, kendi özellikleri ve davranışları olan sistemin birinci sınıfı bir bileşeni olmasından kaynaklanmaktadır.

Agvent genel manada, *"Yayımcılar tarafından üretilen olay verisini Kayıtçılara ulaştırmak üzere ABDES Sistemi üzerinde göç ederek hareket eden ve bağlantı düğümlerinde aktif hale getirilerek göç edeceği hedef makineyi/makineleri seçip kendisini özerk olarak bu makinelere yönlendirilmesini sağlayan kod ve veri bütünü"* olarak tanımlanmıştır.

Agventler Yayımcılar tarafından üretilerek, Kayıtçılara ulaşmaları için herhangi bir topoloji ile birleştirilmiş AS'*larından* oluşan *ABDES Sistemine* gönderilirler. Her düğümde yer alan bir AS'su, kendine ulaşan agventlerin çalışması, kendi yaratılma amacına uygun işlevler yapabilmesi ve ihtiyaç duyulması durumunda kendisini (veya kopyasını) hedef bir makineye göndermesi için uygun bir platform oluşturur.

Her agvent sahip olduğu yönlendirme yordamına göre hangi hedeflere, hangi ölçütler doğrulandığında ulaşacağına karar verir. Bu karar verme işlemi iki aşamada yapılmaktadır. Bunlar:

- *Bağlı Bulunduğu AS Üzerinde (1. Aşama):* Üretilen bir agvent, ABDES Sistemine bağlı bulunduğu AS'su üzerinden gönderilir. AS'su agventin ilk ulaştığı AS olduğu için aktif hale getirilir. Aktif hale gelen agvent kendisi için gerekli olan yönlendirme yol grafını oluşturması için çalıştığı AS'su üzerinde saklanan verilere (bilgi tabanına) erişmesi gerekir. Bu erişim işlemleri için agvente doğrudan yetki verilmesi, agventin sunucu üzerindeki işlemlerinin denetim dışında kalmasına ve erişim yetkisi dışında kalan bilgilere de erişilmesine olanak sağlayacaktır. Güvenliğin ve bilgi saklamanın ön planda olduğu sistemlerde böyle bir durum ciddi sakıncalara yol açabilir. Bu nedenle her agvent ihtiyaç duyduğu verilere erişebilmek için çalıştığı sunucu üzerindeki Agvent Yöneticisi ile haberleşerek gerçekleştirdiği işbirliği sonucu ilgilendiği verilere, sunucunun bilgi tabanından ulaşır.



Şekil 7.8 : Bağlı Bulunulan AS Üzerindeki Aşama

Şekil 7.8'de görüldüğü gibi, ABDES Sistemindeki ilk AS'suna ulaşan agvent, aktif hale getirilmektedir. Aktif hale getirilen agvent, ulaştığı AS'nun bilgi tabanındaki Kayıtçının tanımlayıcısını, adresini (bağlı bulunduğu AS'nun adresini) ve kayıt yaptığı ölçütlerini inceleyerek bu Kayıtçıya doğru kendini yönlendirip yönlendirmemeye karar verir. Bu yönlendirme aşamasında hedef adres olarak Kayıtçının bağlı bulunduğu AS'sunun adresi alınır ve *hedef AS listesi* oluşturulur.

Hedef AS listesi oluşturulduktan sonra agvent, kendisinin birer kopyasını oluşturarak bu AS'lara doğru gönderir. Burada yönlendirme işlemi agventin kendisi tarafından başlatılır. Bu aşamadan sonra hedefteki AS'lar dışındaki AS'larda, agvent yeniden aktif hale getirilmeyip, gelen agvent doğrudan ilgili AS'larına doğru yönlendirilir. Böylece sistemde dolaşan agventin her AS'da değil, sadece gerekli AS'lar olan *agventin gönderildiği ilk AS ve Kayıtçının bağlı bulunduğu AS* üzerinde aktif hale getirilmesi sağlanır. Agvent bu hedef AS'lardan birine ulaştıktan sonra yeniden aktif hale getirilir ve ilgili

Kayıtçıya gönderilip gönderilmeyeceğine karar verilir. Bu karar AS'su tarafından değil agventin kendisi tarafından verilecektir.

AS üzerinde ilk kez aktif hale gelen bir agventin izleyeceği algoritma aşağıdaki gibidir.

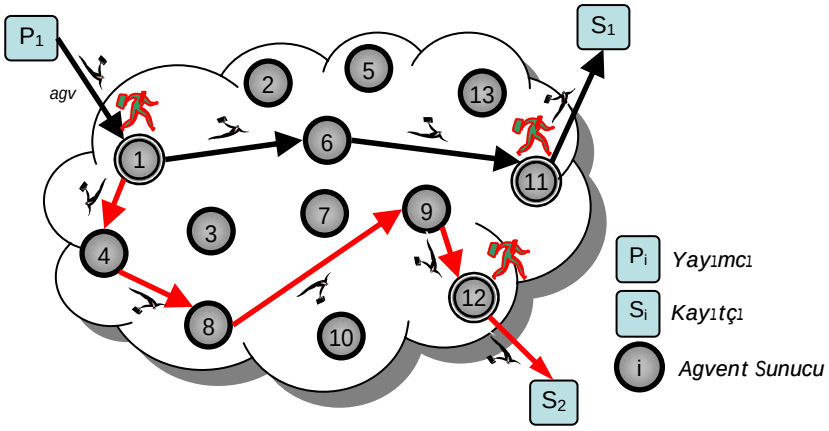
```
1       ASnun Bilgi tabanından kendi adı üzerine kayıtlı olan kayıt mesajlarını al
2       Eger Kayıtçının isim ve/veya adresi Yasak Listesindeyse
3               Işlem Yok
4       Degilse
5               Kayıt ölçütlerini kontrol et.
6               Uygun Kayıtçıların adreslerini Yönlendirme Listene ekle
7               Kendini, AS'nun Giden Agvent Kuyruguna koy.
8               Kendini Sonlandır.
```

- *Kayıtçının Bağlı Bulunduğu AS Üzerinde (2.Aşama):* ABDES Sistemi, birinci aşamada aktif hale gelerek, kendini yönlendireceği Kayıtçıları seçen agvent kendisinin Şekil 7.9'da görüldüğü gibi, ilgili AS'larına yönlendirilmelerini sağlar.

  Kayıtçı ile ilgili özel, detaylı ve istatistiksel bilgiler, Kayıtçının bağlı bulunduğu AS'su üzerinde tutulur. Örneğin bir şirkete ait "çalışan sayısı, net gelir, kısa vadeli borç miktarı, uzun vadeli borç miktarı, toplam mal varlığı, yatırım miktarı, aktiflerinin değeri, pasiflerini değeri" gibi her AS'su üzerinde tutulmasına gerekli olmayan özel bilgiler tutulacağı gibi, hedefteki Kayıtçı bilgisayarın performansını gösteren CPU hızı, bellek miktarı, hard diskteki boş alan gibi bilgiler Kayıtçının bağlı bulunduğu AS'su üzerinde tutulmaktadır.

  Bu bilgileri agventin görebilmesi, buna göre filtreleme yapabilmesi ve kendisini ilgili Kayıtçıya gönderip göndermemeye karar verebilmesi açısından agventin, ABDES Sistemi üzerindeki son uç nokta olan bu AS üzerinde de aktif hale getirilmesi gerekmektedir. Şekil 7.9'da görüldüğü gibi, bu uç AS'larında aktif hale getirilen Agventler Kayıtçının buradaki bilgilerini de değerlendirerek kendilerini Kayıtçılara yönlendirirler veya yönlendirmezler.

  Bu aşamadan sonra Kayıtçıya doğru gönderilen agvent ktif hale getirilir ve üretim amacına göre çalışarak, kendine yüklenen eylemleri yerine getirir.



Şekil 7.9 : Agventin Kayıtçının Bağlı Bulunduğu AS Üzerindeki Çalışması

7.3. Hata Kotarma

Olay tabanlı sistemlerdeki hata hoşgörü mekanizmaları iki ana başlık üzerinde toplanmaktadır. Kendi Kendine Stabilizasyon (Self Stabilization) ve Yeniden Yapılandırma (Reconfiguration). Bazı sistemler [80, 81] dağıtım mekanizması içindeki bazı hatların kopabileceği öngörüsü üzerinde durarak hata hoşgörü için yönlendirme tablosunu yeniden yapılandırmayı ön plana çıkarmışlardır. Yeniden yapılandırma hat kopmasından sonra oluşan yeni topoloji için yönlendirme tablolarının doğru olmayacağı ve yeniden düzenlemesi gerektiğini ifade etmektedir. Böyle bir yeniden yapılandırma düğümler(aracılar) arasındaki bir veya daha fazla hatta kopukluk olması veya yeni bağlantıların kurulması durumunda tetiklemektedir. Kendi Kendine Stabilizasyon içerdiği hata hoşgörü modelinde daha iyimser bir yaklaşım sergileyerek sistemin ölçeklenebilirliğini artırmaktadır [82, 83]. Stabilizasyon ile geçerliliği kaybetmiş düğüm bilgileri ile bağlantı kesilen düğüm bilgileri yönlendirme tablolarında belirli aralıklarla güncellenerek düğümlerin kendi aralarında senkronize edilmeleri sağlanmaktadır. Bu modellerin her ikisi de ağ üzerindeki trafiği ciddi boyutlarda artırmaktadır. Bu nedenle bazı araştırmacılar [84] ise sadece bu hata hoşgörü modellerindeki ağ trafiğini azaltmak için bazı çalışmalar yapmışlardır.

Az sayıda araştırmacı ise hat veya düğümümdeki bir problemden dolayı sistemin sorunsuz çalışması için başka düğümlerin devreye sokulmasını denemişlerdir [85]. Bu model de bir düğümün veya hattın çökmesi durumunda başka bir düğüm , çöken düğümün yerini almakta ve işlemleri gerçekleştirmektedir. Hattın kopması durumunda ise veri transferi hat yeniden aktif hale gelene kadar bekletilmekte daha sonar yeniden gönderilmektedir. Bunun temel dezavantajı ise hat yeniden aktif hale gelmez ise mesaj dağıtılamayacak olmasıdır.

Kesintisiz çalışan sistemde değişik nedenlerden dolayı bazı AS'ların çökmesi karşılaşılabilecek bir durumdur. Böyle bir sorun karşısında sistemin sürekliliğini sağlaması gerekmektedir. Bunun için bir AS'sunun çökmesi durumundaki mesajların dağıtılmasını (hata kotarılmasını) üç farklı kategoride incelemek gerekmektedir [86]. Bunlar;

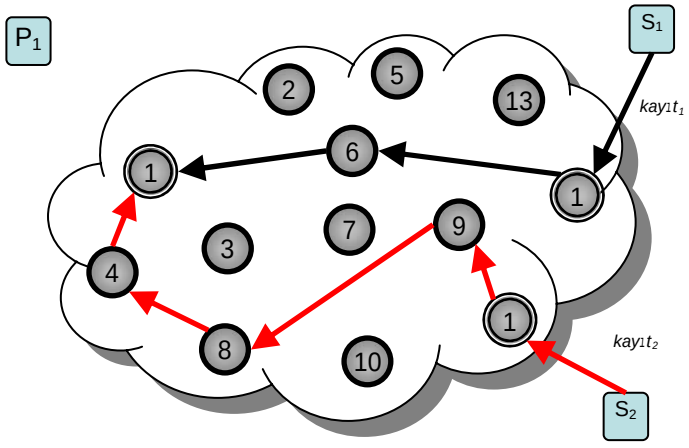
- *Duyuru Mesajlarının Dağıtımında Hata Kotarma*
- *Kayıt Mesajlarının Dağıtımında Hata Kotarma ve*
- *Agventlerin Dağıtımında Hata Kotarma*

7.3.1. Duyuru Mesajlarının Dağıtımında Hata Kotarma

ABDES Sisteminde etkin haberleşmenin sağlanması, sistem bileşenleri arasında gereksiz mesajlaşmanın engellenmesi, kayıt mesajlarının ve agventlerin kısa yoldan hedeflerine ulaşmalarının sağlanması amacıyla duyuru mesajlarının kullanıldığından daha önceki bölümlerde bahsetmiştik. Duyuru mesajları sistem üzerinde yayın (broadcast) şeklinde tüm AS'lara dağıtılır. Bu yayın aşamasında hatta olmayan AS'su gönderilen duyuru mesajlarında haliyle haberdar olamayacaktır. Aktif olmayan bu AS'ya herhangi bir Kayıtçının da bağlı olamayacağı için hata kotarma için özel bir yapı geliştirilmemiştir. AS'su sisteme dahil olduğunda, komşu AS larla kendi bilgilerini senkronize eder.

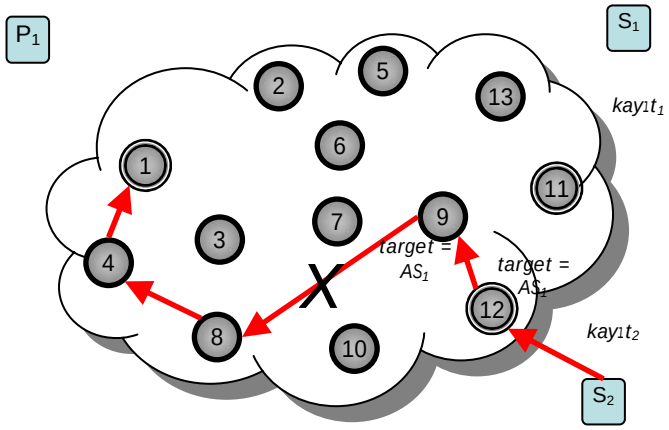
7.3.2. Kayıt Mesajlarının Dağıtımında Hata Kotarma

Kayıt mesajları, sistemde yayınlanan duyuru mesajlarının tam tersi yönünde gönderilerek Yayımcının bağlı bulunduğu AS'suna ulaştırılır. Şekil 7.10'da S_1 ve S_2 Kayıtçılarının göndermiş oldukları bir kayıt mesajının AS'lar üzerinde nasıl hareket ederek hedef AS olan AS_1 e yönlendirilmelerinin gerektiği gösterilmektedir.



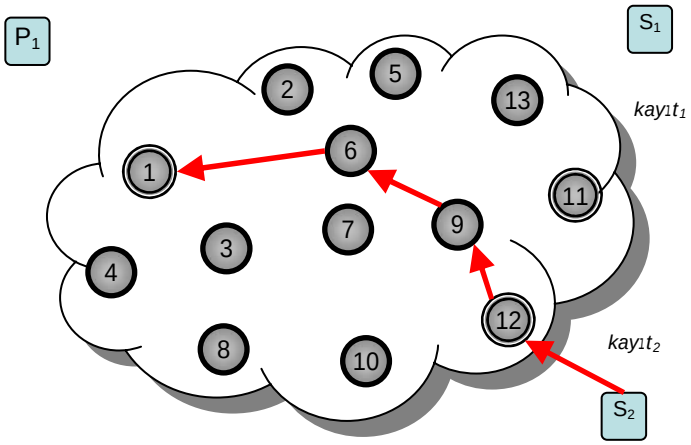
Şekil 7.10 : Kayıt Mesajlarının ABDES Sisteminde Yönlendirilmesi

Kayıt mesajlarının bu şekilde yönlendirilmesi sırasında her hangi bir sebepten ötürü Şekil 7.11'de görüldüğü gibi, AS_9'dan AS_8'e ulaşılamaması durumunda dahi kayıt mesajının AS_1'e ulaştırılması gerekmektedir. Bunun için asıl rota kullanılamadığı için AS'su tarafından tutulan alternatif bir rota kullanılmaktadır. Her AS'su kendine ulaşan *duyuru mesajları, kayıt mesajları ve agventleri* inceleyerek kendi alternatif rota tablosunu oluşturur.



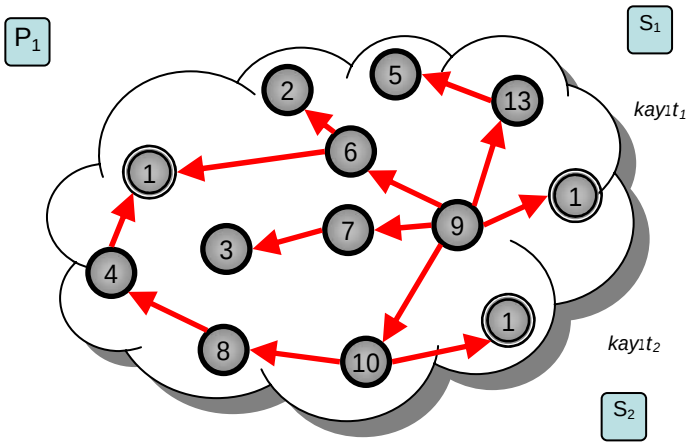
Şekil 7.11 : Kayıt Mesajlarının Yönlendirilmesinde bir AS'nun Çökmesi Durumu

AS_9 kendine ulaşan agventi AS_1 e doğru AS_8 üzerinden göndermesi gerekmektedir. Ancak AS_8 e ulaşamadığı için AS_1 e ulaşmak için alternatif rota tablosuna bakarak Şekil 7.12'de gösterildiği gibi başka bir yoldan AS_1 ulaşır.



Şekil 7.12 : Kayıt Mesajının Alternatif Rota Üzerinden Yönlendirilmesi

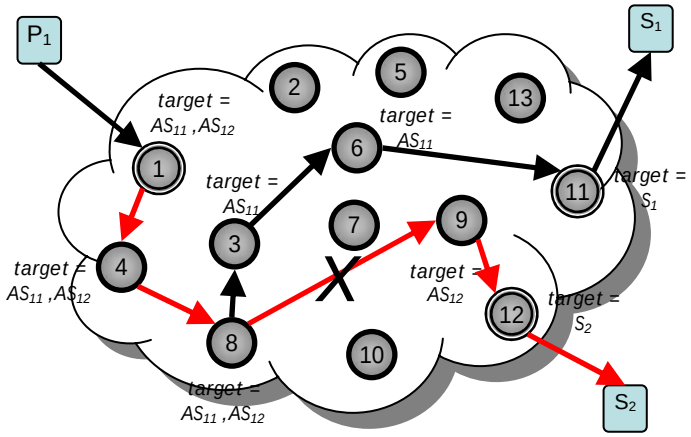
Eğer AS üzerinde AS_1 için bir alternatif rota bulunamamış ise o zaman bu kayıt mesajının AS_1'e ulaştırılması için kayıt mesajı AS_9 üzerinden *yayın* şeklinde gönderilerek sistem üzerinde dağıtılması sağlanır (bakınız Şekil 7.13). Bu gönderilen kayıt mesajının hedef makinesi AS_1 olduğu için diğer sunucuların kendi kayıt tablolarını güncellemelerine gerek kalmaz. Gelen bu mesajı kendi alternatif rota tablolarını güncellemekte kullanabilirler.



