İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

ETMEN TABANLI BİR GRİD SİSTEMİ

YÜKSEK LİSANS TEZİ

Uygar GÜMÜŞ

Anabilim Dalı : Bilgisayar Mühendisliği

Programı : Bilgisayar Mühendisliği

HAZİRAN 2009

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

ETMEN TABANLI BİR GRİD SİSTEMİ

YÜKSEK LİSANS TEZİ

Uygar GÜMÜŞ
(504061534)

Tezin Enstitüye Verildiği Tarih : 04 Mayıs 2009
Tezin Savunulduğu Tarih : 03 Haziran 2009

Tez Danışmanı : Prof. Dr. Nadia ERDOĞAN (İTÜ)
Diğer Jüri Üyeleri : Yrd. Doç. Dr. D. Turgay ALTILAR (İTÜ)
Yrd. Doç. Dr. Yunus Emre SELÇUK (YTÜ)

HAZİRAN 2009

Eşime ve aileme,

ÖNSÖZ

Tez çalışmam süresince desteğini ve ilgisini hiç eksik etmeyen, anlayışlı tavrıyla zevkli bir çalışma ortamı sağlayan değerli hocam Sayın Prof. Dr. Nadia Erdoğan'a teşekkürlerimi sunarım.

Yüksek lisans eğitimini ve tez çalışmasını destekleyen TÜBİTAK Bilim İnsanı Destekleme Başkanlığı (BİDEB)'e teşekkürlerimi sunmayı borç bilirim.

Aileme, hayatımın her alanında olduğu gibi eğitimim süresinde gösterdikleri her türlü destek için ayrıca minnettar olduğumu belirtmeliyim.

Özellikle yaşantıma anlam katan biricik eşim Sezgi Yılmaz Gümüş'e bu çalışmam sürecinde de gösterdiği sınırsız sabır ve destek için ayrıca şükranlarımı sunarım.

Haziran 2009 **Uygar Gümüş**

İÇİNDEKİLER

Sayfa

KISALTMALAR

3APL : Artificial Autonomous Agents Programming Language
ACC : Agent Communication Channel
ACL : Agent Communication Language
AID : Agent Identifier
AMS : Agent Management System
API : Application Programming Interface
ARPA : Advanced Research Projects Agency
BDI : Belief Desire Intention
CORBA : Common Object Request Broker Architecture
CPU : Central Process Unit
DCE : Distributed Computing Environment
FIPA : Foundation for Intelligent Physical Agents
FTP : File Transfer Protocol
GASS : Global Access to Secondary Storage
GRAM : The Grid Resource Allocation and Management
GSI : Grid Security Infrastructure
GUI : Graphical User Interface
HTTP : Hyper Text Transfer Protocol
IP : Internet Protocol
JADE : Java Agent Development Framework
KQML : Knowledge Query Manipulation Language
MDS : Monitoring and Discovery Service
MPI : Message Passing Interface
OGSA : Open Grid Services Architecture
PRS : The Procedural Reasoning System
DF : Directory Facilitator
PVM : Parallel Virtual Machine
RMA : Remote Management Agent
RMI : Remote Method Invocation
RPC : Remote Procedure Call
SOAP : The Simple Object Access Protocol
TCP : Transmission Control Protocol
W3C : The World Wide Web Consortium
WS : Web Service
WSDL : Web Service Description Language
WSRF : Web Service Resource Framework
XML : Extensible Markup Language

ŞEKİL LİSTESİ

Sayfa

ETMEN TABANLI BİR GRİD SİSTEMİ

ÖZET

İnternet ağının ve son derece hızlı bağlantı imkânlarının yaygınlaşmasıyla bilgisayarlar arasında kaynakların paylaşımı konusu oldukça popüler bir bilgisayar bilimleri konusu haline gelmiştir. Web servisi uygulamaları ve onların kaynak paylaşımına dayalı özelleşmiş versiyonları olan gridler gerek ticari gerekse de bilimsel alanda birçok uygulama alanı bulmuşlardır. Gridler sayesinde çok uzun sürebilecek birçok bilimsel hesaplama makul sürelerde tamamlanabilir hale gelmiştir. Yüksek işlemci kapasitesine sahip sanal süper bilgisayarlar pahalı donanım maliyetleri olmadan kolaylıkla oluşturulabilmiştir. Dağıtık hesaplamanın bir başka dalı olan etmen sistemleri de yapay zekâyla dağıtık sistemleri bir araya getirmiştir. Elektronik ticaretten savunma sanayiye kadar birçok alanda kendine yer bulan etmen sistemleri üzerinde yoğun miktarda çalışılan bir bilgisayar bilimleri konusudur

Bu tez çalışmasında dağıtık sistemlerin benzer ama farklı bakış açılarına sahip iki alanı olan gridler ve etmenler bir araya getirilerek, etmen tabanlı bir grid sistemi önerilip gerçeklemesi yapılmıştır. Öncelikle gridler ve etmenler hakkında literatür araştırılması yapılmıştır. Grid sistemlerinin genel özellikleri, güçlü ve zayıf yanları ortaya konulmuştur. Benzer bir çalışma da etmen sistemleri için yapılmıştır. Etmen sistemlerinin sınırları çizilip değerlendirmeler yapılmıştır. Daha önerilen sistemin tanımı yapılıp sınırları çizilmiştir.

Etmen tabanlı bir grid sisteminde olması gereken etmen tipleri, yani sistemin aktörleri belirlenmiştir. Daha sonra sistemde bulunan her bir etmen türünün görevleri ve sorumlulukları belirlenmiştir. Etmenlerin bu görevlerini yerine getirmesi için uygulamak zorunda oldukları protokoller belirlenmiştir. Etmenlerin yönetici etmenle ve diğer katılımcılarla nasıl etkileşime ve iletişime geçecekleri belirtilmiştir. Etmenlerin sisteme bağlanmasında ve sistemden ayrılmasında, görev atanmasında, görevlerin yürütülmesinde ve sonuçlarının bildirilmesinde izlenecek yöntem belirlenmiştir. Gridin yaşam döngüsü tanımlanmıştır. Aynı zamanda etmenlerin birbirleriyle hızlı ve etkili biçimde iletişime geçebilmeleri için esnek bir mesajlaşma altyapısı kurulmuştur.