Şekil 7.13 : Kayıt Mesajlarının Yayın ile Gönderilmesi

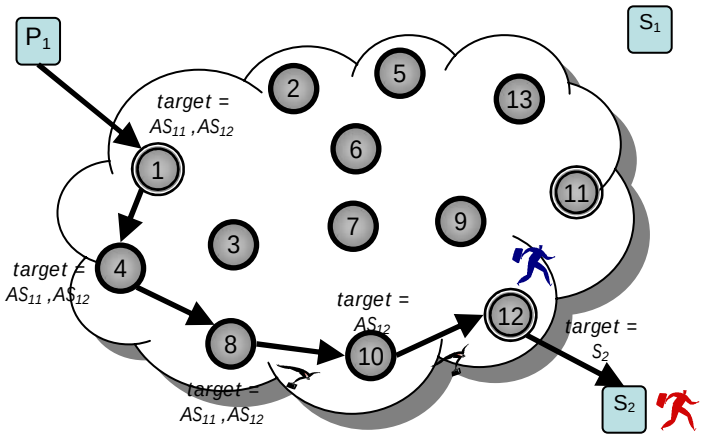
7.3.3. Agventlerin Dağıtımında Hata Kotarma

Yukarıda da anlatıldığı gibi, sisteme gönderilen agvent, sistem üzerindeki ilk AS üzerinde aktif hale getirilerek kendisini Şekil 7.14'te gösterildiği hedef AS'lara doğru yönlendirmesi sağlanır.



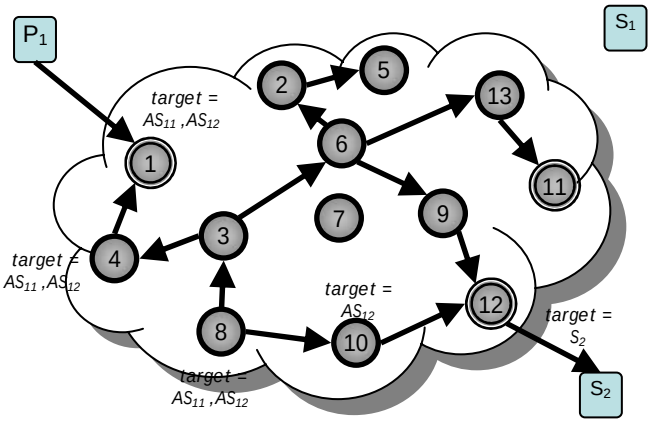
Şekil 7.14 : Agventlerin Yönlendirilmesinde bir AS'nun Çökmesi Durumu

Şekilde de görüldüğü üzere AS_{11} ve AS_{12}'ye doğru yönlendirilen agventler, AS_8'den AS_9'a herhangi bir sebepten ötürü ulaşamamaktadır. Bu durumda, aynı kayıt mesajlarında da olduğu gibi, agventlerin yönlendirilmelerinde de AS'ların alternatif rota tablolarından faydalanılır. Şekil 7.15'teki gibi AS_8 üzerindeki bir agvent AS_9'a ulaşılamadığı için, AS_{10} üzerinden hedef AS'su olan AS_{12}'ye ulaştırılır.



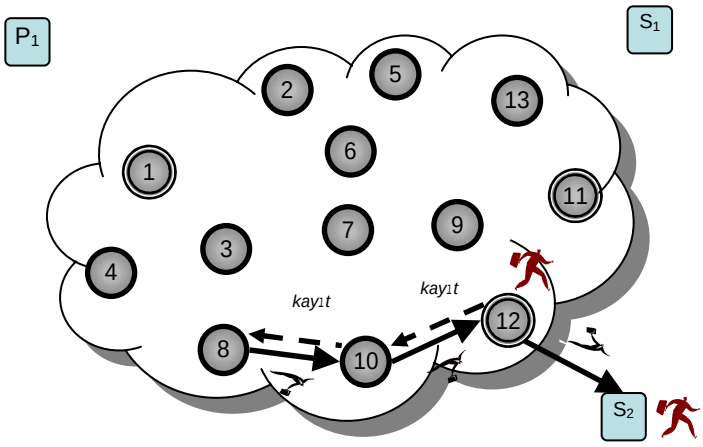
Şekil 7.15 : Agventin Alternatif Rota Üzerinden Yönlendirilmesi

Agventlerin bu şekilde yönlendirilmelerinin, kayıt mesajlarının yönlendirilmelerinden bir farkı yoktur. Ancak eğer AS'su üzerinde alternatif rotaya ulaşılamamışsa, kayıt mesajlarında olduğu gibi agventlerinde bir yayın ile sisteme dağıtılması pek mantıklı olmayacaktır. Çünkü agventler yapıları gereği, bir veri ve kod kümesi olmalarından dolayı büyük boyutlu olabilirler. Büyük boyutlu agventlerin sistemde yayınlanması ise yavaşlamaya yol açacaktır. Bunun için yukarıdaki gibi bir çökme örneğinde, AS_8 üzerinden yeniden bir duyuru mesajı yayınlanarak (bakınız Şekil 7.16), AS_{12}'ye bir alternatif rota belirlenmesine çalışılır.



Şekil 7.16 : Duyuru Mesajının Yeniden Yayınlanması

Kendine ulaşan duyuru mesajına AS_{12} cevap vererek, Şekil 7.17'deki gibi bir alternatif rota oluşturulmasını sağlar. Bu rota oluştuktan sonra AS_8 üzerinde bekleyen agvent bu alternatif rota üzerinden AS_{12} ye doğru gönderilir.



Şekil 7.17 : Oluşan Alternatif Rota Üzerinden Agventin Yönlendirilmesi

8. UYGULAMA ÖRNEĞİ

Bu bölümde, tezde önerilen ABDES Sisteminin çalışma metodolojisi bir uygulama örneği üzerinde gösterilmekte olup, elde edilen test sonuçları incelenmektedir. Uygulama örneği olarak bir Yayın Dağıtım/Bildirim Sistemi [87] geliştirilmiştir.

Araştırma görevlisi veya öğretim üyesi gibi tekil bir şahsın, her hangi bir konuyla ilgili yayınlara erişmek istediğinde bu yayını aramasının geleneksel yolu; Internet üzerinden arama motorlarından, üye olunan dergilerden veya kütüphaneler üzerinden arama şeklindedir. Yayınlarla ilgilenme olayına biraz daha geniş kapsamda bakıldığında, ilgilenenler sadece tekil şahıslar olmayıp, kütüphaneler, araştırma kuruluşları, şirketler ve kitapevleri gibi kurumsal varlıklar da olabilmektedir. Bu gibi varlıklar ise belirli bir konu üzerindeki yayınlarla değil daha genel olarak her türlü yayınla ilgilenmekte ve ilgi duydukları yayınların üretilmesi/yayınlanması durumunda haberdar edilmek istemektedirler.

Dünya üzerinde çok sayıdaki yayıncı, kitapevi, dergi hatta Internet'in dünya üzerinde hızla yayılması ile tekil olarak yazarlar farklı konularda sürekli olarak yayın üretmekte ve bunları yayımlamaktadırlar. Mevcut kütüphanelerin en büyükleri bile göz önüne alındığında üretilen bu yayınların hepsini satın alarak kendi bünyesinde barındırmadığı görülmektedir. Gerek kütüphanenin yer problemi gerekse maddi problemler bunu engellemektedir. Bunun yerine kütüphaneler, üretilen yayınları belirli ölçütlere göre incelemekte ve bunlar arasından uygun gördüklerini seçmektedirler.

Üretilen yayınlara genel kapsamda bakıldığında bunlar özel kitlelere ve geniş kitlelere hitap eden yayınlar olarak sınıflanabilir. Örneğin NASA tarafından üretilen bir yayın, Boeing veya Airbus gibi uçak üreticilerini kapsayan özel tüketicilerin işine yarayan az talepli bir yayın olabileceği gibi, "Dağıtılmış Hesaplama" üzerine düzenlenen bir sempozyumun bildiri kitapçığı, Bilgisayar Mühendisliği Bölümü olan çok sayıdaki üniversitenin ilgi duymasının ötesinde, konu üzerine çalışan araştırmacıların, öğrenciler ve öğretim görevlileri de ilgi duyabileceği geniş bir tüketici kitlesine hitap edebilmektedir.

Akademik yayınlar hakkında bildirim yapısı zaten web üzerinden Springer [88], Elsevier [89] veya ACM Digital Library [90] gibi yayıncılar tarafından basit manada gerçeklenmektedir. Bu sunucular sadece yayıncının verdiği bilgilerden faydalanmakta, yayınının büyük bir kısmıyla ilgili bilgi sahibi olunmadığı veya bu

değerli bilginin gösterilmesi istenmediği için bildirim yapısı sınırlı fonksiyonelliğe sahip olmaktadır. Temelde bu servislerin yapmış olduğu bildirimler yayının içeriği, içindekiler bilgisi, yazar bilgileri veya anahtar sözcükleri ile ilgilidir. Kullanıcılar daha önce belirtmiş oldukları ölçütlere uyan yayınların üretilmesinden elektronik posta ile haberdar edilmektedir.

Uygulama örneği olarak değişik üreticiler tarafından üretilen yayınların, kütüphanelere, kurumlara veya tekil kullanıcılara iletilmesi, tanıtılması, hatta pazarlanabilmesine olanak sağlayan bir *Yayın Dağıtım Sistemi* geliştirilmiş ve sistemin performansı test edilmiştir.

Örnek uygulamada üretilen yayınların birer agvent olarak yapılandırıldığı, ve tüketici/kayıtçı olarak sadece kütüphanelerin değil, aynı zamanda araştırma kuruluşlarının, kitapevlerinin, şirketlerin ve tekil kullanıcıların da sisteme aktif olarak katılabileceği bir "Yayın Dağıtım/Bildirim Sistemi" uygulaması geliştirilmiş ve test edilmiştir. Bu sistemde arama motorlarından elde edilen anlık sonuçlardan farklı olarak uzun süre çalışmaya göre düzenlen yapısı sayesinde Kayıtçıların üretilen yayınlardan hızlı şekilde ve sürekli olarak haberdar edilmelerine olanak sağlayan bir uygulama geliştirilmiştir. Bu sistemde olay üreticisi olan yayınevleri veya özelde tekil yazarlar, oluşturdukları yayının sistem üzerinden pazarlanmasını amaçlamaktadırlar. Sisteme kayıt olan olay tüketiciler/kayıtçılar ise belirli ölçütlere sahip olay verilerinin (yayınların) kendine ulaşmasını istemekte ve kendine ulaşan veriler neticesinde seçimini yapmakta ve seçilen yayımcı ile bağlantı kurarak ulaşmak istediği yayına doğrudan ulaşabilmektedir. Olay verisi (üretilen yayın) kendine ulaşana kadar tüketiciler ve üreticiler birbirlerini bilmemektedir.

Böyle bir dağıtım sisteminde yayımcı ürününü mümkün olduğu kadar fazla tüketiciye pazarlamak isterken, kendi üretmiş olduğu yayın ile ilgili fazla bilgi dışarıya vermek istememektedir. Günümüzde sayısal verilerin kolaylıkla kopyalanıp çoğaltılabildiği ortamda, yayınlanan bir yayının bilgilerini dışarıya göstermeden pazarlamasının yapılması önemlidir. Bu nedenle üretilen verinin (yayının) tamamının konduğu halde veri saklamanın sağlanabildiği bir yapı olarak *ABDES Sistemi* uygulamanın amacına uygun olmaktadır. Bu uygulamada yayınlar olarak kastedilen teknik raporlar, sempozyum bildirileri, dergi makaleleri veya akademik kitaplar gibi bilimsel yayınlar olabileceği gibi, romanlar, macera kitapları ve magazin dergileri gibi gündelik hayatla ilgili sosyal yayınlar da olabilmektedir. Bu uygulamadaki bileşenlerin/katılımcıların şu özellikleri olduğu öngörülmüştür.

Kayıtçılar: Bunlar kütüphaneler, araştırmacılar, öğretim üyeleri ve öğrenciler olarak değerlendirilen sistem varlıklarıdır. Bir Kayıtçı, sistemden belirli ölçütlere sahip

yayınların seçilmesi/satın alınması amaçlanmaktadır. Bu ölçütler iki kısımda somutlaştırılmıştır. Bunlar:

- *Agvent Sunucular üzerinde değerlendirilecek ölçütler:* Bunlar olmazsa olmaz türünden ölçütler olup, Kayıtçı sadece bu ölçütlere uyan yayınların kendine ulaşmasını beklemektedir. Bu ölçütler sistemin geliştirim amacına uygun olarak sadece agventin özelliklerine göre değil, aynı zamanda davranışlarına göre de belirlenebilmektedir. Bu ölçütlere örnek olarak:
    - bir yayının adına göre,
    - özünde yer alabilecek kelimelere göre,
    - satış fiyatına göre (satın alınacak miktara ve teslim yerine bağlı olarak),
    - veya içeriğinin yapısına göre tanımlanabilecek ölçütler verilebilir.

  Sunucu üzerinde değerlendirilecek ölçütler, Kayıtçı tarafından bir kayıt mesajı ile ABDES Sistemine bildirilmektedir. Bu mesaj sistem tarafından uygun şekilde işlenmekte ve sisteme gönderilen agventlerin yönlendirilmesinde faydalanılmaktadır.
- *Kayıtçı üzerinde değerlendirilecek ölçütler:* Bu ölçütler ise değerlendirilmesi uzun zaman alacak ve Kayıtçılar arasında farklılık gösterebilecek ölçütlerdir. Sistem üzerinde ölçülmesi *uzun zaman* alacak ölçütlerin değerlendirilme görevini Agvent Sunucular üzerine yüklenmesi sistemin yavaşlamasına sebep olacaktır. Bu nedenle bu gibi karmaşık ölçütlerin Kayıtçılar üzerinde değerlendirilmesi faydalıdır.

  Örneğin bir yayında geçen özel bir kelimenin yer aldığı cümlelerin Kayıtçıya sunularak, Kayıtçının bu yayın üzerinden karar vermesine olanak sağlanması, Kayıtçının özel ölçüti olarak değerlendirilmektedir. Yayın içinde bu cümlelerin seçilmesi, bütün yayının baştan sona cümle cümle incelenmesi gerektirdiği için uzun süre alacaktır. Bu nedenle bu işlemin Agvent Sunucular üzerinde değil de, Kayıtçının makinesinde yapılması gerekmekte, bu işlemde Kayıtçının etmeni tarafından, kendine ulaşan agventten yapılan bir taleple neticelendirilmektedir.

  Benzer şekilde agventten karşılıklı konuşma ile elde edilecek bilgilere göre de Kayıtçı, bu agventin içerdiği yayını almaya karar verebilmektedir. Bunun içinde Kayıtçı ile agventin mesaj alışverişlerinin aynı makine üzerinde (Kayıtçı makinesi üzerinde) yapılması gerekmektedir. Bu görüşmenin neticesi ise Kayıtçının karar vermesinde önemli bir rol oynamaktadır.

Yayımcılar: Yayınevleri, dergiler, aracı kurumlar veya doğrudan yayıncıların/ yazarların kendileri olarak düşünülebilir. Yayımcıların amaçları ile kütüphanelerin (Kayıtçıların) amaçları örtüşmemektedir. Yayımcı ürününü mümkün olduğu kadar fazla tüketiciye pazarlamak istemekte, ancak bu dağıtım sırasında oluşturmuş olduğu agventin Kayıtçıların ölçütlerine göre hareket etmesi gerekmektedir.. Üretilen agventte, agventin kendi filtrelemesine yönelik iki noktaya dikkat çekilmektedir. Bunlar:

- *Kayıtçıların seçilmesinde kullanılan filtreleme:* Yayımcı üretmiş olduğu agventi sisteme gönderirken kendi amacına yönelik olarak belirli ölçütleri de agvente eklemektedir. Örnek uygulamada, yayımcı istekçinin tekil şahıs mı yoksa kurumsal bir varlık mı olduğunu, kurumsal bir varlık ise sahip olduğu kitap miktarı, yıllık bütçesi veya üye sayısı gibi ölçütleri göz önüne alarak kendisini bu Kayıtçılara doğru yönlendirmek isteyebilmektedir.

  Bu filtreleme mekanizması daha öncede bahsedildiği üzere, Kayıtçının bağlı bulunduğu Agvent Sunucu üzerinde yapılır. Agvent burada aktif hale getirilerek, bilgi tabanındaki bilgilere göre kendisini ilgili Kayıtçıya yönlendirir veya yönlendirmez

- *Kayıtçı ile mesajlaşırken/haberleşirken kullanılan filtreleme:* Üretilen agvent, Kayıtçı üzerine ulaşınca da Kayıtçının isteklerine cevap vermesi gerekmektedir. Ancak agvent kendi amaçları doğrultusunda Kayıtçının taleplerine uygun cevaplar vermektedir. Örneğin;

  - Kayıtçı kendi üzerindeki agventten “aranan bir kelimenin kullanıldığı cümleleri” isteyebilmektedir. Günlük kullanımda çok sık kullanılan bazı kelimelerin (“and”, “or”, “of”, “the”, “a”, vs) istenilmesi yayının büyük bir kısmını Kayıtçıya göstereceği için bu gibi kelimeler agvent tarafından filtrelenerek Kayıtçının bu yöndeki isteklerine cevap verilmemektedir.