Kademe kademe gerçekleme yapılarak bir önceki gerçeklemede karşılaşılan sorunlar çözümlenmeye çalışılmıştır. Sonuç olarak FIPA standartlarına uygun etmenler kullanılarak geliştirilmiş bir gird sistemi önerilmiştir. Böylece etmenlerin zeki ve ensek yapısıyla gridlerin kullanım alanlarının bir araya getirilmesiyle dağıtık hesaplama alanında kullanılabilir bir çalışma ortaya çıkartılmıştır.

AN AGENT BASED GRID SYSTEM

SUMMARY

Sharing computer resources becomes a popular topic of computer sciences while Internet and very fast connections become widespread. Web services and grids, which are a special version of web services based on resource sharing, have application areas both on scientific and commercial areas. Most of the scientific calculations take very long time using conventional methods and algorithms. However, after development of grid applications that kind of calculations can be done within acceptable times. Grids give us capability of building virtual super computers with high processing power without expensive hardware requirements. Another application of distributed computing is agents systems. Agent systems combine artificial intelligence and distributed computing together. Agents, which are widely used from defense industry to electronic trade, are a trendy topic of computer science.

In this thesis an intelligent grid system, which combines this two different perceptions of distributed computing (grids and agents) is proposed and implemented. Firstly, a research of prior works done. Strong and weak parts of grids and agent systems are identified. A formal and broadly accepted definitions and borders of these systems are given. Lastly, a grid system based on agents proposed.

Agent types and actors of the system are determined. After analyzing responsibility and task for each actor, the protocols defined for each analyzed role of agents. Interaction ways between agents and management etiquettes are described. Methods of connection, disconnection, task assignment, task executing and result delivery are declared. Life cycle of the proposed grid is characterized. In addition, an efficient and fast messaging infrastructure is development for effective agent communication.

The grid development is done systematically. Up to the final version, three versions of application are implemented. On each version, the problems of perspective defined and problems of prior versions fixed. Finally, a grid system developed using agents that are compatible with FIPA standards. By merging smart and flexible structure of agent system with stable and robust structure of grid, an agent based grid system developed.

1. GİRİŞ

Dağıtık sistemler ve dağıtık hesaplamalar uzun zamandır bilgisayar bilimlerinin popüler konularının ilk sıralarında bulunmaktadırlar. Özellikler iletişim alanındaki gelişmeler, çok geniş ve hızla bağlantı imkânlarının sunulması ve İnternetin bu denli yaygınlaşıp günlük hayatta yerini alması dağıtık sistemler konusuna daha fazla ilgi duyulmasına sebep olmuştur. Bu tezde dağıtık sistemlerin iki farklı kolu ve uygulaması olan gridler ve etmenler kullanılarak etmen tabanlı bir grid sistemi geliştirilmiştir.

Gridler farklı konumlarda bulunan çok çeşitli ve heterojen kaynakların bir ağ yapısı üzerinde ortaklaşa kullanılmasına, dağıtılmasına ve paylaştırılmasına olanak sağlayan sistemlerdir [1]. Etmenler ise kendilerine verilen görevleri fiziksel olarak farklı konumlara yerleşerek yerine getiren, çevresiyle esnek ve otonom ilişkiler kurabilen akıllı ve enkapsüle yazılım sistemleridir [2]. Bu tezde grid sistemleriyle etmen sistemlerini bir araya getirerek, iki farklı bakış açısının güçlü yönlerini de kullanan esnek, canlı ve akıllı bir etmen tabanlı grid sistemi oluşturmak hedeflenmiştir. Böylece kullanıcıların kendi ihtiyaçları doğrultusunda yazdıkları etmenleri kullanarak grid sisteminden istedikleri hizmeti kolay ve esnek biçimde alabilmeleri hedeflenmiştir.

Gridler kaynak paylaşımı esası üzerine kurulmuş sanal organizasyonlardır. En genel bakış açısıyla yazılım gridleri bir yazılım sisteminde olan tüm kaynakları paylaşabilmektedirler. Bu kaynaklar depolama alanı, işlemci gücü, grafik işleme yeteneği ya da özel donanıma bağlı veriler (giriş çıkış cihazlarından gelebilecek veriler v.b.) gibi oldukça çeşitlilik gösterebilmektedirler. Önerilen sistemdeki etmenli grid sistemi sadece işlemci gücüne dayalı kaynakların paylaşımına destek verebilmektedirler.

Grid sistemlerinde sistemin tutarlılığı ve kararlılığı önemli bir kriterdir. Çok yoğun ve önemli hesaplamaların yapıldığı bu tarz sistemlerde verilerin ve hesaplama sonuçlarının güvenliği ve güvenilirliği önemli bir husustur. Dolayısıyla grid sistemlerinde tüm akışı kontrol eden bir ya da birden fazla koordine yapı bulunur.

Ancak etmen sistemlerinden ve dolayısıyla etmenlerden beklenen en önemli özelliklerden birisi özerkliktir. Geleneksel etmen sistemlerinde hiçbir etmenin sistemin tamamına hükmetme yetkisi olmaz. Etmen sistemlerinde hiçbir etmen problemin çözümü için gereken tüm bilgilere sahip değildir. Etmenler bir araya gelip etkileşerek sosyal bir sanal organizasyon oluşturlar ve her birinin sahip oldukları kısıtlı bilgi ve yetenekle beraberce problemi çözerler.

1.1 Tezin Bölümleri

Tezde öncelikle grid ve etmen sistemleri tanıtılmıştır, daha sonra da önerilen sistemin genel ayrıntılarına yer verilmiş test sonuçları ve geleceğe yönelik iyileştirmeler tartışılmıştır

Grid sistemleri ayrıntılarıyla 2. bölümde tartışılmıştır. Grid sistemlerinin tanımı ve genel özellikleri anlatılmıştır. Gridlerin kullanım amaçları ve alanları tanıtıldıktan sonra mevcut grid yapıları ve özellikleri hakkında kısaca bilgi verilmiştir.

Bölüm 3. ’de ise etmen sistemleri ayrıntılarıyla tartışılmıştır. Etmenlerin ve etmen sistemlerinin taşıması gereken karakteristik özellikleri anlatıldıktan sonra geliştirilmiş mevcut etmen sistemlerinden bahsedilmiştir. Etmenlerin kullanım amaçları ve alanları tanıtılıp önerilen sistemde kullanılan JADE ana çatısının tanıtılması ve özelliklerinin belirtilmesi de 4. bölümde yapılmıştır.

Bölüm 5. önerilen etmen tabanlı grid sistemin tanıtılmasıına ayrılmıştır. Öncelikle etmen tabanlı grid sistemleri, kullanım amaçları ve faydaları üzerinde durulmuştur. Sonra önerilen sistemin ayrıntılarının, sistemin parçalarının ayrıntılı bir biçimde tanıtılmasına başlanmıştır. Sistemi oluşturan etmenler, bunlar arasındaki ilişki ve bu bu ilişkileri düzenleyen protokollerden bahsedilirken gerekli görülen noktalarda gerçekleme ayrıntısına yer verilmiştir. Bu bölümde aynı zamanda önerilen sistemin gerçeklenmesi sırasında oluşan sorunlar ve versiyonlar arası bu sorunların nasıl giderildiğinden de bahsedilmiştir. 5. bölümde yapılan çeşitli testlere ve sonuçlarına yer verilmiştir.

Son bölüm olan 6. bölümde ise sonuçlara ve önerilere yer verilmiştir. Sistemin genel bir değerlendirmesi yapılıp güçlü ve zayıf olduğu yönler belirtilmiştir. Ayrıca sistem üzerinde gelecekte yapılabilecek olası iyileştirmeler ve geliştirmeler de bu bölümde belirtilmiştir.

2. GRİD SİSTEMLERİ

2.1 Grid Sistemlerinin Tanımı

Grid sistemleri için kullanım amacına göre birden fazla tanımlama yapılabilinir. Ancak hepsinin ana yapısı grid sistemlerinin ilk uygulandığı elektrik alanındaki kavramlara dayanır. Elektrik grid sistemlerinde her bir kullanıcı prizlerden gelen enerjinin nereden geldiğini ve nasıl üretildiğini bilmeden, arkadaki karmaşık yapıdan bağımsız olarak sistemden hizmet alır.

Bu açıdan bakıldığında grid hesaplama sistemleri kullanıcılara işlemci gücü, saklama alanı, veri ve uygulama gibi birçok kaynağa erişmesini sağlar. Bu hizmet sırasında kullanıcının bu kaynağın nerede olduğuna ve/veya bu kaynağın altında nasıl teknolojiler olduğuna dair bir bilgisi olmasına gerek yoktur [3].

Yazılım grid sistemleri ise her çeşit bilgisayar kaynağını (saklama alanı, işlem gücü v.b.) heterojen bir ağ yapısı üzerinden paylaştırmayı hedefler [4]. Grid sistemleri fiziksel olarak ayrık olan tüm bu heterojen kaynakları kullanıcıya sanal bir kaynakmış gibi göstermek suretiyle ihtiyaçları oldukları hizmete/kaynağa istedikleri zamanda erişmesini sağlar [5].

Grid sistemleri bu hizmetleri kullanıcılarına düşük maliyetlerle ve yüksek kaliteyle sunmayı hedeflemektedirler [6]. Yazılım gridleri geleneksel dağıtık sistemlerden farklı olarak esas amacı büyük ölçekli kaynak paylaşımı ve yüksek performans hedefleri olan, yeni gelişmekte olan bir dağıtık sistemdir [7].

2.2 Grid Sistemlerinin Kullanım Amaçları ve Alanları

Elektronik ticarette ve bilimsel hesaplamalarda sıklıkla dağınık konumlarda bulunan heterojen kaynakları, hizmetleri ve bilgileri bir arada kullanma ve paylaşma ihtiyacı güdülmektedir. Bu işlem tüm bu servislerin farklı platformlar üzerinde çalıştığı düşünüldüğünde oldukça meşakkatli bir yapısı olduğu ortadadır [1]. Yakın zamana kadar yazılımcılar hedef platformlarının homojen, güvenilir, güvenli ve merkezi bir yönetime sahip olduklarını varsaymaktaydılar.

Ancak günümüzde uygulamalar birçok farklı teknolojiyle, belirli bir platforma yönelik olabilmektedirler. Tüm bu yapıların kendilerine özgü kaynak kullanımı ve yönetimi yapıları vardır. Fakat tüm bu farklılaşmaya karşı günümüzün ihtiyaçları farklı platformlarda farklı altyapılarla, farklı API'ler kullanılarak geliştirilen tüm bu uygulamaların uyum içinde, ortak bir amaca yönelik çalışmasını gerektirmektedirler. Bu bağlamda bu heterojen kaynakların paylaşımını hedefleyen grid sistemlerinin gerekliliği açıktır [1].

Gridlerin ilk gerçeklemeleri belirli bir şirkete, gruba ve organizasyona yönelik hizmetler sunmaktayken, günümüzde organizasyonlar arası gridler gerçeklenmektedir. Bu gridler geleceğin ticaret, iş ve bilgisayar sektörünün önemli bir parçası olacağı düşünülmektedir. Grid sistemlerinin sağladığı ve sağlaması gereken şu özellikleri vardır [3].

- **Mevcut kaynakların kullanımı:** Grid uygulamalarının kullanım alanlarından birisi var olan bir yazılımın başka bir bilgisayarda yürütülmesi olabilir. Kullanıcı bilgisayarın işlem yükünün fazla olmasından ötürü gereken bir program başka bir bilgisayarda koşturulabilir.

- **Paralel CPU kapasitesi:** Gridlerin sunduğu potansiyel büyük CPU kapasitesi onların en çok ilgi çeken yanlarından birisidir. Endüstrideki gelişmeler yüksek işlemci kapasitesi gerektiren bilimsel çalışmaların yanında bio-medikal, finansal modelleme ya da görüntü işleme alanlarında da büyük CPU kaynağına ihtiyacı arttırmıştır.

- **Ortaklıklar için sanal kaynak ve altyapı:** Gridlerin sağladığı bir başka yararsa daha çok katılımcı arasında bir işbirliği ortamını oluşturmaktır.

- **Ekstra kaynaklara erişim:** Aynı CPU kapasitesinin paylaşımında olduğu gibi bilgisayar sistemlerinin diğer kaynaklarına da (veri, disk kapasitesi...) erişimi sağlar.

- **Yük dengeleme:** Gridler katılımcıları arasında iş ataması sırasında akıllı yük dengeleme algoritmaları yürüterek işlemlerin sistemi en az yoracak biçimde hızlıca tamamlanmasını sağlarlar.