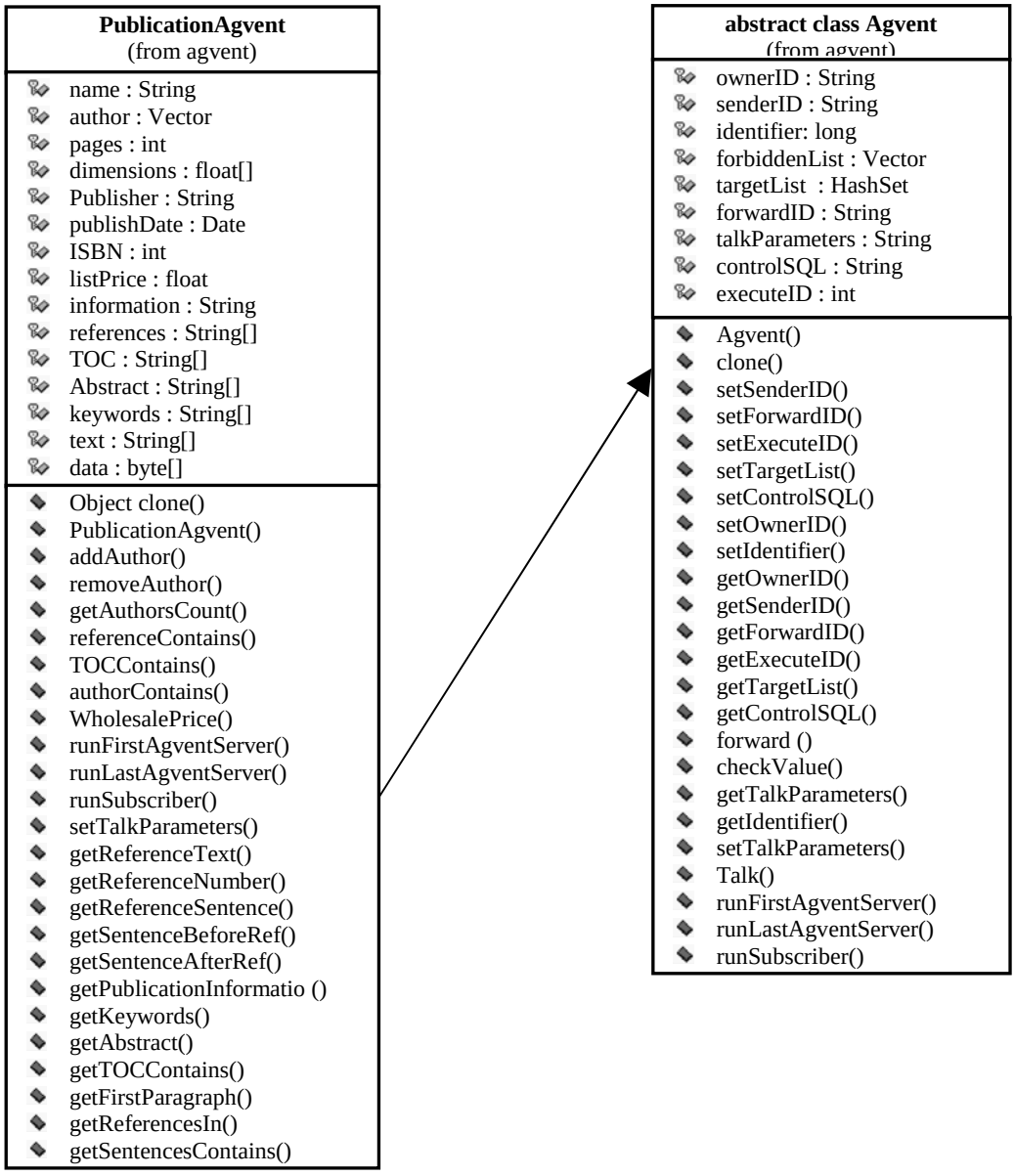
  - Kayıtçı ile agvent arasında yapılan karşılıklı görüşmeler sırasında, agventin üretim amacına uygun olarak Kayıtçının bazı sorularına/taleplerine cevap verilmeyebilmektedir.

Bundan sonraki bölümde yukarıda bahsedilen işlemleri yerine getirebilecek bir *PublicationAgvent* sınıfının geliştirilmesinde izlenen yollar gösterilmiştir.

8.1. PublicationAgvent’in Tasarımı

Agvent sınıfı bütün agventlerde olması gereken ortak özelliklerin tanımlandığı soyut bir sınıftır. Yukarıda özellikleri izah edilen sistemi gerçeklemek için bu sınıfın alt

sınıfı olan bir *PublicationAgvent* sınıf yapısı geliştirilmesi gerekmektedir. UML diyagramı Şekil 8.1'de gösterilen *PublicationAgvent* sınıfının nasıl geliştirildiğinin gösterilmesi, bu sistemi kullanacak uygulamaların, nasıl geliştirilebileceğini de göstereceği için biraz daha detaylı anlatılmaktadır.



Şekil 8.1 : PublicationAgvent Sınıfının UML Diyagramı

8.1.1. Agvent Sınıfının Bir Alt Sınıfı Olarak Tanımlama

Geliştirilen *PublicationAgvent* sınıfının sistem üzerinde kullanılabilmesi için soyut olarak tanımlanan *Agvent* sınıfının bir altsınıfı olarak tanımlanması gerekmektedir. Bu tanımlama ile agvent sınıfın soyut metotları olan *Talk(), setTalkParameters(), runFirstAgventServer(), runLastAgventServer() ve runSubscriber()* metotlarının PublicationAgvent sınıfında somutlaştırılması/gerçeklenmesi gerekmektedir.

PublicationAgvent sınıfının tanımlaması aşağıdaki kodlamaya benzer bir yapıda gerçeklenir.

```
public class PublicationAgvent extends Agvent {
 public void setTalkParameters()      { ********** };
 public String[] Talk(String question, String parameter)      { ********** };
 public void runFirstAgventServer()      { ********** };
 public void runLastAgventServer();      { ********** };
 public void runSubscriber() ;      { ********** };
}
```

Burada tanımlanan metotlar, agventin farklı konaklar üzerindeki davranışlarını tanımlamaktadır. Yayımcı tarafından agventin üretim aşamasında bu metotların içerisi doldurulmalı ve davranışlar belirlenmelidir. Geliştirilen uygulamadaki bu metodların işlevleri şu şekilde belirlenmiştir:

- ***public void setTalkParameters():*** PublicationAgvent'in, Kayıtçı ile haberleşmesi sırasında kullanılabilecek metot çağrıları ve bu çağrılarda kullanılacak parametrelerin tanımlandığı metoddur.
- ***public String[] Talk(String question, String parameter***): PublicationAgvent'in, Kayıtçı ile haberleşmesi için kullanılan metod çağrısıdır. Bu metod çağrısında kullanılabilecek parametreler, agventin getTalkParameters() metodundan temin edilerek karşılıklı haberleşme sağlanır.
- ***public void runFirstAgventServer():*** PublicationAgvent ulaştığı ilk agvent sunucu üzerinde, kendi üzerine yapılan kayıt ölçütlerini değerlendirerek, kendini yönlendireceği Kayıtçıları seçer ve yönlendirme işlemine başlar.
- ***public void runLastAgventServer():*** Üretilen örnek PublicationAgvent'in son agvent sunucu üzerinde çalıştırılması istenen herhangi bir ölçüt tanımlaması yapılmamıştır. Ancak örnek bir ölçüt tanımlaması şu şekilde yapılabilir; "PublicationAgvent, sempozyumda sunulan bir bildiriyle ilgili bilgileri içermekte olup, sadece üniversite kütüphanelerine ulaştırılması istenmektedir".
- ***public void runSubscriber():*** Üretilen PublicationAgvent kendi içerdiği bilgiyi/yayını pazarlamayı amaçlamaktadır. PublicationAgvent Kayıtçı üzerine ulaşınca, kayıtçının kendisine sorduğu sorulara cevap vererek içerdiği bilgiyi/yayını pazarlamaya çalışmaktadır.

8.1.2. PublicationAgvent'in Sınıf Değişkenlerinin Belirlenmesi

Yeni bir agvent oluştururken, bu agventin üretim amacına uygun olarak agventin sahip olacağı özellikleri ve nitelikleri belirten sınıf değişkenlerinin uygun şekilde

tanımlanması gerekmektedir. Örnek uygulamada geliştirilen *PublicationAgvent* sınıfının değişkenleri aşağıdaki şekilde belirlenmiştir.

Bu değişkenler agventin etkin şekilde çalışmasında kullanıldığı gibi, Kayıtçıların sisteme kayıt olurken agvent üzerinden bu alanlar üzerinden de filtreleme yapılabilmesine olanak sağlamaktadır.

```
private String name;          // yayının ismi
private int pages;            // yayının sayfa sayısı
private String publisher;     // yayıncı kuruluş
private Date publishDate;     // yayın tarihi
private int ISBN;             // ISBN numarası
private float listPrice;      // liste fiyatı

private Vector author;        // yazar isimleri
private String[] references;  // yayının kullandıgı referanslar
private String[] TOC;         // yayının icerigi (table of contents)
```

8.1.3. PublicationAgvent'in Metotların Belirlenmesi

Geliştirilecek agventin niteliklerini, kapasitesini belirleyen onun davranışlarını gösteren metotlarıdır. Yeni bir agventin geliştirilmesinde üç tip metot kullanılmaktadır. Bunlar:

1. *Nesnesel metotlar:* Üretilen agventin, sınıf değişkenlerine erişimi sağlanan atayıcı (setter) ve alıcı (getter) metotlarıyla beraber kullanılan yapılandırıcıları (constructor) içermektedir.
2. *Soyut Metotlar:* Agvent sınıfının soyut metotları olan *Talk(), setTalkParameters(), runFirstAgventServer(), runLastAgventServer()* ve *runSubscriber()* metotlarıdır.
3. *Filtrelenebilir Metotlar:* Kayıtçıların, kayıt ölçütlerini oluşturmaları için kullanabilecek metotlar.
4. *Göreve Özel Metotlar:* Agventin davranışlarını ve kapasitesini belirleyen metotlardır. Bu metotlar agventin Kayıtçı üzerindeki haberleşmelerini sağlamakta ve kendisine yüklenen görevi yerine getirmede kullanılmaktadır.

Örnek uygulamada geliştirilen *PublicationAgvent* sınıfının üçüncü tipteki filtrelenebilir metotları ve bunların işlevlerini kısaca şu şekilde açıklanabilir.

- ***private boolean referenceContains(String auth):*** Parametre olarak gönderilen bir yazar adının, yayının referanslarında faydalanıp faydalanılmadığını kontrol eden bir metot çağrısıdır *(örneğin referenceContains ("Tanenbaum")).*

- ***private boolean TOCContains(String content):*** Parametre olarak gönderilen bir anahtar kelimeciğin, yayının içerik kısmında kullanılıp kullanılmadığını kontrol eden metot çağrısıdır *(örneğin TOCContains ("distributed agent")).*
- ***private boolean authorContains(String auth):*** Parametre olarak gönderilen bir yazar adının, yayının yazarları arasında olup olmadığını kontrol bir metot çağrısıdır *(örneğin authorContains("Ozgur Koray SAHINGOZ")).*
- ***private float WholesalePrice(int amount, String destination):*** Yayının belirli bir miktarının gönderim yerine bağlı olarak fiyatının ne olacağını hesaplayan metot çağrısıdır *(örneğin WholesalePrice(120, "İstanbul"))*

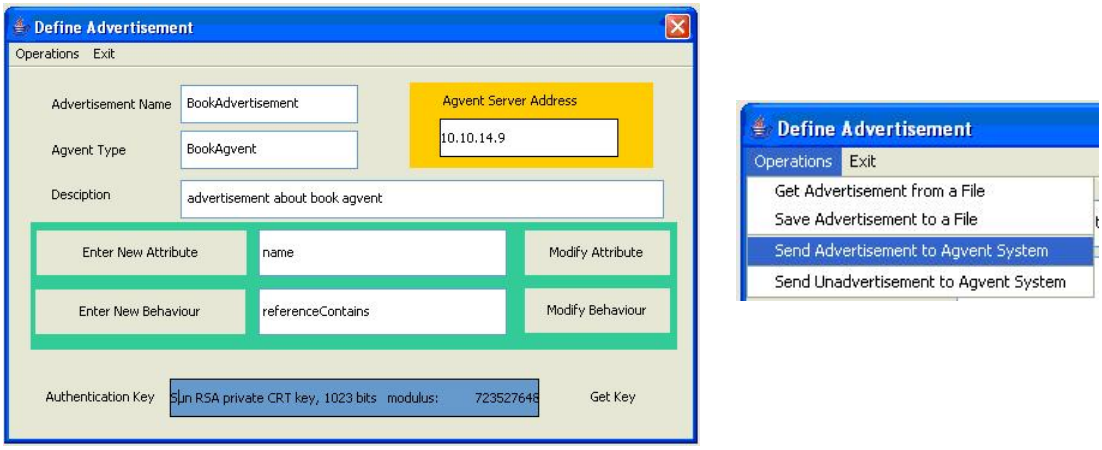
PublicationAgvent sınıfının dördüncü tipteki göreve özel metotlar ve bunların işlevlerini kısaca şu şekilde açıklanabilir.

- ***public String getReferenceText(String authorName):*** Parametre olarak gönderilen bir yazar isminin kullanıldığı referansların geriye döndürüldüğü metot çağrısıdır.
- ***public String getReferenceNumber(String authorName):*** Parametre olarak gönderilen bir yazar isminin kullanıldığı referansların numaralarının geriye döndürüldüğü metot çağrısıdır.
- ***public String getReferenceSentence(String referenceNumber):*** Parametre olarak gönderilen bir referans numarasının kullanıldığı cümleciğin geriye döndürüldüğü metot çağrısıdır.
- ***public String getSentenceBeforeRef(String referenceNumber):*** Parametre olarak gönderilen bir referans numarasının, kullanıldığı cümleden önceki cümleyi geriye dönderen metot çağrısıdır.
- ***public String getSentenceAfterRef(String referenceNumber):*** Parametre olarak gönderilen bir referans numarasının, kullanıldığı cümleden sonraki cümleyi geriye dönderen metot çağrısıdır.
- ***public String getPublicationInformation():*** Yayınla ilgili telefon numarası, faks numarası, web sayfası, yayıncı kuruluş gibi gerekli erişim bilgilerinin geriye döndürüldüğü metot çağrısıdır.
- ***public String getTOCContains(String searchWord):*** Parametre olarak gönderilen bir kelimenin kullanıldığı içerik (table of contents) satırının geriye döndürüldüğü metot çağrısıdır.
- ***public String getFirstParagraph(String sectionNumber):*** Parametre olarak konu numarasının verildiği bir bölümün ilk paragrafının parametre olarak geriye döndürüldüğü metot çağrısıdır.

- ***public String getReferencesIn(String sectionNumber):*** Parametre olarak konu numarasının verildiği bir bölümde faydalanılan referans numaralarının geriye döndürüldüğü metot çağrısıdır.
- ***public String getSentencesContains(String searchWord):*** Parametre olarak gönderilen bir kelimenin geçtiği cümlelerin geriye döndürüldüğü metot çağrısıdır.
- ***public String getKeywords():*** Yayındaki anahtar kelimelerin geriye döndürüldüğü metot çağrısıdır.
- ***public String getAbstract():*** Yayının özetinin geriye döndürüldüğü metot çağrısıdır.

8.1.4. PublicationAgvent Duyurusunun Yapılması

Üretilen agventin yayımlanmasından önce, filtrelenebilir özelliklerinin ve davranışlarının sisteme ilan edilmesi/duyurulması gerekmektedir. ABDES Sisteminde Şekil 8.2.a'da görüldüğü gibi, bir duyurunun tekil tanımlayıcısı, ilan edilen agvent tipi, bağlantı kurulacak agvent sunucunun adresi gibi bilgiler kullanıcı ara yüzünden temin edilerek duyuru mesajı oluşturulur. Oluşturulan duyuru mesajı daha sonra kullanılmak üzere yayımcının makinesinde saklanabileceği gibi, saklanan duyurularda alınarak güncellenebilir. Bu işlemler içinde Şekil 8.2.b'de gösterilen menü komutları kullanılır



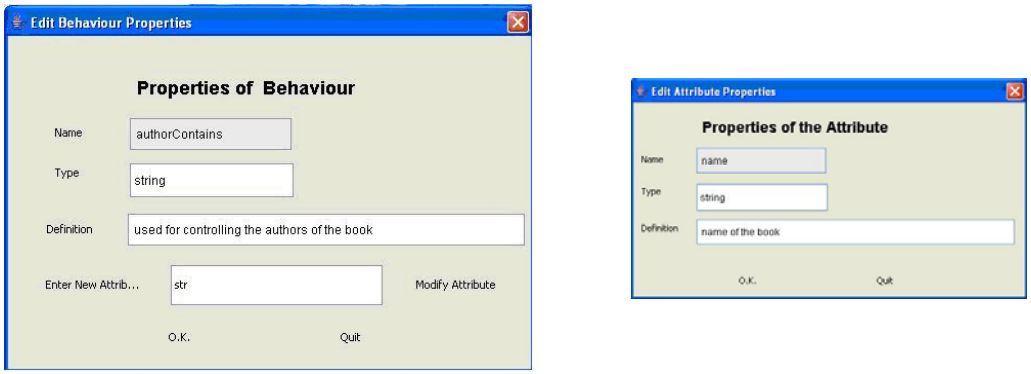
a) Duyuru Mesajı Tanımlama Ekranı

b) Duyuru Ekranı Operasyonları

Şekil 8.2 : Duyuru Oluşturma Ana Ekranları

Duyuru mesajını oluştururken ilan edilecek olan davranışlar Şekil 8.3.a'daki gibi bir kullanıcı ara yüzü üzerinden tanımlanır ve bu ara yüz vasıtası ile agventin metotlarının isimleri, işlevleri ve gönderilen parametre tipleri belirtilir. Agventin

özellikleri de benzer şekilde Şekil 8.3.b'deki gibi kullanıcı ara yüzü üzerinden tanımlanarak agventin sınıf değişkenlerinin isimleri, tipleri ve işlevleri belirtilir.



a) Agventin Davranışlarının Tanımlanması

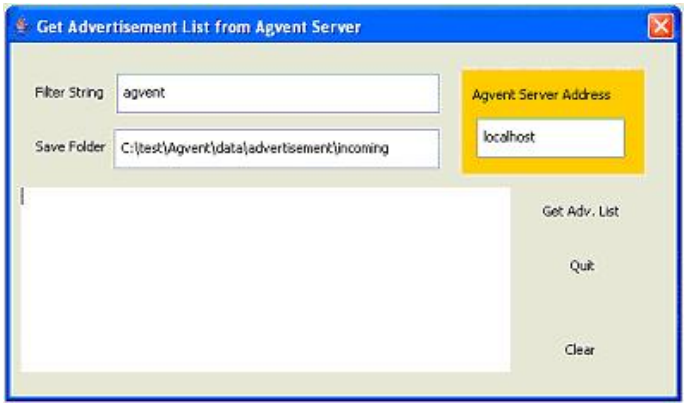
b) Agventin Özelliklerinin Tanımlanması

Şekil 8.3 : Duyuru Oluşturma Yardımcı Ekranları

Duyuru mesajı oluşturulduktan sonra Şekil 8.2.b'deki menü erişimi ile bağlı bulunulan Agvent Sunucuya gönderilerek agventin yayımlanması işlemi başlatılır.