- **Güvenilirlik:** Geleneksel yüksek kapasiteli hesaplama sistemleri güvenilirliği

arttırmak için pahalı donanımlar kullanmaktadırlar. Bu sistemlerde çift depolama üniteleri, çift güç üniteleri ve yedek işlemci yapıları kullanılarak sistemde oluşacak hataları en az kayıpla giderilmesi çalışılmaktadır. Ancak grid teknolojisi fiziksel olarak ayrı konumlarda bulunan katılımcılara sahip olmalarından ötürü birisinde oluşacak elektriksel ya da mekanik bir arıza sistemin geri kalanını etkilememektedir. Grid yönetici yazılımları böyle bir hatayı kolayca sezip aynı işi başka bir katılımcıya verebilme kapasitesine sahiptirler.

- **Yönetim:** Gridlerin yönetim yazılımları sistemlerin yük durumlarının, kaynaklarının kolaylıkla izlenebildiği alt yapılar sunmaktadırlar.

2.3 Grid Sistemi Uygulamaları

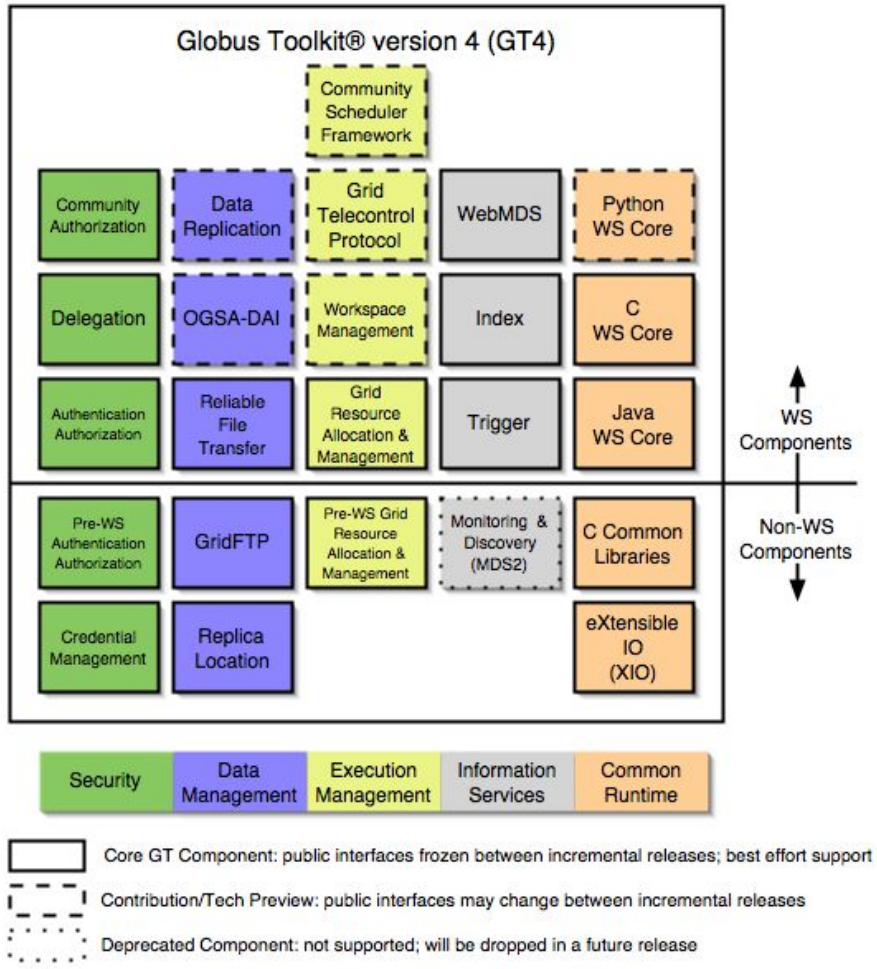
Günümüzde birçok akademik ve ticari ürün sundukları ara yüzlerle, protokollerle ve servislerle kullanıcılarına heterojen ve müstakil ağlara dağıtılmış kaynakların ortaklaşa kullanılmasını sağlayan grid sistemleri oluşturmuşlardır [4].

Jia Yu ve Rajkumar Buyya tarafından yapılan "*A Taxonomy of Workflow Management Systems for Grid Computing*" isimli çalışmada grid sistemlerinin sınıflandırılması yapılmıştır [8]. Bunlar arasında Globus, Condor-G ve Legion en çok bilinenleridir [4].

- **Globus Toolkit:** Globus Alliance tarafından geliştirilen [9] bu topluluk destekli, açık mimarili ve açık kaynak kodlu bir kütüphanenin ve uygulamanın oluşturduğu bir yazılım kümesidir. Genel olarak güvenlik, bilgi çıkarımı, kaynak ve veri yönetimi, iletişim, hata tespiti ve taşınılabilirlik konuları üzerine eğilen Globus toolkit yüzlerce site ve birçok büyük grid projelerinde kullanılmaktadır [1]. Globus toolkitin mimarisi hakkında daha detaylı bilgi Şekil 2.1' de verilmiştir.

- **Condor-G:** Condor ve Globus projelerinin evliliği sonucu ortaya çıkan bir grid yapısıdır. Güçlü ve tam donımlı bir görev dağıtıcı olan bu sistem farklı sitelerde koşan binlerce işi koordine ederken bu görevlerin izlenmesi, kayıtlarının düzenlenmesi, politika atanması gibi karmaşık birçok işi halleder [10].

- **Legion:** Kullanıcılarına kolay kullanım imkanı sunan bir portal projesidir. Virginia Üniversitesinde geliştirilen bu sistem MPI ve PVM de yazılan kodları destekleyerek kullanıcılarına büyük ölçekli ve karmaşık kaynak havuzlarından yararlanma imkanı sunar [11,12].



Şekil 2.1 : Globus Toolkit Yapısı [9]

Grid sistemlerinde GRAM GASS MDS-2 ve GSI isimli grid protokoleri yaygın olarak kullanılmıştır. Globus Toolkit bu protokollerin gerçeklemesine ilişkin algoritmalar içermektedir [10].

- **GRAM:** Grid kaynak tahsis ve yönetim (The Grid Resource Allocation and Management) protokolü bir işlem isteğinin uzaktaki bir kaynağa atanmasını ve hesaplamanın gözlemlenmesini düzenler.

- **GASS:** Ekstra disk alanlarına küresel erişim (Global Access to Secondary Storage) uzaktaki HTTP, FTP yada GASS sunucularına erişimi ve ordan

hizmet almayı düzenleyen protokoldür.

- **MSD-2:** Grid kaynaklarının durumu ve yapıları hakkında bilgilerin alınmasıyla ilgili işlemleri düzenler.

- **GSI:** Grid güvenlik altyapısı (Grid Security Infrastructure) açık anahtar şifreleme yöntemiyle kimlik denetleme ve yetkilendirme işlemlerini düzenleyen protokoldür

2.3.1 Web servisleri

Daha önce de belirtildiği gibi Globus Toolkit, Open Grid Services Architecture (OGSA) tarafından gerçekleştirilmiş ve kolaylıkla bir Grid altyapısı oluşturulma amacı güden bir programlar topluluğudur. OGSA ise Grid sistemlerinde Web servis konsepti ve teknolojileri üzerine geliştirilmiş bir sistem sunmaktadırlar [13].

Web servisleri yeni bir dağıtık hesaplama yöntemi önermektedir. Özellikle heterojen dağıtık hesaplamalarda kullandığı XML gibi basit Internet temelli standartlarla DCE, CORBA ya da Java RMI gibi diğer yaklaşımlardan ayrılmaktadır.Aralarında SOAP, WSDL, WS-Inspection, WSRF'un da bulunduğu gibi birçok Web servis standartı W3C tarafından tanımlanmaktadır [1]. Bu standartlar şu şekilde özetlenebilir;

- **The Simple Object Access Protokol (SOAP):** Bir servis sağlayıcısı ve istemcisi arasındaki mesajlaşmayı sağlar. SOAP. Uzak mesaj çagrıları (Remote Procedure Call - RPC) için gönderilen XML verilerine uygulanan basit bir protokoldür [1].

- **Web Service Resource Framework (WSRF):** Web servisi kaynak ana çatısı, Kaynakları tanımlamak, denetlemek ve yönetmek için düzenli bir mekanizma tanımlar [2].

- **Web Service Description Language (WSDL):** Web servisi tanımlama dili, web servislerini tanımlanmasını, bulunmasını ve uzaktan yönetilmesini sağlayan XML tabanlı dokumandır [1,2].

- **WS-Inspection (Web servisi denetim dili)**: Web servislerini bulmak ve bunlar arasındaki ilişkiyi tanımlamak için kullanılan XML tabanlı bir dildir [1].

3. ETMEN SİSTEMLERİ

Bilgisayar dünyasının trendi tek bilgisayarda kendi başına çalışan programlardan değişik konumlarda dağıtılmış açık dinamik sistemler için yapılan programlara doğru kaymaktadır [14]. Etmen tabanlı sistemler 1990'lardan itibaren bilgisayar dünyasının en canlı ve önemli araştırma alanlarından biri olmuştur [14,15]. Etmen sistemleri nesneye dayalı programlamadan beri en önemli gelişmelerden birisi olduğu belirtilmektedir. Etmen sistemleri konsepti, bilgisayar ağları, yazılım mühendisliği, nesneye dayalı programlama, yapay zeka, insan bilgisayar etkileşimi, dağınık ve paralel sistemler, mobil sistemler, bilgisayar destekli ortaklıklar, kontrol sistemleri, karar verme, bilgi çıkarımı ve elektronik ticaret gibi birçok bilgi teknolojisinde geniş kullanım alanı bulmaktadır [14]. Etmen sistemlerinin otonom, müşterek, zeki ve hedef tabanlı yapısı çeşitli alanlardaki yeni nesil yazılımlar için umut verici çözümler sunmaktadır [15].

3.1 Etmenler

Etmenler yaklaşık elli yıldır bilgisayar bilimlerinde incelenen ve doksanların sonlarına doğru popülaritesi artan bir araştırma alanı olmasına rağmen, tüm çevrelerce kabul gören ortak bir tanımda karar kılınamamıştır [16]. Franklin ve Graesser, etmen sistemi olarak nitelendirilen birçok yazılımın aslında etmen sistemi gibi tasarlanmadığını belirtmektedirler [17]. En yaygın kabul gören tanımı [18] ise Jennings ve Wooldridge tarafından yapılan tanımdır [2]; "*Etmenler kendilerine verilen görevi yerine getirmek için belirli yerlere yerleştirilmiş, çevresiyle esnek, otonom ilişkiler kurabilen enkapsüle bilgisayar sistemleridir.*"

Etmenler bu özerk, sosyal, çevresini algılayan, kendisini eğitebilen ilerici yapısıyla diğer etmenlerle, insanlarla ve yazılımlarla etkileşimde olabilen ve hatta ortaklaşa hareket edebilen yazılım parçalarıdırlar [14,19].

Bir yazılım sisteminin etmen olarak adlandırılabilmesi için şu özellikleri sağlaması gerekmektedir [16,20].

- **Otonomluk – Özerklik:** Etmenlerin diğer etmenlerin erişimine açık olmayan nitelikleri vardır. Etmenler bu niteliklerin değerlerine bakarak başka yapılarla (insan, bilgisayar, diğer etmeler) doğrudan etkileşime girmeden karar verebilme yetenekleridir.
- **Duyarlılık – Algılayıcılık:** Etmenler fiziksel dünya, kullanıcı ara yüzlü bilgisayar programları, diğer etmenlerden oluşan toplulukların bulunduğu ortamlarda yer alabilirler. Etmenler bulundukları bu ortamı algılama ve oluşan değişikliklere karşı tepki verebilme ve bulunduğu çevrede değişiklik yapabilme yeteneklerine sahiptirler.
- **Akıllı davranışlılık:** Etmenler çevrelerindeki olan değişimlere sadece basit davranışlarla tepki göstermezler, aynı zamanda belirli bir amaca yönelik hareket ederler, zekidirler.
- **Sosyallik:** Etmenler diğer etmenlerle (bazen de insanlarla) etmen dili denilen dillerle iletişime geçebilirler ve hatta amaçlarına yönelik ortaklıklar da kurabilirler.

3.2 Çoklu Etmen Sistemleri

Daha önce de bahsedildiği gibi etmenler sadece çevrelerini algılamak ve onlara tepki vermekte yetinmezler. Etmenler sosyal ilişkiler kuran aktif yazılım bileşenleridir. Etmenler arasındaki ilişki bazen rekabet, bazen ortaklık temelli bazen de her iki temele birden dayanan şekildedir. Çoklu etmen sistemleri, birbirleriyle iletişim kuran birden fazla zeki etmenin bir araya gelerek oluşturduğu sistemlerdir. Etmenler kendi şahsi yetenekleri ve bilgilerinin çok daha fazla isteri olan problemleri bir araya gelerek oluşturdukları bu sistemlerle çözebilirler. Çoklu etmen sistemlerinin şu özellikleri vardır [19,21].