8.1.5. Kayıtçıların Duyuru Listesini Almaları

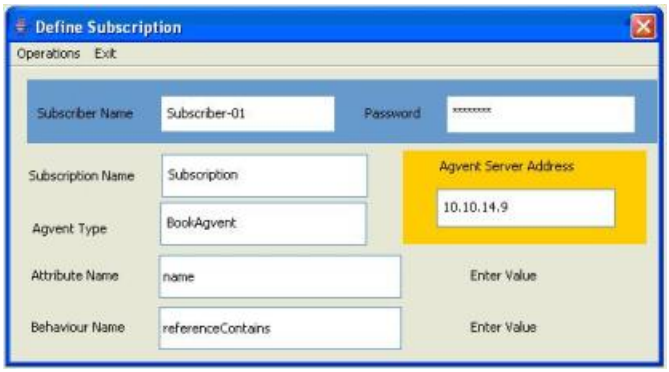
Bir Kayıtçı ABDES Sistemine *Kayıt* mesajı göndermeden önce hangi agventlerin sisteme duyurulduğunu, bu agventlerin filtrelenebilir özelliklerinin ve davranışlarının neler olduğunu sisteme gönderdiği bir *DuyuruListesiAl* mesajı ile öğrenir. Eğer Kayıtçı bütün duyurular ile ilgilenmiyorsa, istek duyduğu duyuruları bir filtreleme ile azaltır. Kayıtçı bu *DuyuruListesiAl* mesajını oluşturmak için Şekil 8.4'teki ekrandan faydalanmaktadır. Kayıtçı tarafından alınan duyurular, Kayıtçı makinesine tanımlanmış bir dizin içine kaydedilir ve daha sonra *Kayıt* mesajlarının oluşturulmasında bu duyurulardan faydalanılır.



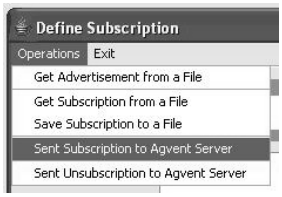
Şekil 8.4 : Duyuru Listelerini Alma Ekranı

8.1.6. Kayıt Mesajının Oluşturulması

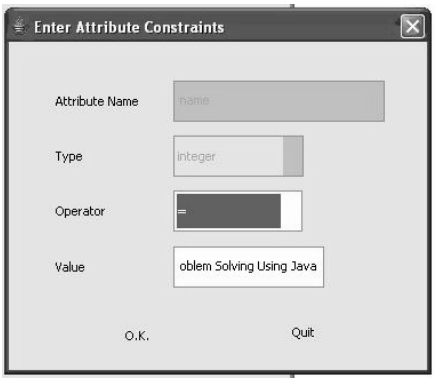
Bir Kayıtçı, sistemden istediği ölçütlere sahip agventlerin kendine ulaşması için bu ölçütleri tanımlayan bir kayıt mesajını sisteme göndermesi gerekmektedir. Bu *Kayıt* mesajı Şekil 8.5.a'da görüldüğü gibi, tekil tanımlayıcısı, kayıt olunacak agventin tipi ile özellik ve davranış ölçütleri alınarak oluşturulur. Oluşturulan kayıt mesajı daha sonra kullanılmak üzere Kayıtçının makinesinde saklanabileceği gibi, saklanan kayıtlarda alınarak güncellenebilir. Bu işlemler içinde Şekil 8.5.b'de gösterilen menü komutları kullanılır



a) Kayıt Mesajı Tanımlama Ekranı



b) Kayıt Ekranı Operasyonları



c) Özellik Filtresi Oluşturma Ekranı

d) Davranış Filtresi Oluşturma Ekranı

Şekil 8.5 : Kayıt Ekranları

Kayıt mesajını oluşturulurken, özellikler üzerine oluşturulacak ölçütler Şekil 8.5.c'deki kullanıcı ara yüzü üzerinden tanımlanır. Bu tanımlama sırasında daha önceden sisteme tanıtılmış olan karşılaştırma operatörleri kullanılır. İhtiyaç durumunda bu operatörler daha da genişletilebilir. Benzer şekilde kayıt olunacak agventin davranışları üzerine tanımlanacak ölçütlerde Şekil 8.5.d'deki kullanıcı ara yüzü üzerinden halledilir. Oluşturulan kayıt mesajı Şekil 8.5.b'deki menü erişimi ile bağlı bulunulan Agvent Sunucuya gönderilerek sisteme kayıt olunur ve sistemden bir agventin Kayıtçıya ulaşması beklenir.

8.1.7. Kayıtçı Üzerinde Haberleşme

Kayıtçının, amacına uygun olarak kendine ulaşan agventle konuşması/haberleşmesi gerekebilir. Bu haberleşme neticesinde kayıtçı bir karara ulaşarak ilgili yayını almaya veya almamaya karar vermesi beklenmektedir.

Kayıtçı makinesi üzerinde, Kayıtçı ile kendine ulaşan *PublicationAgvent* arasında yapılabilecek bazı diyalog örnekleri Şekil 8.6'da gösterilmiştir. Bu örneklerde

- İlk bloktaki ölçüt, Kayıtçının sisteme kayıt olurken kullandığı ölçüti, yani kendine ulaşmasını istediği agventin özelliklerini ifade etmektedir.
- İkinci bloktaki kısımda ise Kayıtçı ile agvent arasındaki diyalog gösterilmektedir. Kayıtçı kendine ulaşan agventten sahip olduğu yayınlarla ilgili belirli bilgileri görmek istemekte, ve bunlara göre de bu yayını almaya veya almamaya karar vermektedir.

"Carzaniga" ya Referans Verilen Yayınları Görmek İstiyorum

"Carzaniga" nın geçtiği Referansı göster
"Carzaniga" nın Yayınlarına Verilen Referans Numaralarını/Kodlarını Al
Referans Verilen Cümleyi/Cümleleri Al (Referans Kodu İle Erişim)
Referansımın Verildiği Cümleden Önceki Cümleyi Getir. (Referans Kodu İle Erişim)
Referansımın Verildiği Cümleden Sonraki Cümleyi Getir. (Referans Kodu İle Erişim)
Yayının Erişim Bilgilerini Al
Agventi Sonlandır

a) İsme Verilen Referansa göre Kayıt ve PublicationAgvent ile haberleşme

İçeriğinde(Table Of Contents) "Distributed Systems" Geçen, Kitap Olarak Yayımlanan Yayınları Görmek İstiyorum

"Distributed System" Geçen Konunun Başlığını Göster
Konunun Başlangıç Paragrafını Göster
O Konuda Faydalanılan Referansları Göster
Yayının Erişim Bilgilerini Al
Agventi Sonlandır

b)Kitap türünden bir akademik yayının içeriğinde geçen metne göre kayıt ve PublicationAgvent ile haberleşme

Anahtar Kelimelerinin Arasında "Multiagent" Geçen Bildiri Olarak Yayımlanan Yayınları Görmek İstiyorum

İçeriğinde "Multiagent" Geçen Cümleyi Göster.
Diğer Anahtar Kelimeleri Göster
Yayının Özetini Göster.
Yayının Erişim Bilgilerini Al
Agventi Sonlandır

c) Bildiri türünden bir akademik yayının Anahtar Kelimelerinde Kullanılan kelimeye göre kayıt ve PublicationAgvent ile haberleşme

Şekil 8.6 : Kullanılabilecek Kayıt Mesajları ve Haberleşme Örnekleri

Kayıtçının Şekil 8.6.a'da görülen örnek kayıt işlemini gerçeklemek için şu şekilde bir kayıt mesajı hazırlaması gerekmektedir.

```
Subscription sub = new Subscription("Pagv_Subscription1","PublicationAgvent");
Parameter par= new Parameter("Author","String","Carzaniga");
BehaviourConstarint bc = new BehaviourConstarint("referenceContains","String","=","true",par);
sub.addBehaviourConstarint(bc);
AgventServer.subscribe("Subscriber1","test1",sub);
```

Geliştirilen grafiksel ara yüz bu kayıt oluşturma işlemlerini daha kolay ve hatasız şekilde oluşturulmasına olanak sağlamaktadır.

Bu kayıt ölçütlerine uyan bir *PublicationAgvent* Kayıtçıya ulaşınca, Kayıtçı bu agventle haberleşerek, ilgili yayını alıp almamaya karar verecektir. Kayıtçı ile *PublicationAgvent* arasında geçen diyalogun programsal olarak şu şekilde ilerlediğini düşünülebilir. Öncelikle Kayıtçı, hangi haberleşme parametrelerini kullanabileceğini agventin *getTalkParameters* metodunu çağırarak öğrenir.

Pagv.getTalkParameters();
Return:

getReferenceText, return Reference Text; parameter, String, Author Name
getReferenceNumber, return Reference #; parameter, String, Author Name
getReferenceSentence return Reference Sentence; parameter, String, Reference #
*
*

Kayıtçı, bu haberleşme parametrelerini aldıktan sonra kendi amacına uygun olarak *PublicationAgvent* ile konuşmaya başlar. Bu karşılıklı konuşma aşamasında agvente mesajı Talk metodu vasıtası ile gönderir. Agvent kendine gelen bu mesajı inceleyerek kendi üretilme amacına uygun olarak Kayıtçıya cevap verir. Örnek bir etkileşimin şu şekilde cereyan ettiği düşünülebilir.

Pagv.Talk("getReferenceText", "Carzaniga");
Return:

A. Carzaniga, D. S. Rosenblum, and A. L. Wolf. Achieving scalability and expressiveness in an internet-scale event notification service. In Proceedings of the Nineteenth Annual ACM Symposium on Principles of Distributed Computing (PODC 2000), pages 219-227, Portland, Oregon, July 2000.

A. Carzaniga, D. S. Rosenblum, and A. L. Wolf. Design and evaluation of a wide-area event notification service. ACM Transactions on Computer Systems, 19(3):332-383, 2001.

Pagv.Talk("getReferenceNumber", "Carzaniga");
Return:
3
4
Pagv.Talk("getReferenceSentence", "2");
Return:

Past work has created numerous publish/subscribe systems, in industry and in academia [27, 2, 8,26, 12, 29, 10, 36].

Pagv.Talk("getSentenceBeforeRef", "2");
Return:

Most publish/subscribe architectures employ an application-level broker network (i.e., overlay) to perform content-based routing of events at application-level.

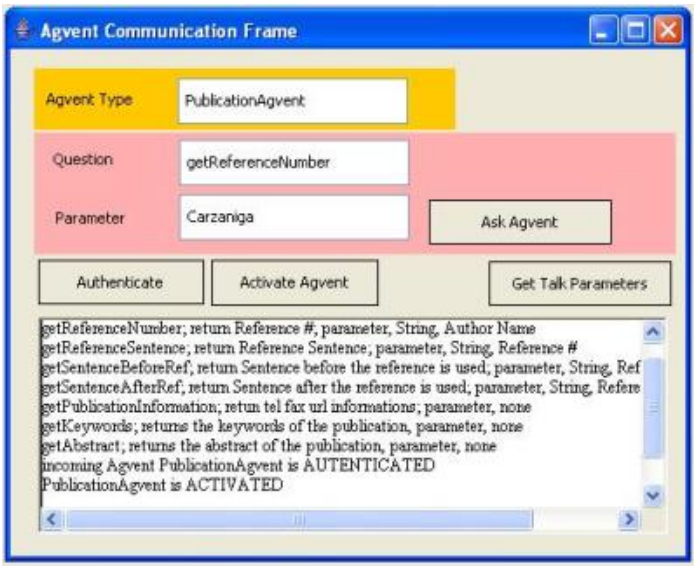
Pagv.Talk("getSentenceAfterRef", "2");
Return:

With the increased availability of powerful mobile computing devices like laptops and IPAQs, and the widespread deployment and use of wireless data communications, there is a pressing need to extend such middleware to the mobile computing domain.

Pagv.Talk(“getPublicationInformation”,””);
Return:

{ networks-opportunistic, author = "Yuan Chen","Karsten Schwan",
title = "Opportunistic Overlays: Efficient Content Delivery in Mobile Ad Hoc Networks", tel = “00 -1-512-2313129”, fax = “00 -1-512-2312842”url = "citeseer.ist.psu.edu/730201.html" price = "$15"}

Bu haberleşme işlemi için Şekil 8.7’deki gibi bir grafiksel kullanıcı ara yüzünden faydalanılarak karşılıklı etkileşim kolaylaştırılmaktadır. Her üretilen agventin Kayıtçı ile konuşması gerekmeyebilir. Eğer Kayıtçı, doğrudan agventten edinmesi gereken bir bilgi var ise bu etkileşimden faydalanılır.



Şekil 8.7 : Agvent Haberleşme Ekranı

9. TEST SONUÇLARI

Dağıtılmış sistemlerin performans ölçümlerinin değerlendirilmesi gerçek bir uygulamanın çalıştırılması veya bir simulator üzerinde testlerin yapılmasıyla sağlanabilir. Her ne kadar uygulamanın dağıtılmış ve geniş ölçekli bir platformda çalıştırılması ile alınan ölçümler gerçek değerleri verse de büyük ölçekli dağıtılmış sistemlerin kurulması, gerekli konakların eklenmesi, ağ genişliklerinin ayarlanması, gerçek verilerin temin edilmesi (üretilmesi) ve sistemin performansının ölçülmesi çok zordur. Onun yerine tez çalışmasının sonucunun geliştirilen bir ABDES Sistemi Simulatörü (AgvSim) üzerinde yapılmasının faydalı olacağı değerlendirilmiştir. Büyük ölçekli bir sistemin simülasyonunun yapılmasında gerçekçi verileri almak ve uygun ağ topolojisini oluşturarak gerek duyuru/kayıt mesajlarının gerekse agventlerin dağıtım sürelerinin hesaplanması amaçlanmıştır. Bu nedenle öncelikle Agvent Sunucular üzerinde yapılan temel işlemlerle ilgili değerler hesaplanarak, bu değerlerin parametrelere göre olan değişimleri gösterilmiştir.

ABDES Sisteminin gerçeklenmesinde ölçeklenebilir ve etkin dağıtıma olanak sağlayan bir sistem geliştirilmesi amaçlanmıştır. Sistem özellikleri itibarı ile incelendiğinde daha önce geliştirilen olay sistemleri ile ciddi farklılıkları olduğu için sistem performansının bu sistemlerle birebir karşılaştırılması uygun olmamaktadır. Geliştirilen diğer olay sistemlerinde öncelikli amaç mümkün olduğunca hızlı şekilde üretilen olay verisinin olay tüketicilere ulaştırılmasıdır. Bu sistemlerde olay verileri basit yapıda geliştirilmiş küçük boyutlu verilerdir. ABDES Sisteminde ise öncelikle verinin gizlenmesi sağlanmakta ve bunun için bir etmen yapısı kullanılmaktadır. Geliştirilen etmen uygulama örneğine bağlı olarak büyük boyutlu bilgiler içerdiği için üretilen ve yayımlanan agventin boyutu da büyümektedir. Bunun sonucu olaraktan agventin hedef tüketicilere ulaşması daha fazla süre almaktadır.

Agventlerin dağıtım ağaçları, kendi üzerilerine yapılan kayıt mesajları ile belirlenmektedir. Üretilen mesajların hedef makineler ulaşmasında ve sistem performansının test edilmesinde dağıtım sürelerinin hesaplanması esas alınmıştır. Dağıtım süresi iki kapsamda incelenmektedir; *duyuru/kayıt mesajlarının dağıtım süreleri* ve *agventlerin dağıtım süreleri*. Sistemin geliştirim amacı agventlerin dağıtım süresi olan “üretilen bir agventin sistemdeki kayıtlı tüm Kayıtçılara ulaştırılması için gerekli sürenin” en aza indirilmesidir. Dağıtım sürelerinin hesaplanmasında şu ölçütler önemlidir;

Atlama Sayısı: Agventin bir Kayıtçıya ulaşmak için üzerinden geçmesi gereken sistem bileşeni (Agvent Sunucu ve Kayıtçı) sayısıdır. Her atlama sayısı agventlerin dağıtım süresini artırdığı için bu sayı mümkün olduğunda az olacak şekilde ağ topolojisinin yapılandırılması önemlidir.

Bant Genişliği: Geliştirilen sistem, yapısı gereği çalıştığı ağ üzerinde başka uygulama programlarıyla aynı fiziksel ağ yapısını paylaşmaktadır. Bu nedenle agvent dağıtım ağacı mümkün olduğunca az ağ genişliğini kullanmalı, mümkün olduğunca az agvent, duyuru ve kayıt mesajı dağıtımı yapmalıdır. Kullanılan duyuru mekanizması sayesinde gereksiz mesajların dağıtımının en aza indirilmesi sağlanmış olup, agvent dağıtımındaki temel etkenlerin başında kullanılabilen ağ genişliği olduğu değerlendirilmektedir. Sistem bir yerel alan ağı üzerinde test edildiği için ağ genişliği sabit 100 Mbit/s olarak kullanılmıştır.

Konak Performansı: Sistem üzerinde dağıtılan mesajların yönlendirilmesinde Agvent Sunucular üzerinde yapılması gereken işlemler mesajların dağıtımında büyük süreler almaktadır. Bu nedenle sistem Tablo 9.1'de özellikleri belirtilen üç farklı konak makinede test edilerek genel performans testinde bu makinelerin ortalamaları alınmıştır. Makinelerin işlem gücünün yüksek olması, konak makineler üzerinde yapılacak işlemleri hızlandıracağı için dağıtım süresine azaltmaktadır.

ABDES Sistemi için yapılan başarım ölçümleri özellikleri Tablo 9.1'de gösterilen ağ yapısı üzerinde gerçekleştirilmiştir. Sistemin performans ölçümünün yapılabilmesi için sistemin çalışması sırasında gerçekleştirilen alt işlemlerin başarım ölçümleri yapılarak sistemin bir bütün olarak başarım ölçüsü hesaplanmıştır. Bu alt işlemler devam eden bölümlerde incelenmektedir.