- Herbir etmenin problemin tamamını çözmek için eksik bilgisi ve yeteneği vardır.
- Tüm sistemi kontrol edemezler.
- Veri merkezileşmemiştir. Bir tek noktadan tüm veriye erişim mümkün değildir.
- Hesaplamalar asenkron yapılır.

3.2.1 Etmen iletişim yöntemleri

Etmen sistemlerinin gücü ve etkinliği etmenler arasındaki iletişime bağlıdır. Etmenler diğer etmenlerle, insanlarla, kaynaklarla yani çevresiyle iletişim halinde olmalıdırlar. Etmen sistemlerinin kullanım alanları geliştikçe ve daha zor problemlere uygulandıkça daha güçlü iletişim yöntemleri ve dilleri geliştirilme gereği açıktır.

KQMLve FIPA ACL günümüzde yaygın kullanımı olan iki etmen iletişim dili olarak ön plana çıkmaktadır. 1990'larda Amerikan hükümeti tarafından kurulan ARPA Bilgi paylaşım grubu tarafından geliştirilen KQML (Knowledge Query and Manipulation Language) etmenler arası bilgi ve mesaj aktarımında kullanılan bir dil ve protokol tanımlar [14,22].

Bir diğer popüler etmen iletişim dili ise FIPA tarafından geliştirilien FIPA ACL isimli açık standartları olan dildir [14,23].

3.3 Etmen Sistemleri Ana Çatıları

Etmen sistemlerinin bilgisayar bilimlerinde popülaritesinin artmasından sonra birçok etmen sistemi ana çatısı geliştirilmiştir. Bu ana çatılardan öne çıkanları şu şekilde özetlenebilir:

- **AgentSpeak(L):** Olaylar ve davranışlar üzerine dayalı bir etmen geliştirme dili tanımlar. Etmenler davranışlarını bu dille geliştirilmiş programlara göre yaparlar [24].

- **3APL:** Kavramsal etmenler oluşturmak için kullanılan bir dildir [26].

- **JACK:** Kullanıcılarına sağlam, kararlı, hafif, esnek ve geliştirlebilir bir anaçatı sunar [27, 28].

- **PRS:** The Procedural Reasoning System – Prosedural sonuç çıkarma sistemi, dinamik bir domainde hareketler ve prosedürler hakkında sonuç çıkarımı için genel bir mimari tanımlar [29].

- **PRS Lite:** PRS in bazı özelliklerini sağlamayan basitleştirilmiş versiyonudur [30].

- **dMars:** PRS projesinin devamı niteliğinde olan bu anaçatı BDI (Belief Desire Intention) kavramı üzerine kurulmuştur [31].

- **JAM:** PRS mantığı üzerine geliştirilmiş başka bir sistemdir [25].

- **SPARK:** BDI modelinde geliştirilmiş ve PRS türünden etmen sistemlerinden esinlenmiş bir sistemdir [25].

- **JADE:** Java ile geliştirilmiş bu anaçatı FIPA spesifikasyonlarıyla tam uyumludur. Günümüzün lider etmen anaçatısı olan JADE, yaşam döngüsü servisleri, iletişim ve ontoloji desteği, güvenlik ve platformlar arası etmen taşınırlığı desteği ve sistem yönetimi modülleriyle tam bir etmen anaçatısıdır [21,32].

Bunların yanında var olan bazı çalışmalar söyle özetlenebilir. Mitkas ve arkadaşlarının "*An Agent Framework for Dynamic Agent Retraining Agent Academy*" isimli çalışmasında JADE tabanlı bir etmen sistemi ana çatısı önerilmiştir [33]. Nienaber ve Barnard "*A Generic Agent Framework to Support the Various Software Project Management Processes*" isimli çalışmalarında yazılım yönetim prosesleri için genel bir etmen sistemi önermişlerdir [34]. "*A Java Framework for Multi-agent Systems"* çalışmasında ise Avancini ve Amandi tarafından Java dilinde geliştirilmiş bir etmen sistemi tanıtılmıştır [35]. Java diliyle geliştirilmiş bir diğer etmen ana çatısı ise açık kaynak kodlu olan Fleeble [36] iken Lost Wax Agent Framework ise Lost Wax tarafından geliştirilen ticari bir etmen ana çatısıdır [37].

4. JADE ANA ÇATISI

JADE - Java Agent Development Framework (Java Etmen Geliştirme Ana çatısı) FIPA standartlarına tam uyumlu, çoklu etmen sistemi geliştirme ana çatısıdır. JADE biri FIPA uyumlu etmen platformu, diğeri ise Java ile etmen geliştirme paketi olmak üzere iki ana yapıdan oluşur. JADE, tamamiyla Java dilinde yazılarak bu dilin esnek yapısını ve soyut ara yüzlerini, hazır paketlerini kullanarak kolaylıkla etmen geliştirme yapısı sunmayı hedeflemiştir [38].

JADE kullanıcılarına şu özellikleri hazır olarak sunmaktadır [38].

- Dağıtık etmen platformu: Sistem birden fazla katılımcı makine arasında bölüştürülebilinir.
- Grafik ara yüzü ile etmenlerin kontrolü.
- Platformlar arası taşınırlık: Etmenlere ilişkin kod ve verinin katılımcı bilgisayarlar arasında göç edebilmesini sağlar.
- Davranış modelleri ile paralel ve eş zamanlı etmen aktivitesini düzenler.
- AMS (Agent Management System – Etmen Yönetim Sistemi), DF (Directory Facilitator – Sarı Sayfalar Servisi), ACC (Agent Communication Channel – Etmen iletişim kanalı) başta olmak üzere FIPA standartlarına uyumludur.
- Aynı etmen platformu içerisinde etken ACL mesajlaşma sistemi sunar.
- FIPA standartlarına göre etkileşim için kullanıma hazır yapılar içerir.
- Etmenlerin AMS'ye otomatik bağlanıp, ayrılmalarını sağlayan sistemler.
- FIPA standartlarına uygun etmen kimlik ataması yapar.
- Uygulamaya özel içerik dilleri ve ontolojiler tanımlamaya izin verir.
- Prosesler arası etkileşim için sunduğu ara yüzlerle harici uygulamaların da etmen yaratıp sisteme dâhil olmasını sağlar.

4.1 JADE Ana Çatısının Genel Yapısı

Jade aşağıdaki yazılım paketlerinden oluşur [38].