Tablo 9.1. Test İşlemlerinde Kullanılan Sistem Özellikleri

Özellik	Açıklama
Kullanılan Ağ Tipi	Yerel Alan Ağı (100 Mbit Eternet)
Ağ İşletim Sistemi	Windows NT
Bilgisayar Özellikleri	Pentium III 1000MHz İşlemci 512 Mb RAM 256 Kb Cep Bellek 133 MHz FSB Windows XP İşletim Sistemi
	Pentium IV 3.00 GHz İşlemci 512 Mb RAM 512 Kb Cep Bellek 800 MHz FSB Windows XP İşletim Sistemi
	Pentium Centrino 1.60 GHz İşlemci 512 Mb RAM 2 Mb L2 Cep Bellek 400 MHz FSB Windows XP İşletim Sistemi

9.1. Nesne Veri Tabanı İşlemleri

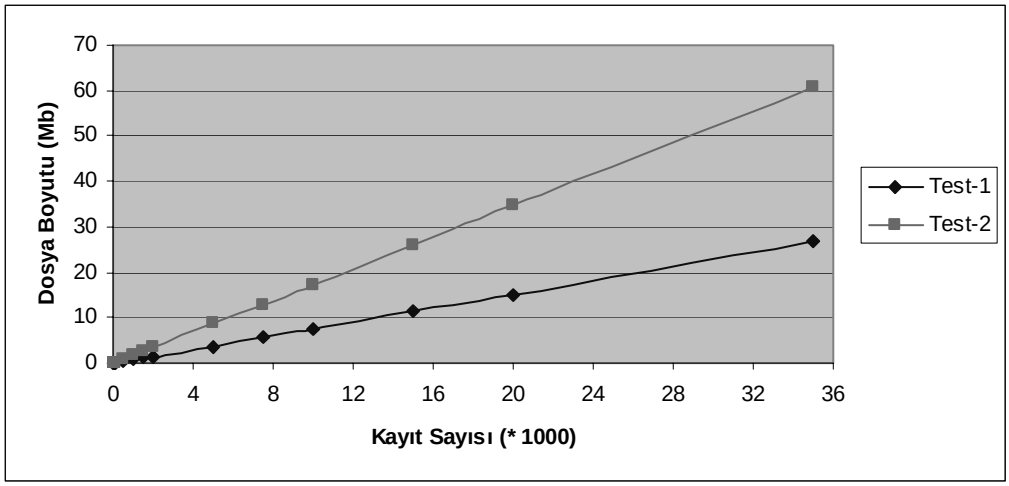
Sistem performansını etkileyen temel unsurlardan biri Agvent Sunucular üzerinde yapılan veri erişim işlemleridir. Duyuru mesajlarının, kayıt mesajlarının ve agventlerin dağıtım işlemlerinde Agvent Sunucular üzerinde sorgulama, tabloya ekleme ve tablodan silme işlemleri yapılmaktadır. Agvent Sunucu üzerinde bu işlemler ObjectDB nesne veritabanı üzerinde yapılmaktadır.

Duyuruların ve Kayıtların nesne veritabanında ne kadar yer aldıklarının hesaplanması için rasgele sayıda özellikleri ve metotları olan duyurular üretilerek saklanmıştır. Bu duyuruların nitelikleri Tablo 9.2'de gösterilmektedir.

Tablo 9.2. Üretilen Duyuruların Özellikleri

	Miktar	
Nitelik	**Test-1**	**Test-2**
Özellik Sayısı	0-5 adet	0-15 adet
Metod (Davranış) Sayısı	0-5 adet	0-10 adet
Metoda Gönderilen Parametre Sayısı	0-5 adet	0-5 adet

Bu şekilde rasgele üretilen duyuruların saklandığı duyuru tablosunun boyutu sahip olduğu duyuru sayısına göre değişmektedir. İçerdiği kayıt sayısına göre dosya boyutunun değişimi Şekil 9.1'de gösterildiği gibi doğrusal bir artış göstermektedir.



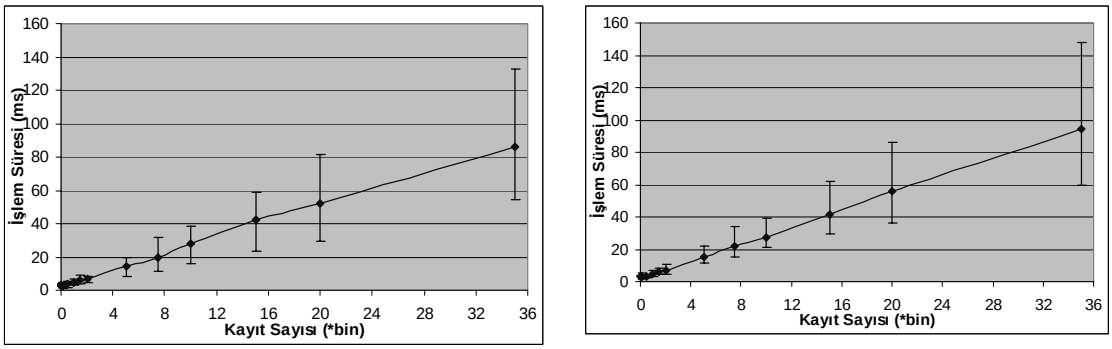
Şekil 9.1 : Duyuru Tablosunun Boyutu

Tablonun indeksli veya indeksiz olması dosya boyutunu etkilememektedir[33]. Sistem üzerinde nesne veritabanlarında saklanan bu verilerin konak makine üzerinde makul ölçüde yer işgal ettikleri öngörülmektedir. Veritabanının fazla geniş olmayacağı, PDA'ler ve yeni nesil cep telefonlarının bir AS olarak kullanılabileceği

[33] Dosya boyutunun değişikliği % 0,1 den daha az olduğu için dikkate alınmamıştır.

uygulamalarda da, önceden bir sistem kurumu gerekmeyeceği[34] için kolay kullanılır ve taşınabilir yapısı olacağı değerlendirilmektedir.

Gerek agventlerin gerekse duyuru/kayıt mesajlarının sistemdeki dağıtım sürelerinin hesaplanmasında, nesne veri tabanı üzerinde yapılan veri erişim sürelerinin bilinmesi gerekmektedir. Tez çalışmasında kullanılan ObjectDB nesne veri tabanına bir duyuru kaydının eklenmesi için gereken süre, veri tabanındaki duyuru nesnelerinin miktarına, işlemcinin özelliklerine ve bu veri tabanının indeksli olup olmadığına bağlıdır. İndekssiz bir nesne veri tabanına bir duyuru kaydını eklemek için ihtiyaç duyulan sürenin tablonun sahip olduğu nesne sayısına göre değişen grafiği Şekil 9.2.a'da sunulmuştur. Kayıt ekleme işlemi eğer indeksli bir tablo üzerinde yapılırsa, bir duyuru nesnesinin tabloya eklenmesi işleminin süresi Şekil 9.2.b'de gösterilmektedir.[35]



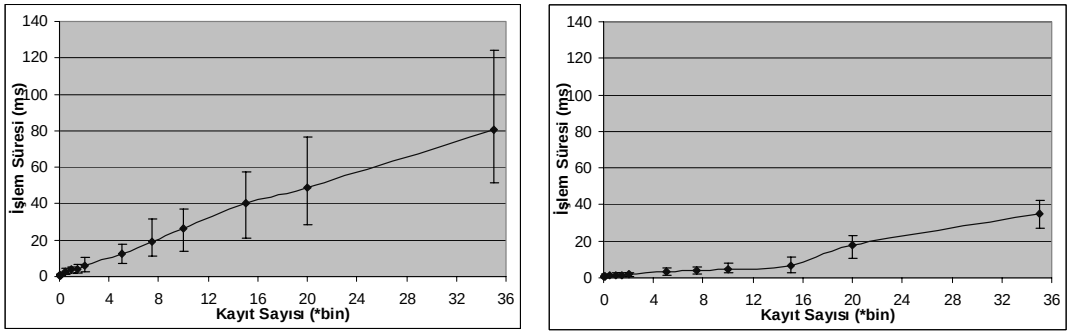
a) İndekssiz Tablo b) İndeksli Tablo

Şekil 9.2 : ObjectDB'ye Kayıt Ekleme Süreleri

Bu süreler daha önce belirtilen üç farklı bilgisayarda ayrı ayrı ölçülmüştür. Her ölçüm için on adet test işlemi yapılarak bu değerlerin ortalaması alınmıştır.

Yukarıdaki şekillerden de görüleceği üzere kayıt ekleme işleminin indeksli bir tablo üzerinde yapılması işlem süresini uzatmaktadır. Veri Tabanı Sistemlerinde de olduğu gibi, indeksli tablo kullanımı veri güncelleme işleminin süresini uzatırken, veri erişim süresini azaltmaktadır. ObjectDB'de indekssiz bir tablo üzerinde yapılan sorgulama işleminin süresi Şekil 9.3.a'da, indeksli bir tabloda yapılan sorgulama süresi ise Şekil 9.3.b'de gösterilmektedir.

[34] Bir Oracle veya SQL Server veri tabanının kurumu gibi.
[35] Nesne Veri Tabanına erişim işlemleri 10'ar kez, ve 10'ar adet verinin işlenmesi olarak (toplam 100 örnek üzerinden) test edilerek ortalaması alınmıştır.

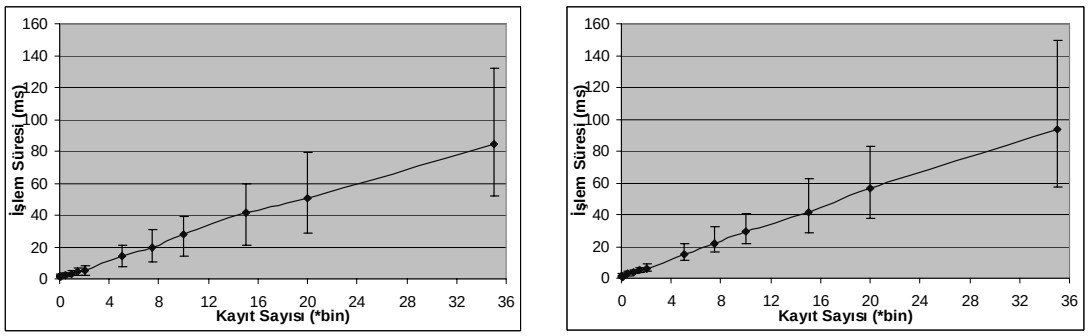


a) İndekssiz Tablo b) İndeksli Tablo

Şekil 9.3 : ObjectDB'de Sorgulama Süreleri

Şekil 9.3'teki grafiklerden de indeks yapısının sorgulama işlemlerini nasıl hızlandırdığı görülmektedir. Agventlerin sistem üzerinde dağıtımı sırasında, kendi üzerlerine yapılan kayıt mesajlarına ulaşmaları gerekmektedir. Bunun içinde yapılacak sorgulamanın hızlı şekilde neticelenmesi önemlidir. Bu nedenle tez çalışmasında indeksli tablolardan faydalanılarak agventlerin dağıtımı/yönlendirilmesi hızlandırılmıştır.

Duyuru_Yap ve Kayit_Ol işlemleri nesne veri tabanlarına veri eklerken, Duyuru_Sil ve Kayıt_Sil işlemleri de bu veri tabanlarındaki kayıtları silmektedir. ObjectDB üzerinden yapılan bir veri silme işleminin indekssiz bir tabloda ihtiyaç duyduğu süre Şekil 9.4.a'da, indeksli bir tabloda ihtiyaç duyduğu süre ise Şekil 9.4.b'de gösterilmektedir.



a) İndekssiz Tablo b) İndeksli Tablo

Şekil 9.4 : ObjectDB'den Kayıt Silme Süreleri

ABDES Sisteminde, agventlerin hızlı dağıtımı amaçlandığı için indeksli tablo yapıları kullanılmıştır. Her ne kadar gerek duyuruların yayımında gerekse kayıt mesajlarının dağıtımında bu tablo yapısı kayıt ekleme ve kayıt silme işlemlerinden ötürü işlem süresini az miktarda uzatsa da, sistem üzerinde daha yoğun şekilde

dağıtılacağı öngörülen agventlerin dağıtımında, sunucular üzerinde yapılacak sorgulama süreleri azalmaktadır. Dolayısıyla sistemin ana performans kriteri olarak düşündüğümüz agventlerin dağıtım süresi azalmaktadır.

Elde edilen test sonuçları, veri tabanlarının boyutlarının artmasına rağmen, veri erişim sürelerinin oldukça düşük olduğunu ve kalıcı olarak saklanacak verilerin bir nesne veri tabanında saklanmasının sistem performansını artırdığını göstermektedir.

9.2. Dağıtım İşlemleri

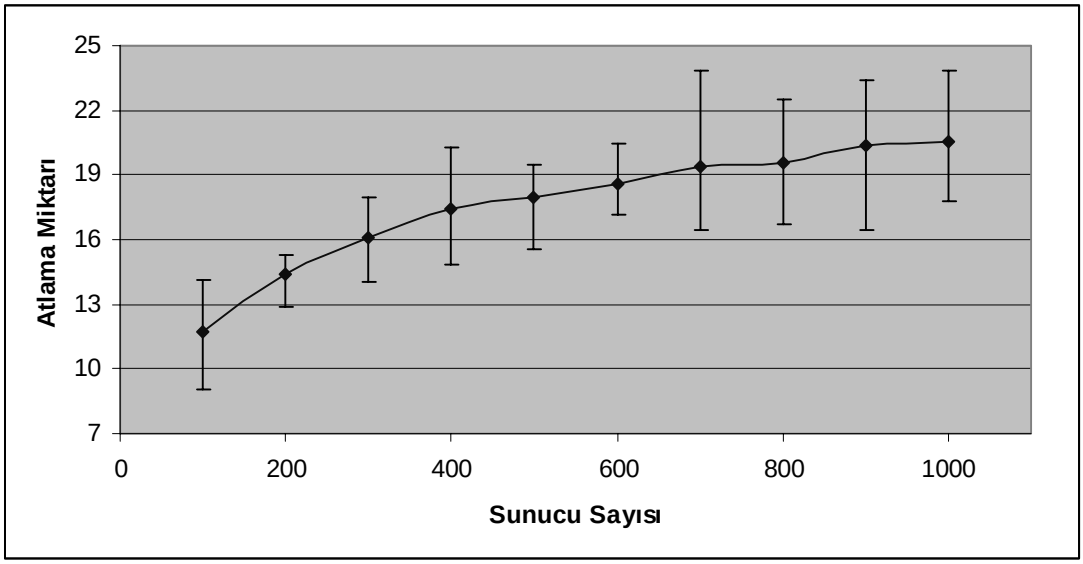
Sistemin performansının test edilebilmesi için farklı sayılarda agvent sunuculardan oluşmuş topolojilerdeki sistem örnekleri üzerinde testler yapılmıştır. Bunun içinde rasgele üretilmiş topolojileri oluşturmak amacıyla Boston Üniversitesi tarafından geliştirilen BRITE: Boston university Representative Internet Topology gEnerator [91, 92] sisteminden faydalanılmıştır. BRITE topoloji üreticisi esnek yapısı, geliştirilebilir ve taşınabilir olmasının yanı sıra kullanım kolaylığı ve diğer ağ simulatörleri ile de uyumlu veri üretebilmesi açısından tercih edilmiştir.

Sistemin performansının test edilmesi için 100 ile 1000 arasında ki agvent sunuculardan oluşan 10 farklı topoloji yapısı üzerinde duyuru/kayıt mesajlarının ve agventlerin dağıtım performansları incelenmiştir. BRITE topoloji üreticisi tarafından üretilen bu topolojilerin anlık farklılıklarını ortadan kaldırmak için her topolojiden 10 ar adet üretilmiş ve hesaplamalarda bunların ortalaması alınarak simulasyon çalıştırılmıştır.

Tablo 9.3. Test Topolojilerindeki Atlama Miktarları

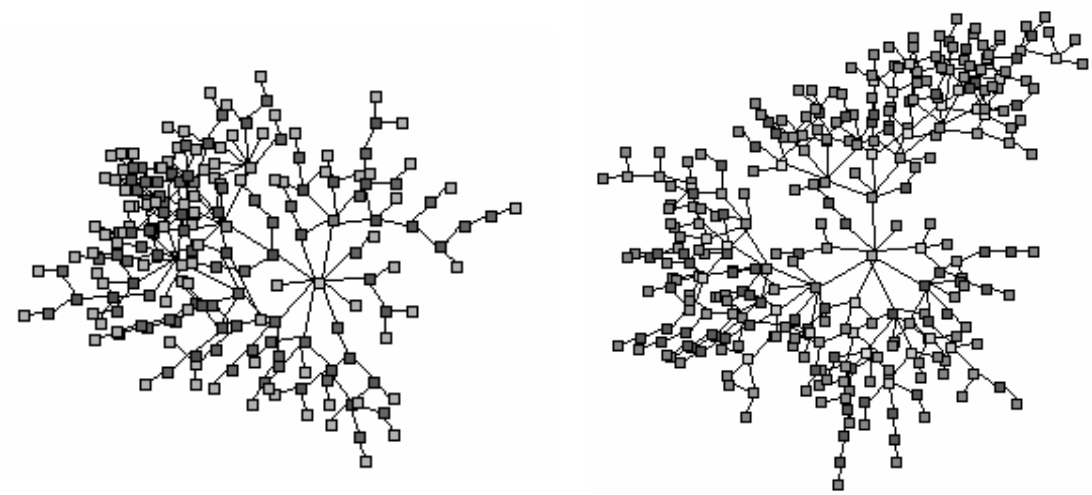
AS Sayısı	Ortalama Atlama Miktarı	Minimum Atlama Miktarı	Maksimum Atlama Miktarı
100	11,73	9,03	14.09
200	14,41	12,87	15.30
300	16,10	14,08	17.93
400	17,47	14,80	20.32
500	17,57	15,16	19.09
600	18,59	17,14	20.48
700	18,64	15,72	23.08
800	19,59	16,69	22.47
900	20,39	16,47	23.37
1000	20,58	17,75	23.86

Üretilen bu topolojilerdeki atlama miktarı dağıtım sürelerini etkileyecek temel etkenlerden olduğu için bu değerler tablo formunda Tablo 9.3'te bir grafik olarak ta Şekil 9.5'te gösterilmektedir.



Şekil 9.5 : Test Topolojilerindeki Atlama Miktarları

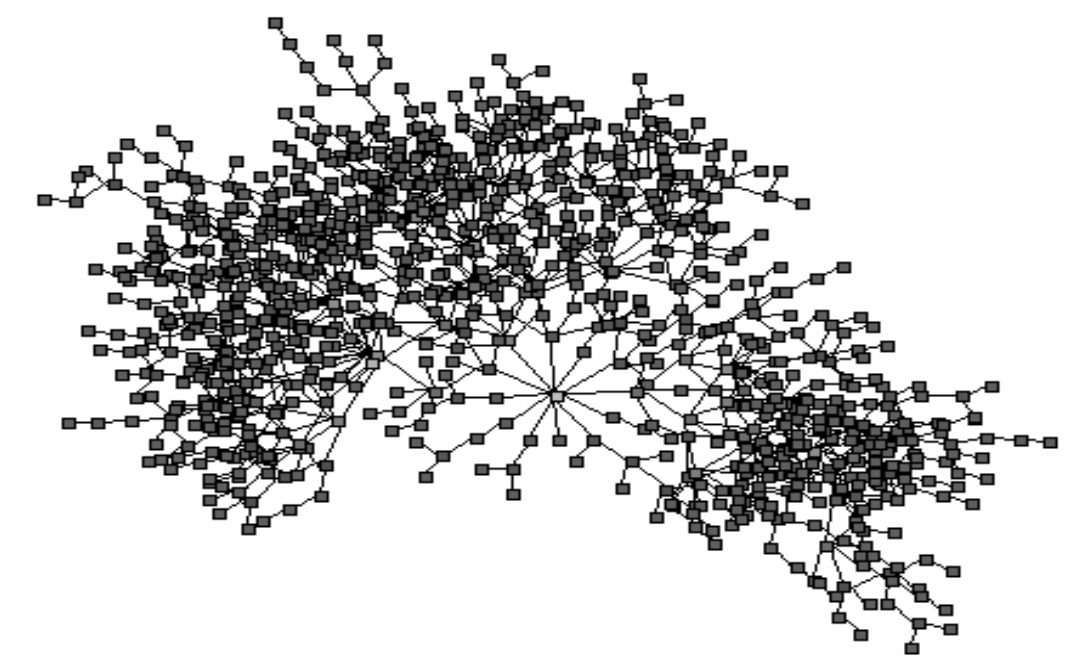
Bu şekilde üretilen 200 agvent sunuculu topoloji Şekil 9.6.a'da, 300 agvent sunuculu topoloji Şekil 9.6.b'de gösterilmektedir.



a) 200 Agvent Sunuculu Topoloji b) 300 Agvent Sunuculu Topoloji

Şekil 9.6 : 200 ve 300 Agvent Sunuculu Rasgele Üretilmiş ABDES Sistemi

Son olaraktan 1000 agvent sunucunun bulunduğu topoloji Şekil 9.7'deki gibi üretilerek test edilmiştir. Rasgele üretilen bu topolojilerin görselleştirilmesinde CAIDA, (the Cooperative Association for Internet Data Analysis) tarafından üretilen Otter [93, 94] arayüzünden faydalanılmıştır.



Şekil 9.7 : 1000 Agvent Sunuculu Rasgele Üretilmiş ABDES Sistemi

Sistem üzerinde gönderilen agventlerin dağıtım işlemlerinin test edilmesinde temel işlemler Agvent Sunucular üzerinde yapılmaktadır. Bu nedenle test ortamındaki Agvent Sunucuların sahip oldukları özellikler doğrudan dağıtım sürelerini etkilemektedir. Üretilen topolojilerdeki diğer temel kriterler ve Agvent Sunucudaki ilgili parametreler Tablo 9.4'te gösterilmektedir.

Tablo 9.4. Dağıtım Sürelerinin Hesaplanmasında Kullanılacak Ölçütler

Nitelik	**Değer Aralığı**
Duyuru Sayısı	5.000-15.000
Kayıt Sayısı	10.000-30.000
Komşu Agvent Sunucu Sayısı	1-7
Bir Agvent Sunucuya Kayıtlı Kayıtçı Sayısı	1-10
Sistemdeki Agvent Sunucu Sayısı	100-1000
Alternatif Yönlendirme Tablosu (AYT) Kayıt Sayısı	100-1000
Kayıtların Ölçüt Miktarları	
Özellik Sayısı	0-15
Metod (Davranış) Sayısı	0-10
Metoda Gönderilen Parametre Sayısı	0-5

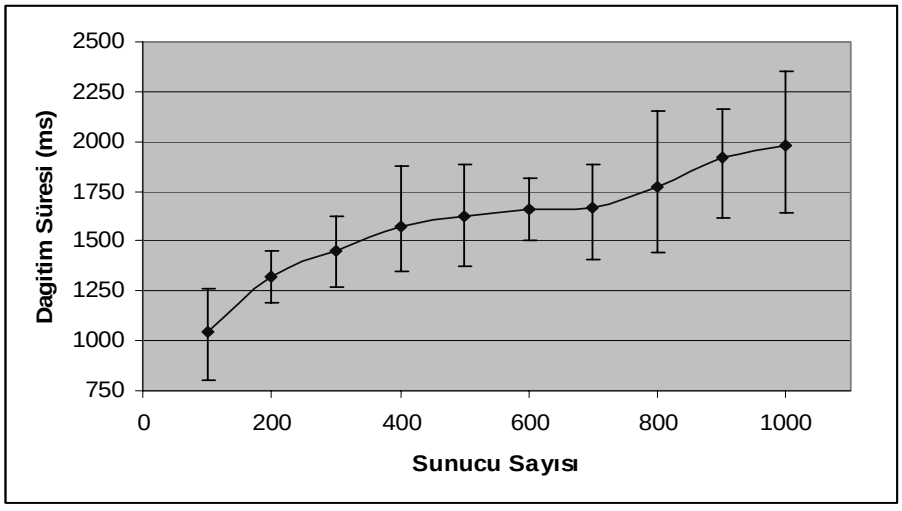
9.2.1. Duyuru Mesajlarının Dağıtım İşlemleri

ABDES Sisteminde üretilen bir duyuru mesajının sistem üzerindeki tüm Agvent Sunuculara dağıtılması gerekmektedir. Bu dağıtım yayın mekanizması ile gerçekleştirilmektedir. Duyuruların dağıtımı sırasında bir Agvent Sunucu üzerinde yapılan ardışık işlemler ve ortalama 10000 duyuru mesajının bulunduğu öngörülen JDO veri tabanında bu işlemlerin aldıkları süreler örnek olarak Tablo 9.5'te gösterilmektedir.

Tablo 9.5. Duyuru Mesajlarına Yapılan Temel İşlemlerin Süreleri

İşlem	Süre (milisaniye)
Duyuru Sorgulama	4.95
Dosya Erişim İşlemleri	25.10
Duyuru Ekleme	27.32
Komşu AS Bilgilerin Erişim	4.60
Komşu Agvent Sunucu ile RMI Bağlantısı Kurma	4.80
Duyuru Mesajının Gönderilmesi (100 Mbit/s)	4.31
AYT'nu Güncelleme	4.74

Daha gerçekçi değerlendirme sonuçlarının alınmasında ise sistem üzerindeki tüm Agvent Sunucular birer duyuru mesajı göndererek bu duyuru mesajlarının sistem üzerinde dağıtım süreleri hesaplanmıştır. Yapılan karşılaştırmalar sırasında dağıtım süresini daha önce bahsettiğimiz parametre değerlerin haricinde, sistemdeki Agvent Sunucu sayısı ve kullandığımız topolojideki atlama sayısı (hop count) da etkilemektedir. Buna göre üretmiş olduğumuz yedi farlı topolojideki duyuruların dağıtım performansları Şekil 9.8'de gösterilmektedir



Şekil 9.8 : Duyuru Mesajlarının Dağıtım Sürelerinin Karşılaştırması

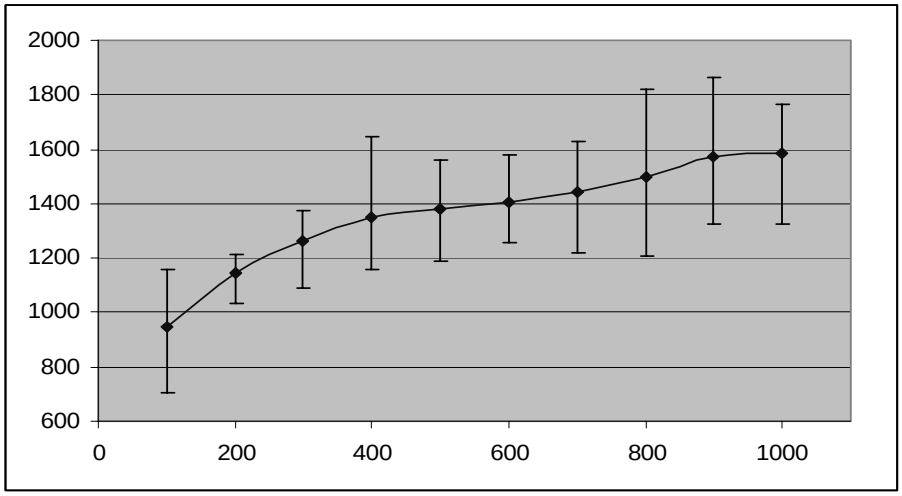
9.2.2. Kayıt Mesajlarının Dağıtım İşlemleri

Kayıtçı tarafından üretilen bir kayıt mesajı, sistem üzerinde dağıtımı sırasında ilgili duyuru mesajlarının çizdiği yoldan faydalanmaktadır. Duyuru mesajların kullanımı sayesinde kayıt mesajlarının dağıtımı hem hızlanmakta hem de gereksiz sayıda kayıt mesajı üretilmeyerek sistemdeki mesaj yoğunluğu azaltılmaktadır. Kayıt mesajlarının sistem üzerinde dağıtımı sırasında Agvent Sunucu üzerinde yapılan işlemler ve ortalama 10000 duyuru mesajı ile ortalama 10000 kayıt mesajının bulunduğu öngörülen JDO veri tabanında bu işlemlerin aldıkları süreler örnek olarak Tablo 9.6'da gösterilmektedir.

Tablo 9.6. Kayıt Mesajlarına Yapılan Temel İşlemlerin Süreleri

İşlem	Süre (milisaniye)
Kullanıcı Adı ve Şifre Kontrolü	0.92
Kayıt Tablosu Kontrolü	17.89
Kayıt Tablosuna Ekleme	55.53
Dağıtım Listesinin Seçimi için Duyuru Tablosundan Sorgulama	4.95
Dağıtım Listesini Oluşturma	2.0
RMI Bağlantısı Kurulumu	4.8
Kayıt Mesajının Gönderilmesi	8.91
Yönlendirme Tablosunu Güncelleme	4.74

Daha gerçekçi değerlendirme sonuçlarının alınması için sistem üzerindeki tüm Agvent Sunucularının üzerinde birer tane yayımcının ilgi duyulan agvent tipinden veri üretebildiği göz önüne alınmıştır. Bu şekildeki bir sistemde her Agvent Sunucun duyuru mesajlarının yayınladığı ortamdaki kayıt mesajlarının dağıtım süreleri hesaplanmıştır. Buna göre üretmiş olduğumuz on farlı topolojideki kayıt mesajlarının dağıtım performansları Şekil 9.9'da gösterilmektedir



Şekil 9.9 : Kayıt Mesajlarının Dağıtım Sürelerinin Karşılaştırması

Üretilen kayıt mesajlarının boyutunun duyuru mesajlarının boyutundan daha küçük olmasından dolayı kayıt mesajlarının dağıtım süreleri duyuru mesajlarının dağıtım sürelerinden daha düşük çıktığı görülmektedir. Aynı zamanda sistemdeki atlama sayıları da duyuruların atlama sayılarına yakın ama farklı değerlerdedir.

9.2.3. Agvent Dağıtım İşlemleri

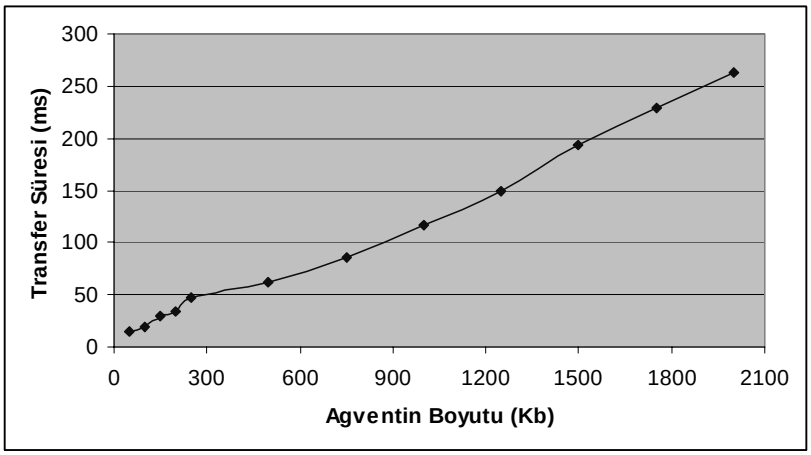
Agventler yapıları gereği içlerinde değerli verileri saklamaktadırlar. Kullanılan agvent modeline/örneğine göre agventin boyutu farklılıklar göstermektedir. Bu da agventin dağıtım süresini etkilemektedir. Örnek uygulamada gerçekleştirilen PublicationAgvent modeli özellikle akademik yayınların dağıtımında kullanılmak üzere geliştirilmiş olmasına rağmen, normal köşe yazıları, dergi yazıları ve kitaplar içinde kullanılabilmektedir. Agventlerin boyutunu etkileyen temel faktör içerdiği veridir. PublicationAgvent örneğinin içerdiği yayındaki kelime ve karakter sayısına göre yaklaşık boyutu Tablo 9.7'de gösterilmektedir.

Tablo 9.7. PublicationAgvent Boyutu Tablosu

Kelime Sayısı	Karakter Sayısı	Sayfa Sayısı	Yaklaşık Agvent Boyutu (Kb)
641	4301	Köşe Yazısı	4.60
1943	12658	Araştırma Yazısı (Dergi)	13.93
3116	17260	8 Sayfa Bildiri	22.35
4139	22706	10 SayfaBildiri	29.68
7504	40660	12 Sayfa Bildiri	53.81
44650	318920	Doktora Tezi	320.17
237300	1308750	50 Bildirili Semp. Kitabı	1700.00
325070	1600654	870 Sayfa Kitap (Comp. Netw. Tanenbaum)	2330.00

Agventlerin dağıtımında, agventlerin kaç AS üzerinden atlayarak Kayıtçıya ulaşacağı önemlidir. Bu sayının artması ağ üzerindeki yoğunluğun artmasına ve dağıtım süresinin uzamasına sebep olmaktadır.

Test işlemlerinde amaç en uygun koşullarda bir agventin sistem üzerindeki kayıtlı bütün ilgili kayıtçılara ulaştırılması olduğu için, bu işlemler sırasında ABDES Sistemi üzerinde her hangi çökme işleminin olmadığı öngörülerek yapılmıştır. 100 Mbit/s lik bir bant genişliğine sahip ağ üzerinde, üretilen agventin boyutuna göre iki sistem bileşeni arasındaki transfer süreleri Şekil 9.10'da gösterildiği gibi doğrusal olarak artmaktadır.



Şekil 9.10 : İki Konak Arası Agvent Gönderim Süresi

Agventlerin yönlendirme mekanizması gerek duyuru mesajlarının gerekse kayıt mesajlarının dağıtımından biraz farklıdır. Bir agventin dağıtımı sırasından AS üzerinde yapılan işlemler Tablo 9.8'de listelenmektedir.

Tablo 9.8. Agventlerin Dağıtımında Agvent Sunucu Üzerinde Yapılan İşlemler

Sıra No	İşlemler	Dolaşılan Agv. Sun.	Süre (ms)
1	Kuyruktan Agvent Alınması	İAS,AAS,SAS	11.1
2	Agvent Aktif hale getirilmesi	İAS,SAS	2.3
3	runFirstAgventServer'ın çalışımı	İAS	2.5
4	runLastAgventServer'ın çalışımı	SAS	2.0
5	Kayıt Tablosundan ilgili kayıtlar alınması[36]	İAS	4.95
6	Bir kayıt'ın özellik ölçütlerinin incelenmesi (ort 5 özellik)	İAS	1.3
7	Bir kayıt'ın davranış ölçütlerinin incelenmesi (ort 4 metod)	İAS	2.2
8	Kayıtçı Tablosunun İncelenmesi (ort 100 kayıtçı)	SAS	0.92
9	Yönlendirme Tablosunun güncellenmesi	İAS,AAS,SAS	5.3
10	Agvent Dağıtım Kuyruğunda Bekleme	İAS,AAS,SAS	29.2
11	RMI Bağlantısı Kurulması	İAS,AAS,SAS	4.8
12	Agventin Gönderilimi[37]	İAS,AAS,SAS	85.9

Bu tabloda ;

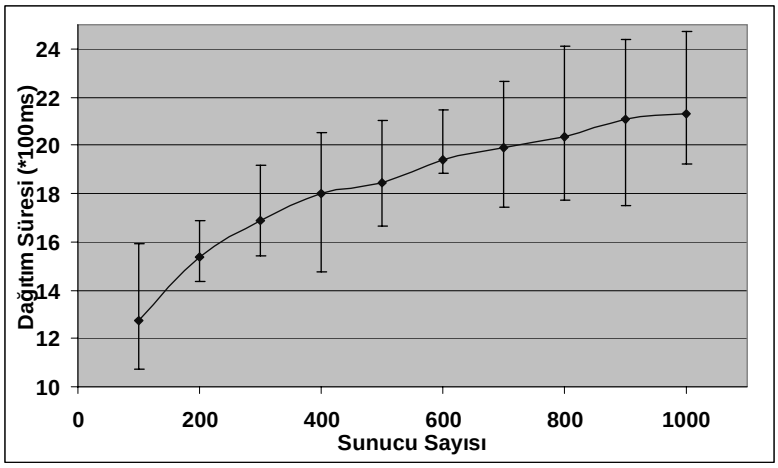
- **İAS :** Agventi ulaştığı ilk Agvent Sunucuyu,

[36] Sorgulama süresinin hesaplanmasında 9. Event Composition in Time-depend.b'de gösterilen indeksli tablodaki sorgulama süreleri referans olarak alınarak ortalama 10000 kayıt için olan değer tabloda gösterilmiştir.

[37] Agvent gönderim süresinin hesaplanmasında ent Distributed Systems, *In Proceedi*'de gösterilen iki konak arasındaki agvent gönderim süreleri referans olarak alınarak ortalama 750 Kb boyundaki agventin gönderim süresi tabloda gösterilmiştir.

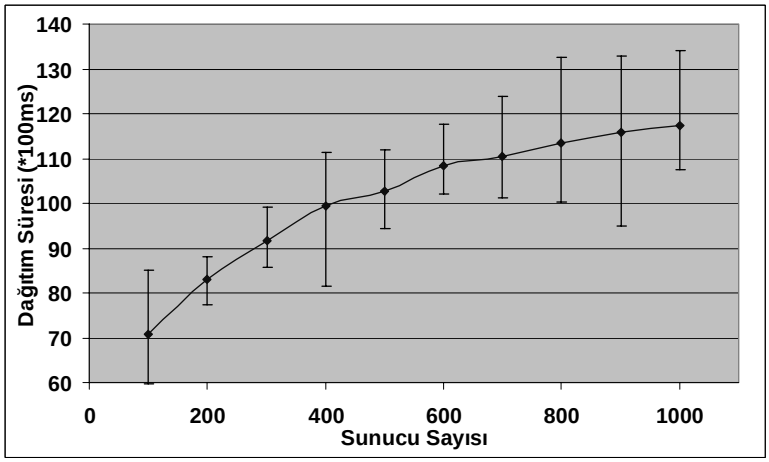
- **SAS:** Agventin Kayıtçıya ulaşmasından önceki son Agvent Sunucuyu,
- **AAS :** Agventin üzerinden geçtiği diğer Agvent Sunucuları ifade etmektedir.