- jade.core paketi sistemin çekirdeğini oluşturur. Agent isimli sınıfı içerir. Yazılacak sistemdeki tüm etmenler bu sınıftan türetilirler. jade.core.behaviours isimli alt paketindeki Behaviour sınıfı ise etmenler tarafından yürütülecek görev sınıfları için ana sınıfıdır. Bu pakette çeşitli amaçlara yönelik hazırlanmış davranış-görev sınıfları bulunur.

- jade.lang.acl paketi ise FIPA ACL standartlarına uygun olarak etmen haberleşmesini sağlayan sınıfları barındırır.

- jade.content paketi uygulamalar tarafından tanımlanan ontolojileri ve içerik dillerini desteklemek için kullanılan sınıflardan ve alt paketlerden oluşur.

- jade.domain paketi ve alt paketleri FIPA standartları tarafından AMS ve DF özelliklerini yani yaşam döngüleri ve sarı sayfalar servislerini sağlayan etmen sınıfları tutarlar.

- jade.gui paketi oluşturulacak uygulamalar için kullanıcı ara yüzü geliştirmekte kullanılan sınıfları barındırır.

- jade.mtp paketi sistemlerin JADE ana çatısına entegre çalışması için gerçeklemeleri gereken Mesaj İletim Protokolüne dair ara yüzleri ve bu ara yüzlerinin gerçeklenmiş bazı örneklerinden oluşur.

- jade.proto paketinin içerdiği sınıflar standart etkileşim protokollerini gerçeklerler. Aynı zamanda kullanıcılar tarafından protokol tanımlanması için gereken ana sınıflar da bu pakettedir.

- jade.wrapper paketi JADE'e üst seviye özellikler eklemek ve başka uygulamaların JADE etmenlerini yürütmelerini sağlayan sınıflardan oluşur.

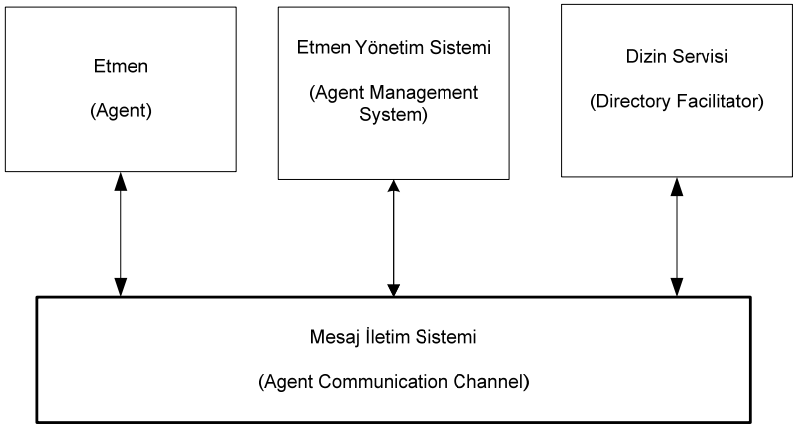
Bunların yanında JADE de kullanıma hazır yönetimsel ve etmen sistemi geliştirme amaçlı bazı araçlar vardır

Herbiri jade.tool paketindeki ilgili alt pakette bulunan bu araçlar özetle şu şekildedir.

- RMA (Uzaktan yönetim etmeni): Etmenleri uzaktan yönetmek için bir kullanıcı ara yüzü sunar.
- The Dummy Agent (Sahte Etmen): Sistemi gözlemleme ve hata ayıklama için kullanılır.
- Sniffer (Koklayıcı): İletimde olan ACL mesajlarını yakalayıp grafiksel olarak sunar. Hata ayıklamada kullanılır.
- Introspector (Gözlemci): Etmenlerin yaşam döngülerini gözlemleyen bu uygulama da hata ayıklamada kullanılır.
- DF GUI: Sarı sayfalar servisinin durumunu kullanıcı ara yüzüyle görüntüler.
- LogManagerAgent: Sistem kayıtlarını tutan etmendir.
- SocketProxyAgent: JADE platformu ile herhangi bir TCP/IP bağlantısı arasında iki yönlü bir ana kapı (gateway) görevi gören etmendir.
- Ayrıca JADE Java dilinin Serializable (seri hale getirilip kullanılabilme) özelliğini kullanarak mesajların içeriğine istenilen bir nesneyi ekleyebilmektedir.

4.2 FIPA Standartlarına Uyumlu Etmen Yapısı

FIPA tarafından önerilen standart bir etmen sisteminin bileşenleri Şekil 4.1'de görülebilinir [38].



Şekil 4.1 : FIPA Standartlarında Etmen Sistemi.

AMS – Agent Management System – Etmen Yönetim Sistemi denilen bir etmen tüm etmen platformunun yönetiminden sorumludur. Her sistemde bir ve yalnız bir tane AMS bulunur. AMS etmenler için yaşam döngüsü servisleri sunar, etmenlere kimlik atanması (AID) ve durumlarının takip işlemlerini yapar. Platforma bağlanacak her etmen AMS den geçerli bir kimlik almak zorundadır.

DF (Directory Facilitator – Dizin Servisi) Etmenler arası sarı sayfalar servisi sunar

Mesaj İletim Sistemi – ACC (Agent Communication Channel) platform içinde mesaj iletimi yapısını kuran bileşendir.

JADE FIPA standartlarıyla ve yukarıda tanıtılan mimariyle tam uyumludur. JADE platformu çalıştırıldığında AMS, DF ve ACC bileşenleri yaratılır ve çalıştırılır. Aynı zamanda bu bileşenlerin her biri farklı bilgisayarlarda çalıştırılabilecek yapıya sahiptirler [38].

5. ÖNERİLEN ETMEN TABANLI GRİD SİSTEMİ

Yapılan çalışmada etmenler kullanılarak bir grid sistemi önerilmiş ve gerçeklemesi yapılmıştır. Etmenlerle geliştirilmiş bir grid sisteminin gerekliliği bu kısımda sorgulanacak ve seçilen etmen ana çatısı olarak JADE'in neden tercih edildiği açıklanıp gerçeklenen sistem tanıtılacaktır.