Bu değerleri göz önüne aldığımızda 10-12 sayfalık bir bildirinin (Yaklaşık 50 Kb) tüm sisteme yayımlanması için geçen sürenin sistemdeki sunucu sayısına göre olan karşılaştırması Şekil 9.11'de gösterilmiştir.



Şekil 9.11 : Bir bildiri verisi içeren agventin dağıtım süresi

Uygulaması geliştirilen Yayım Bildirim Sisteminde eğer bir kitap yayımlanması (yaklaşık 2000 Kb) durumunda bu agventin sistem üzerindeki dağıtım performansı Şekil 9.12'de gösterilmektedir.



Şekil 9.12 : Bir kitap verisi içeren agventin dağıtım süresi

10. SONUÇ VE ÖNERİLER

Geleneksel istemci-sunucu mimarisinin kullanıldığı küçük ölçekli dağıtılmış sistemler uçtan uca ve senkron haberleşmeleri sebebi ile statik uygulamaların geliştirilmesinin ötesine geçmemiştir. Bu haberleşmenin büyük ölçekli sistemlerde kullanılması ise uygulamaların hantal ve kullanışsız olmalarını sağlamıştır. Bu kısıtlamalar, dinamik olan ve bağlantısız uygulamaların çalıştığı bir ortamda, daha esnek haberleşme modellerini/sistemlerini kullanan yazılım sistemlerinin geliştirilmesi gerekliliğini ortaya çıkarmıştır.

Günümüze kadar geliştirilen tüm olay sistemleri olay kavramını, veri kaynağı / veri üreticisi tarafından üretilen basit formatta bir veri olarak algılamışlar ve çalışmalarını üretilen bu verinin, uygun formatta gösterilimi, işlenmesi ve hızlı şekilde yayılması üzerine yoğunlaştırmışlardır. Tez çalışmasında olay kavramının üretim niteliğini geliştirilerek, alışılagelen sadece ham veri üretiminin yanı sıra, eylemlerin de üretildiği ileri bir düzeye yerleştirilmektedir. Çalışma bünyesinde bu nitelikteki olayların üretilmesine, dağıtılmasına ve işlenmesine olanak sağlayan etmen tabanlı bir dağıtılmış olay sistemi altyapısı geliştirilmiştir.

Bu düzeydeki bir olay, işlenebilir veriler üretebileceği gibi, kendine özgü bir kontrol akışına sahip olan bir eylemler dizisinin tetikleyicisi de olabilir. Söz konusu eylemler belirli bir amacı gerçeklemeye yönelik işlem dizisine, karar mekanizmalarına ve çoğunlukla birlikte üretildikleri bir veri kümesine sahip olacaklardır.

Tez çalışmasında, bir olayın tetikleyeceği eylem dizisinin ve bu eylemleri yerine getirmede kullanacağı veri bloğunun yer aldığı bir “gezgin etmen” yapısıyla temsil edilmesi öngörülmüştür. Tez çalışmasında üretilen bir olayın yarattığı eylem dizisini temsil eden yapıya “agvent” (**ag**ent e**vent**) adı verilmiş olup sistem bir bütün olarak “Etmen Tabanlı Dağıtılmış Olay Sistemi-Agent Based Distributed Event System” veya kısaca İngilizce kısaltması kullanılarak “ABDES Sistemi” olarak adlandırılmıştır.

Geliştirilen sistem, özellikle içerdiği bilginin önemli olduğu ve bu bilginin diğer sistem varlıklarından gizlenmesinin gerektiği dağıtılmış olay sistemini kullanan farklı uygulama alanları için uygun çalışma ortamı sağlamaktadır.

10.1. ABDES Sisteminin Değerlendirmesi

ABDES Sistemi, hem gezgin-zeki etmen yapısının *amaca yönelik davranan, özerk ve taşınabilirlik* özelliklerinden, hem de kayıt/yayım modeliyle haberleşen olay tabanlı sistemlerin temel özellikleri olan *Bir uçtan çok uca haberleşme asenkron haberleşme ve gevşek bağlı sistemlerde kullanıma uygunluk* özellikleri birleştiren bir Etmen Tabanlı Dağıtılmış Olay Sistemidir. ABDES Sisteminin, daha önceleri geliştirilen dağıtılmış olay sistemlerinden üstün olan temel özelliklerini kısaca şu ana başlıklar altında toplanabilir.

- *Özerklik:* Gezgin etmen olarak üretilen agventlere üretim aşamasında amaçları, yüklenir ve tüketicilere ulaştırılmasında asıl kontrol agventin kendisinde olur.
- *Tip/Agvent Tabanlı Kayıt:* Agventin tipi üzerinden yapılabilen kayıt yapısı sayesinde sadece agventin özellikleri üzerinden değil davranışları üzerinden de kayıt olunabilmekte ve filtreleme yapılabilmektedir
- *Bilgi Saklama:* Agvent ile beraber dağıtılan bilgi gizlenmekte ve gerek Agvent Sunuculara gerekse Kayıtçılara bu bilgi gösterilmemektedir.
- *Kullanıcı Tanımlamalı Agvent Tipleri:* Kullanıcı tarafından farklı agvent tiplerinin üretilmesine ve bunları sistem üzerinde yayımlanmalarına olanak sağlanmaktadır.
- *Kendi Kendini Yönlendiren Agventler***:** Agvent Sunucu üzerinde aktif hale gelen agvent onun bilgi tabanından faydalanarak kendisini yönlendireceği hedef makineleri belirlemektedir.
- *Çift Taraflı Filtreleme:* Filtreleme sadece Kayıtçılar tarafından değil, aynı zamanda Yayımcılar tarafından da yapılabilmektedir.

ABDES Sistemi, yukarıda sıralanan özellikleri açısından yeni bir yaklaşım getirmiş olup diğer dağıtılmış olay sistemlerinin açık kalan bazı noktalarını kapatmıştır. ABDES Sisteminin dördüncü bölümde bahsedilmiş olan diğer dağıtılmış olay sistemleri ile karşılaştırılması Tablo 10.1'de gösterilmektedir.

Tablo 10.1. ABDES Sisteminin Diğer Olay Sistemleri ile Karşılaştırılması

Olay Sistemi	Altyapı	Kayıt Modeli	Yönlendirme Mekanizması	Filtreleme	Bilgi Saklama	Kullanıcı Tanımlamalı Olay Tipleri	Özerklik
Sift	Merkezi	İçerik Tabanlı	Olay Sunucu	Tüketici	Yok	Yok	Yok
Elvin	Merkezi	İçerik Tabanlı	Olay Sunucu	Tüketici	Yok	Yok	Yok
Yeast /Ready /Omninotify	Merkezi /Dağıtılmış /Dağıtılmış	İçerik Tabanlı	Olay Sunucu	Tüketici	Yok	Yok	Yok
Java AWT	Merkezi	Kanal Tabanlı	Olay Sunucu	Tüketici	Yok	Yok	Yok
Corba Olay Sistemi	Merkezi	Konu Tabanlı	Olay Sunucu	Tüketici	Yok	Yok	Yok
Rebeca	Dağıtılmış	İçerik Tabanlı	Olay Sunucu	Tüketici	Yok	Yok	Yok
LeSubscribe	Dağıtılmış	İçerik Tabanlı	Olay Sunucu	Tüketici	Yok	Yok	Yok
Jedi	Dağıtılmış[38]	Konu Tabanlı	Olay Sunucu	Tüketici	Yok	Yok	Yok
Siena	Dağıtılmış	İçerik Tabanlı	Olay Sunucu	Tüketici	Yok	Yok	Yok
Gryphon	Dağıtılmış	İçerik Tabanlı	Olay Sunucu	Tüketici	Yok	Yok	Yok
Corba Bildirim Sistemi	Merkezi	Konu Tabanlı + İçerik Tabanlı	Olay Sunucu	Tüketici	Yok	Yok	Yok
Jecho	Dağıtılmış	Konu Tabanlı +Tip Tabanlı	Olay Sunucu	Tüketici	Yok	Yok	Yok
Steam	Dağıtılmış	Konu Tabanlı	Olay Sunucu	Tüketici	Yok	Yok	Yok
Fuego	Dağıtılmış	İçerik Tabanlı	Olay Sunucu	Tüketici	Yok	Yok	Yok
Dadi	Dağıtılmış	İçerik Tabanlı	Olay Sunucu	Tüketici	Yok	Yok	Yok
Hermes	Dağıtılmış	Tip Tabanlı	Olay Sunucu	Tüketici	Yok	Var	Yok
Obvent	Dağıtılmış	Tip Tabanlı	Olay Sunucu	Tüketici	Var[39]	Var[40]	Yok
ABDES Sistemi	**Dağıtılmış**	**Tip Tabanlı (Agvent Tabanlı)**	**Agvent, Agvent Sunucu[41]**	**Üretici ve Tüketici**	**Var**	**Var**	**Var**

[38] Hiyerarşik olarak dağıtılmış, döngüsel olmayan yapıyı desteklemektedir.
[39] Nesneye yönelik yaklaşımın bir özelliği olan Kapsama(Encapsulation) özelliğine sahip olup, bilgiye erişim işlemleri alıcı ve atayıcı metotlar üzerinden yapılmaktadır.
[40] Tanımlanan her yeni obvent tipi için bu obventin haberleşme sistemi ile bağlantısını sağlayacak bir adaptörün, kendi özel derleyicisi olan PSC derleyicisi ile üretilmesi gerekmektedir.
[41] Agventler, uç konaklarda; Agvent Sunucular ise ara konaklarda yönlendirme işlemlerini yapmaktadırlar. Hedef makinelerin seçimi agventler tarafından yapılmaktadır.

Bu temel özelliklerin yanı sıra geliştirilen ABDES Sistemi

- Ölçeklenebilir yapısı sayesinde Agvent Sunucuların sayısı artırılabilmekte ve daha fazla tüketiciye ulaşılabilmektedir.
- Kayıt ölçütlerini kolay ve anlamlı olarak belirlemeye olanak sağlayan kayıt modeli sayesinde tüketicilerin kendi ölçütlerini kolaylıkla tanımlamasına olanak sağlanmaktadır.
- Kullanılan Duyuru ve Kayıt mesajları sayesinde sistem üzerinde dolaşan mesajların sayısı en aza indirilmektedir.
- Hata Kotarma mekanizması sayesinde Duyuru ve Kayıt Mesajlarının dağıtımı veya Agventlerin yönlendirmeleri sırasında bir Agvent Sunucunun çökmesi halinde alternatif yollar üzerinden hedef makineye ulaşım olanağı sağlanmaktadır.
- Bilgi Tabanı Yönetiminde Nesne Veri Tabanları kullanılması ile Duyuru ve Kayıt mesajlarının Agvent Sunucular üzerinde birer nesne olarak saklanmasına ve bu nesnelere hızlı şekilde erişilmesine olanak sağlanmaktadır.

ABDES Sistemi performansı açısından değerlendirildiğinde, önceden geliştirilen dağıtılmış olay sistemlerinde olay verileri basit formatta tanımlandıkları için sistem üzerinde hızlı şekilde dağıtılmalarına olanak sağlandığı ve bunun performansı daha da artırdığı görülmektedir. ABDES Sistemi, geliştirmesi amaçlanan ve birinci bölümde bahsedilen uygulama alanları açısından basit olay verileri yerine nitelikli olay verileri kullanmaktadır. Bilgi saklamanın ön planda olduğu bu tip uygulamalarda Yayımcı için gizlenmesi gereken ve büyük boyutlu olan veriler agventle beraber yayımlanmaktadır. Bu veriler uygulama örneğine bağlı olarak üretilen agventin boyutunu büyütmektedir. Agventin boyutunun artması ise üretilen agventin tüketicilere ulaştırılma süresini kaçınılmaz şekilde artırmaktadır. Bilgi saklamanın ön planda olduğu ABDES Sisteminin kullanım alanları açısından incelendiğinde Bölüm 9.2.3‘te hesaplanan dağıtım sürelerinin kabul edilebilir boyutlarda olduğu değerlendirilmektedir.

10.2. Geliştirme Önerileri

ABDES Sistemi, hem etmen yapısının özelliklerini, hem de kayıt/yayım modeliyle haberleşen olay tabanlı sistemlerin özelliklerini birleştiren bir Etmen Tabanlı Dağıtılmış Olay Sistemidir. Sistem nitelikli olay verilerinin üretilmesini ve dağıtılmasını amaçlamış olup, bu olay verileri (agventler) sistem üzerinde kendi özerkliği ile dolaşmalarına olanak sağlayan gezgin etmen yapısında geliştirilmiştir.

ABDES Sistemi, Dağıtılmış Olay Tabanlı Sistemlerin geliştirilmesinde yeni bir yaklaşım getirmiş olup, açık kalan değişik problemleri çözmüştür. Sistemin daha da geliştirilebileceği alanlar ve bunlar hakkında çözüm önerileri şu şekilde sıralanmıştır.

- *Sistem Bileşenlerine Hareketsellik Kazandırılması*: ABDES Sistemi, olay mesajları olarak kullandığı agventlere, kullanmış olduğu gezgin etmen yapısı sayesinde bir hareketsellik kazandırmıştır. Sistemin diğer bileşenleri olan gerek Yayımcıların gerekse Kayıtçıların farklı Agvent Sunucular üzerinden sisteme bağlanmalarına olanak sağlanması ABDES Sistemine daha da esneklik kazandıracaktır. Özellikle günümüzde kablosuz ağların hızla yaygınlaştığı görülmektedir. Dizüstü bilgisayarı ile dolaşan bir olay üreticinin, bir gün İstanbul'dan, bir gün Ankara'dan sisteme bağlanacağı gibi, benzer bir olay tüketicisinin de farklı ülkelerden sisteme dahil olması gerekebilir. Bu durumdaki sistem varlığının daha önceki sistem üzerindeki bilgilerinin yeni agvent sunucusuna taşınması ve gerekli yönlendirme ayarlamalarının yapılması gerekmektedir.
- *Nesne Karşılaştırmalarına Olanak Sağlanması:* ABDES Sisteminde değişken tipleri olarak desteklenen sadece basit veri tipleridir. Sistemin esnekliği açısından nesnelerinde karşılaştırılmalarına olanak sağlayacak bir yapının geliştirilmesi sistemin kullanılabileceği uygulama alanlarının artmasını sağlayacaktır. Bunu sağlamak içinde karşılaştırma işlemlerinin yapılabileceği sınıf yapılarının gerçeklenmesi gerekmektedir. Karşılaştırılacak nesneler *ComparableObject* şeklinde bir üst sınıftan (veya arayüzden (interface)) türetilerek, karşılaştırma işlemlerinin bu üst sınıfın tanımladığı metotlarla halledilmesi sağlanabilir.
- *Bileşik Olayların Desteklenmesi:* Bileşik olayların birden fazla olay verisi üzerinden bir Kayıtçının sisteme kayıt olmasını sağlayan bir kayıt mekanizmasıdır. Örneğin bir Kayıtçı bir odadaki ışık miktarı artıyorsa ve ısı değeri yükseliyorsa bu bir yangın başlangıcı olacağı için böyle bir bileşik olaydan haberdar edilmesini ve itfaiyenin aranmasını isteyebilir. Bileşik olayların kullanılması için temel problem dağıtılmış sistemlerde global bir zaman ayarlamasının yapılmasının zorluğudur.
- Çoklu yayım (multicast) üzerine yapılan çalışmalardan da bilinen *güvenirlik (reliability)* [95, 96] ve *sıralama* [97] gibi servis kalitelerinin geliştirilmesi ile sistemin işlevleri ve uygulama alanları geliştirilebilir.
- Büyük ölçekli sistemlerde *güvenlik* daima üzerinden gelinmesi zor görevlerden olmuştur. Özellikle kayıt yayım haberleşme modelini kullanan dağıtılmış olay sistemlerinde dolaylı haberleşmeden ötürü bu problemin

üzerinden gelmek biraz daha karmaşık hal almıştır. Günümüzde bu alanda yapılan az sayıda çalışma bulunmakta olup [98, 99] gelecek çalışmalarda üzerinde durulması gereken bir konudur.

KAYNAKLAR

[1] **Chambers, D., Lyons, G. and Duggan J.,** 2000. Design of Virtual Store using Distributed Object Technology. *In Proceedings of the 5th International Symposium on Software Engineering for Parallel and Distributed Systems,* Limerick, Ireland, 66-75.

[2] **Addlesee, M., Curwen, R., Hodges, S. Newman J. and Steggles, P**., 2001. Implementing a Sentient Computing System, *IEEE Computer,* **34,** 50-56.

[3] **Muller H. and Randell, C.,** 2000. An Event-Driven Sensor Architecture for Low Power Wearables, *In Proceedings of the Workshop on Software Engineering for Wearable and Pervasive Computing,* Limerick, Ireland, 39-41.

[4] **Weiser, M.,** 1993. Ubiquitous Computing, *IEEE Hot Topics,* **26,** 71-72.

[5] **Dearle, A.,** 1998. Towards Ubiquitous Environments for Mobile Users, *IEEE Internet Computing,* **2,** 22-32.

[6] **Sun Microsystems Inc.,** 1997. Java AWT: Delegation Event Model: Sun Microsystems Inc.

[7] **Microsoft Corporation,** 2005. Visual C# Developer Center: The C# Language Specification, http://msdn.microsoft.com/vcsharp/programming/language/.

[8] **Maffeis, S.,** 1999. Developing Publish/Subscribe Applications with iBus, SoftWired AG, *Technical White Paper,* http://www.softwired-inc.com/ibus.

[9] **Kang, S.J., Park, S.H., and Park, J.H.,** 2001. ROOM-BRIDGE: A Vertically Configurable Network Architecture and Real-Time Middleware for Interoperability between Ubiquitous Consumer Devices in Home, *Springer-Verlag,* **2218,** 232-251.

[10] **Wooldridge M. and Jennings, N.R.,** 1995. Intelligent Agents: Theory and Practice, *The Knowledge Engineering Review,* **10(2),** 115–152.

[11] **Distributed.net,** 2005. Project RC5, http://www.distributed.net/rc5/.

[12] **Maffeis, S.,** 2000. Client/Server Term Definition, Eds. Ralston, A., Hemmendinger, D. and Reilly, E., International Thomson Computer Publishing, Zurich.

[13] **Reiss, S.,** 1990. Connecting Tools Using Message Passing in the Field Environment, *IEEE Software,* **7(4),** 57–66.

[14] **Julienne, A.M. and Holtz, B.,** 1994. ToolTalk and Open Protocols, Inter-Application Communication, Prentice–Hall, New Jersey.

[15] **Krishnamurthy, B. and Rosenblum, D.S.,** 1995. Yeast: A General Purpose Event-Action System, *IEEE Transactions on Software Engineering,* **21(10),** 845–857.

[16] **Salton, G.,** 1968. Automatic Information Organization and Retrieval, McGraw-Hill, New York.

[17] **Belki, N. and Croft, W. B.,** 1992. Information Retrieval and Filtering: Two Sides of The Same Coin, *Communications of the ACM,* **35(12),** 29–38.

[18] **Aguilera, M., Strom, R., Sturman, D., Astley, M. and Chandra, T.,** 1999. Matching Events in a Content-Based Subscription System, *In Proceedings of the ACM PODC Symposium on Principles of Distributed Computing,* Atlanta, USA, 53-61.

[19] **Carzaniga, A.,** 1998. Architectures for an Event Notification Service Scalable to Wide-area Networks, *PHD Thesis,* Politecnico di Milano, Italy.

[20] **Fabret, F., Lirbat, F., Pereira, J. and Shasha, D.,** 2000. Efficient Matching for Content-Based Publish/Subscribe Systems, *Technical Report,* Institut National de Recherche en Informatique et en Automatique (INRIA), France.

[21] **Yan, T. W. and Garcia-Molina, H.,** 1995. SIFT - A Tool for Wide-Area Information Dissemination, *In Proceedings of the USENIX Technical*

Conference on UNIX and Advanced Computing Systems, New Orleans, USA, 177-186.

[22] **Carzaniga, A., Rosenblum, D. and Wolf, A.,** 2001. Design and Evaluation of a Wide-area Event Notification Service, *ACM Transactions on Computer Systems (TOCS),* **19(3),** 332–383.

[23] **Liu, L., Pu, C. and Tang, W.,** 1999. Continual Queries for Internet Scale Event-Driven Information Delivery, *IEEE Transactions on Knowledge and Data Engineering ,* **11(4),** 610–628.

[24] **Strom, R., Banavar, G. and Chandra, T.,** 1998. Gryphon: An Information Flow based Approach to Message Brokering, *In Proceedings of the ISSRE International Symposium on Software Reliability Engineering,* Paderborn, Germany, 10-21.

[25] **Fabret, F., Jacobsen, H., Llirbat, F., Pereira, J., Ross, K., and Shasha, D.,** 2001. Filtering Algorithms and Implementation for Very Fast Publish/Subscribe, *In Proceedings of the ACM SIGMOD International Conference on Management of Data,* Santa Barbara, 115-126.

[26] **BEA MessageQ Release 5.0 Documentation**, 2000. http://edocs.bea.com/tuxedo/msgq/.

[27] **IBM Corporation,** 1995. MQseries: An Introduction to Messaging and Queuing, *Technical Report,* **GC33-0805-01**, IBM Corporation, USA.

[28] **OMG - Object Management Group,** 1999. CORBA Notification Service Specification, *Technical Report,* **telecom/98-06-15**, Object Management Group (OMG), Massachusetts.

[29] **Sharma, R., Hapner, M., Burridge, R.,** 2002, Java Message Service API Tutorial and Reference: Messaging for the J2EE(tm) Platform, Addison-Wesley, Boston.

[30] **Tanenbaum A. and Goodman, J.,** 1999. Structured Computer Organization Prentice Hall Inc., Englewood Cliffs, New Jersey.

[31] **Bernstein, P. and Newcomer, E.,** 1997. Principles of Transaction Processing, Morgan Kaufmann Publishers Inc, California.

[32] **Walrath, K., Campione, M., Huml, A. and Zakhour, S.,** 2004. JFC Swing Tutorial, The: A Guide to Constructing GUIs, 2nd Edition, Addison Wesley, Boston.

[33] **Object Technology Internationl Inc,** 2001. Eclipse Platform–Technical Overview (White Paper), *Technical Report*, http://www.eclipse.org/whitepapers/eclipse-overview.pdf.

[34] **Maffeis, S.,** 1997. iBus: The Java Intranet Software Bus, *Technical Report*, Soft-Wired AG, Zurich, Switzerland.

[35] **Sun Microsystems Inc.,** 1997. Java AWT: Delegation Event Model, http://java.sun.com/j2se/1.3/docs/guide/awt/designspec/events.html.

[36] **Object Management Group,** 1995. CORBA Services: Common Object Services Specification, Object Management Group, http://www.omg.org/library/csindx.html.

[37] **Carzaniga, A., Rosenblum, D.S. and Wolf, A.L.,** 2000. Achieving Scalability and Expressiveness in an Internet-Scale Even Notification Service, *In Proceedings of the 19th ACM Symposium on Principles of Distributed Computing*, Oregon, USA, 219 –227.

[38] **Cugola, G., Di Nitto, E. and Fuggetta, A.,** 1998. Exploiting An Event-Based Infrastructure to Develop Complex Distributed Systems, *In Proceedings of the 20th International Conference on Software Engineering*, Kyoto, Japan, 261- 270.

[39] **Cugola, G., Nitto, E. and Fuggetta, A.,** 2001. The JEDI Event-Based Infrastructure and Its Application to the Development of the OPSS WFMS, *IEEE Transaction of Software Engineering*, **27(9),** 827-850.

[40] **SwiftMQ,** 2005. JMS Enterprise Messaging Platform, http://www.swiftmq.com/.

[41] **Sun Microsystems Inc.,** 2005. Java Message Service, http://www.java.sun.com/products/jms/docs.html.

[42] **Carzaniga, A., Rosenblum, D. S., and Wolf, A.L.,** 1998. Design of a Scalable Event Notification Service: Interface and Architecture, *Technical Report*, Department of Computer Science, University of Colorado.

[43] **Segall, B. and Arnold, D.,** 1997. Elvin Has Left The Building: A Publish/Subscribe Notification Service with Quenching, *In Proceedings of the AUUG Australian UNIX and Open Systems User Group Conference,* Queensland, Australia, 243-255.

[44] **Muhl, G., Fiege, L. and Buchmann, A.P.,** 2002. Filter Similarities in Content-Based Publish/Subscribe Systems, *In Proceedings of the ARCS International Conference on Architecture of Computing Systems,* Karlsruhe, Germany, 244-240.

[45] **Eugster, P.T., Guerraoui, R. and Damm, C.H.,** 2001. On Objects and Events, *In Proceedings of ACM Object Oriented Programming Systems Languages and Applications,* Florida, USA, 131–146.

[46] **Sahingoz, O.K. and Erdogan, N.,** 2003. A Two-Leveled Mobile Agent System for Electronic Commerce, *the Journal of Aeronautics and Space Technologies Institute (ASTIN),* **1(2),** 21-32.

[47] **Russell, S. and Norvig, P.,** 1995. Artificial Intelligence: A Modern Approach, Prentice-Hall, New Jersey.

[48] **Odell, J.,** 2002. Objects and Agents Compared, *In Journal of Object Technology,* **1(1),** 41-53.

[49] **Booch, G.,** 1994. Object-Oriented Analysis and Design, Addison-Wesley, Massachusetts.

[50] **Stamos, J.W. and Gifford, D.K.,** 1990. Remote Evaluation, *ACM Transactions on Programming Languages and Systems,* **12(4),** 537-565.

[51] **Baumann, J., Hohl, F., Rothermel, K., and Straber, M.,** 1998. Mole - Concepts of a Mobile Agent System, *The World Wide Web Journal,* **1(3),** 123-137.

[52] **Lange, D.B. and Oshima, M.,** 1999. Seven Good Reasons for Mobile Agents, *Communications of the ACM,* **42(3),** 88-89.

[53] **Brandt, S. and Kristensen, A.,** 1997. Keryx: Internet Notification Service for Dynamic Web Applications, In Proceedings of W3C Workshop on Push Technology, Boston, USA.

[54] **Faensen, D, Faulstich, L.C., Schweppe, H., Hinze, A. and Steidinger, A.,** 2001. Hermes – A Notification Service for Digital Libraries, *In Proceedings of the ACM/IEEE JCDL Joint Conference on Digital Libaries*, Virginia, USA, 373-380.

[55] **Liebig, C., Cilia, M. and Buchmann., A. P.,** 1999. Event Composition in Time-Dependent Distributed Systems, *In Proceedings of the IFCIS CoopIS International Conference on Cooperative Information Systems,* Edinburgh, Scotland, 70-78.

[56] **Shekhar, S. and Fetterer, A.,** 1996. Genesis: An Approach to Data Dissemination in Advanced Traveler Information Systems, *IEEE Bulletin of the Technical Committee on Data Engineering,* **19(3),** 40–47.

[57] **Lexico Publishing Group (LLC),** 2003. Reference.com Information Service, Homepage, http://www.reference.com.

[58] **Arnold, D., Segall, B., Boot, J., Bond, A., Lloyd, M., and Kaplan, S.,** 1999. Discourse With Disposable Computers: How and Why You Will Talk to Your Tomatoes, *In Proceedings of the Usenix Workshop on ES99 Embedded Systems*, Cambridge, Massachusetts, 9-22.

[59] **Gruber, R., Krishnamurthy, B. and Panagos, E.,** 1999. The Architecture of the READY Event Notification Service, *In Proceedings of the IEEE ICDC International Conference on Distributed Computing Systems Middleware Workshop,* Austin, USA, 108-113.

[60] **OmniNotify: Project Group,** 2004. http://omninotify.sourceforge.net/.

[61] **Sun Microsystems Inc.,** 2004 Java Distributed Event Specification, Sun Microsystems Inc., http://java.sun.com/developer/products/jini/specs.

[62] **Object Management Group,** 1996. CORBAservices: Notification Service Specification-Request for Proposal, http://www.omg.org/techprocess/meetings/schedule/Notification_Service_RFP.html.

[63] **Ma, C. and Bacon. J.,** 1998. COBEA: A CORBA-based event architecture, *In Proceedings of the COOTS Conference on Object-Oriented Technologies and Systems,* Berkeley, California, 117-132.

[64] **Pereira, J., Fabret, F., Llirbat, F. and Shasha, D.,** 2000. Efficient Matching For Web-Based Publish/Subscribe Systems, *In Proceedings of the IFCIS CoopIS International Conference on Cooperative Information Systems,* Eilat, Israel, 162-173.

[65] **Fabret, F., Jacobsen, H., Llirbat, F., Pereira, j. and Shasha., D.,** 2001. Webfilter: A High-Throughput XML-Based Publish And Subscribe System, *In Proceedings of the VLDB International Conference on Very Large Databases*, Rome, Italy, 723-724.

[66] **Eisenhauer, G., Bustamante, F. and Schwan, K.,** 2001. A middleware toolkit for client-initiated service specialization, *ACM SIGOPS Operating Systems Review*, **35(2),** 7–20.

[67] **Chen, Y., Schwan, K. and Zhou, D.,** 2003. Opportunistic Channels: Mobilityaware Event Delivery, *In ACM/IFIP/USENIX International Middleware Conference*, Rio de Janeiro, Brazil, 182–201.

[68] **Meier, R. and Cahill, V.,** 2003. Exploiting Proximity in Event-based Middleware for Collaborative Mobile Applications, *In 4th International Conference on Distributed Applications and Interoperable Systems*, Rhode Island, USA, 285–296.

[69] **Chen, M., LaPaugh, A. and Singh, J.P.,** 2003. Content Distribution for Publish/Subscribe Services, *In Proceedings of ACM/IFIP/USENIX International Middleware Conference*, Rio de Janeiro, Brazil, 83-102.

[70] **Cao, F. and Singh, J.P**., 2004. Efficient Event Routing in Content-based Publish-Subscribe Service Networks, *In Proceedings of IEEE INFOCOM,* Hong Kong, 929-940.

[71] **Cao, F. and Singh, J.P.,** 2005. MEDYM: Match-Early and Dynamic Multicast for Content-based Publish-Subscribe Service Networks, *In the 6th ACM/IFIP/USENIX International Middleware Conference,* Grenoble, France, 292–313.

[72] **Pietzuch, P. and Bacon, J.,** 2002. Hermes: A Distributed Event-based Middleware Architecture, *In Proceedings of the 1st International Workshop on Distributed Event-Based Systems,* Vienna, Austria, 611-618.

[73] **Pietzuch, P.R.,** 2004. Hermes: A Scalable Event-Based Middleware, *PhD Thesis,* Computer Laboratory, Queens' College, University of Cambridge.

[74] **Tarkoma, S.,** 2005. Efficient and Mobility-aware Content-based Routing Systems, *PhD Thesis,* University of Helsinki, Department of Computer Science.

[75] **Komu, M., Tarkoma, S., Kangasharju, J. and Gurtov, A.,** 2005. Applying a Cryptographic Namespace to Applications. Dynamic Interconnection of Networks, *Mobicom 2005 ACM Workshop,* Cologne, Germany, 23-27.

[76] **Şahingöz Ö.K. and Erdoğan N**., 2003. RUBCES: Rule Based Composite Event System, *In Proceedings of the International XII. Turkish Symposium of Artificial Intelligence and Neural Networks (in CD, no page numbers),* Canakkale.

[77] **Şahingöz Ö.K. and Erdoğan N.,** 2003. RUBDES: Rule Based Distributed Event System, *Springer-Verlag,* **2869,** 284-291.

[78] **Şahingöz Ö.K. and Erdoğan N.,** 2004. An Agent Based Distributed Event System Framework, *3rd Asian Pacific International Symposium on Information Technology,* Istanbul, 548-554.

[79] **Şahingöz Ö.K. and Erdoğan N.,** 2004. Dispatching Mechanism of an Agent-Based Distributed Event System, *Springer-Verlag*, **3036,** 184-191

[80] **Picco, G.P., Cugola, G. and Murphy, A.L.,** 2003. Efficient Content-Based Event Dispatching in the Presence of Topological Reconfiguration, *23rd IEEE International Conference on Distributed Computing Systems,* USA, 234–243.

[81] **Baldoni, R., Beraldi, R., Querzoni, L. and Virgillito, A.,** 2004. A Self-Organizing Crash-Resilient Topology Management System for Content-Based Publish/Subscribe, *International Workshop on Distributed Event-Based Systems*, Edinburgh, Scotland, 3-8.

[82] **Xu, Z. and Srimani, P.K.,** 2005. Self-Stabilizing Publish/Subscribe Protocol for P2P Networks, *Springer-Verlag*, **3741,** 129-140.

[83] **Buchmann, A., Bornhövd, C., Cilia, M., Fiege, L., Gartner, F., Liebig, C., Meixner, M., and Mühl, G.,** 2004. DREAM: Distributed Reliable Event-Based Application Management, *Web Dynamics: Adapting to Change in Content, Size, Topology and Use*, 319-352. Eds. Levene, M. and Poulovassilis, A., Springer Verlag, Heidelberg.

[84] **Cugola, G., Frey, D., Murphy, A. L. and Picco, G.P.,** 2004. Minimizing the Reconfiguration Overhead in Content-Based Publish-Subscribe, *In Proceedings of the 19th ACM Symposium on Applied Computing (SAC)*, Cyrprus, 1134-1140.

[85] **Oh., S., Pallickara, S.L., Ko, S., Kim, J. and Fox, G.,** 2005. Publish/Subscribe Systems on Node and Link Error Prone Mobile Environments, *Springer-Verlag,* **3515,** 576–584.

[86] **Şahingöz Ö.K. and Sönmez, A.C.,** 2006. Fault Tolerance Mechanism of Agent-Based Distributed Event System, *Springer-Verlag,* **3993,** 903–907.

[87] **Şahingöz Ö.K. and Sönmez, A.C.,** 2006. Mobile Agent Based Publication Alerting System, *Springer-Verlag,* **3993,** 192 – 199.

[88] **Springer Link Alert,** 2006. Service Homepage at www.springerlink.com/ alerting.

[89] **Elsevier Contents Direct,** 2006. Service Homepage at http://contentsdirect. elsevier.com/.

[90] **ACM TOC Alert,** 2006. Access to personalized services via http://portal.acm.org/.

[91] **Medina, A., Lakhina, A., Matta,I. and Byers, J.,** 2001. BRITE: An Approach to Universal Topology Generation, *In Proceedings of the International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunications Systems*, Ohio, USA, 346-356.

[92] **BRITE,** 2001. Boston university Representative Internet Topology gEnerator. http://www.cs.bu.edu/brite/.

[93] **Huffaker, B., Nemeth, E. and Claffy, K.,** 1999. Otter: A General-purpose Network Visualization Tool, International Networking Conference (INET) , http://www.caida.org/Tools/Otter/Paper.

[94] **Otter,** 1999. Tool for Topology Display. http://www.caida.org/tools/visualization/otter/.

[95] **Bhola, S., Strom, R., Bagchi, S., Zhao, Y. and Auerbach, J.,** 2002. Exactly-once Delivery in a Content-based Publish-subscribe System, *In Proceedings of the International Conference on Dependable Systems and Networks*, Bethesda, USA, 7-16.

[96] **Huang, Y. and Hector, G.,** 2001. Exactly-once Semantics in a Replicated Messaging System, *In Proceedings of the 17th International Conference on Data Engeneering*, Heidelberg, Germany, 3-12.

[97] **Liebig, C., Cilia, M. and Buchmann, A.,** 1999. Event Composition in Time-dependent Distributed Systems, *In Proceedings of the 4th Intl. Conference on Cooperative Information Systems*, Edinburgh, Scotland, 70-78.

[98] **Opyrchal, L. and Prakash, A.,** 2001. Secure Distribution of Events in Content-Based Publish-subscribe Systems, *In Proceedings of 10th USENIX Security Symposium*, Washington, DC, USA, 281-295.

[99] **Wang, C., Carzaniga, A., Evans, D., and Wolf, A.,** 2002. Security Issues and Requirements for Internet-scale Publish-subscribe Systems, *In Proceedings of the Thirty fifth Hawaii International Conference on System Sciences*, Big Island, Hawaii, 9-16.

ÖZGEÇMİŞ

24 Şubat 1973 yılında Kayseri'de doğan Özgür Koray ŞAHİNGÖZ, 1988 yılında Kayseri Lisesinden mezun olmuştur. Boğaziçi Üniversitesi Bilgisayar Mühendisliği bölümünden 1993 yılında mezun olmuştur. Yüksek Lisans Eğitimini İstanbul Teknik Üniversitesinde Kontrol ve Bilgisayar Mühendisliği Anabilim Dalında 1998 yılında tamamlamıştır. 1998 yılında aynı programda doktora öğretimine başlamıştır. 1994 yılından bu yana Hava Harp Okulu Dekanlığı Bilgisayar Mühendisliği Bölümünde öğretim görevlisi olarak çalışan Özgür Koray ŞAHİNGÖZ evli ve iki çocuk babasıdır.