5.1 Etmen Sistemleri ve Gridler

Dağıtık sistemlerde bağımsız birçok yazılım bileşeni bir araya gelerek ortak bir amaca yönelik çalışmaktadırlar. Bu tarz sistemler günümüzün bilgi sistemlerinde yaygın kullanım alanı bulmaktadırlar

Grid mimarilerinin temel olarak dört adet yönetimsel katmanı bulunmaktadır. Çatı katmanı gridi oluşturan hesaplama, ağ, kod depoları gibi ana kaynaklardan oluşur. Bağlantı katmanı kolay ve güvenli bir iletişim yapısı sunar. Kaynak katmanı bireysel kaynaklardan oluşurken müşterek kaynak katmanı ise dizinleme servisi, kaynakların yayımlanmasını, izlenmesini ve değerlendirilmesini sağlayan servisleri sunar. Tüm bu servislerin ve katmaların uyum içinde çalışması özünde zor bir iştir. Özellikle kaynakların sayısının artması ve geniş alana yayılmış dağıtık sistemlerin yönetimi çoklu etmen sistemleri tarafından etkili biçimde yapılabilir [39].

Grid sistemleri genel olarak dağıtık yapının güvenliğine, güvenilirliğine ve bileşenlerinin uyum içinde birlikte çalışabileceği bir alt yapıya önem verirler. Gridlerin ana amacı dinamik ve çok katılımcılı sanal organizasyonlarda kaynak paylaşımı ve problem çözme sistemleri yaratmaktır [2]. Bu nedenden ötürü grid sistemlerinin tanımladığı protokollerde veri güvenliği, kararlılık daha çok öne çıkan tasarım isterleri olarak öne çıkmaktadır.

Etmenler ise daha kaygan zeminler üzerinde inşa edilmiş sistemleri oluşturmaktadırlar. Etmen sistemleri dikkatlerini esnek ve bağımsız çalışan yazılım bileşenleri oluşturmaya daha çok yoğunlaştırmışlarıdır. Dolayısıyla etmenler sistem ve problem hakkında sahip oldukları sınırlı bilgi ve birikimle bireysel işlemlerini

yaparken tüm sistemin kazanımın yanında kendi kazanımlarını da düşünen sosyal yazılım bileşenleridir [2]. Etmenlerin tasarımında esas tasarım kriteri esnekliktir. Klasik etmen sistemlerinde katılımcı etmenler dağıtık yerlerde yaptıkları işler sistemin ana amacı için önemli olsa da varlığı tüm sistemin oluşturulma amacı için kesin gerekli değildir.

Örneğin sahada yayılmış ve çevreden aldığı çeşitli bilgileri işleyerek gönderen etmenlerde oluşan bir sistemde sahanın yakın yerlerinde birden fazla etmen bağımsız olarak çalışmaktadır. Böyle bir sistemde etmenlerden birisinden bilgi gelmemesi ya da bilgide gürültü olması tüm sistem için çok önemli değildir. Ancak bir bilimsel hesaplama yapan grid sisteminde katılımcılardan birisinin sonuç dönmemesi ya da iletişim sırasında sonucun bir şekilde değiştirilmesi ya da çalınması sistem için son derece önemli sonuçlar doğurabilir ve hesaplamaların yanlış olmasına sebep verebilir. Böyle bir sistemde olası hataların sezilmesi, giderilmesi ve güvenlik son derece önemlidir.

Görüldüğü gibi her iki sistemde farklı tasarım kararları bulunsa da temel amaçları sanal organizasyonlar veya birlikler yaratarak bir problemi çözmeye ya da bir amacı yerine getirmek üzere dağıtık sistemler oluşturmaktır [2,7]. Önerilen sistemin amacı dağıtık hesaplamaya iki ayrı bakışı olan bu sistemlerin karakteristik özelliklerini birleştirerek zengin, esnek, mobil, güvenli ve güvenilir bir grid sistemi oluşturmaktır.

5.2 Benzer Sistemler

Etmen ve grid sistemlerin dağıtık sistemlerde kullanımı son yılların bilişim alanında ve akademik çalışmalarında yaygın araştırma ve kullanım alanı bulmaktadır.

Athanaileas ve arkadaşları "*An agent-based framework for integrating mobility into grid services*" isimli çalışmalarında grid sistemlerine etmen tabanlı bir taşınırlık özelliği eklemeyi tartışmışlardır [7].

Internet tabanlı grid sistemleri için etmen destekli *AgentScape* isimli uygulama B. J. Overeinder ve arkadaşları tarafından geliştirilmiştir [39].

Etmen tabanlı bir diğer grid sistemi ise *DARPA* tarafından geliştirilen *Control of Agent-Based Systems (CoABS)* isimli çalışmadır [40].

Fukuda ve Smith tarafından geliştirilen *UWAgents*, grid sistemleri oluşturmak için Java tabanlı bir mobil etmen uygulamasıdır [41].

Poggi, Michele ve Turci JADE ana çatısının grid sistemleri için nasıl geliştirilebileceği üzerine çalışmalar yapmışlardır [32].

5.3 Etmen Tabanlı Grid Sisteminin Temel Yapısı

Tez çalışmasında JADE etmen ana çatısı kullanılarak bir grid sistemi önerisi ortaya atılmıştır. Bölüm 4. 'te belirtildiği gibi JADE, FIPA standartlarına tam uyumlu ve aktif geliştirilme sürecinde olan bir etmen ana çatısıdır. Uygulamanın JADE ana çatısını kullanılarak geliştirilmesinin nedenleri şu şekilde özetlenebilir.

- JADE, FIPA standartlarına uygun yazılmış başka etmenlerle FIPA ACL (etmen iletişim dili) ile iletişim kurabilir. Böylece sisteme platformdan ve yazılım dilinden bağımsız olarak başka etmenlerin de dahil olması mümkündür. Bu nedenle sistem bu standartlarla uyumlu şekilde yazılmış etmenlerle çalışabilir.

- JADE ile gelen hazır uygulamalar (RMA, Dummy Agent v.b.) hata ayıklama ve sistem geliştirme işlemlerini hızlandırmaktadır.

- JADE Telecom Italia SpA tarafından aktif geliştirilme aşamasında olmasından ötürü geliştiricileri ve kullanıcıları tarafından sağlanan topluluk desteğine sahiptir [42].

5.3.1 Sistemin katılımcı etmenleri

Sistemdeki etmenleri yöneteci, temsilci, istemci ve işçi olmak üzere dört gruba ayırmamız mümkündür. Bu etmenler grid sisteminin çalışması süresince farklı farklı görevler üstlenmektedirler. Her bir türün görevleri genel hatlarıyla şu şekildedir

1. **Yönetici Etmen:** Sistemdeki diğer etmenlerin sisteme bağlanıp hizmet almasını ve vermesini koordine eden, gridin akışını düzenleyen etmendir. Sistemde bir tane yönetici etmen bulunur.
2. **Temsilci Etmenler:** Yönetici etmene yardım amacıyla yönetici tarafından yaratılıp sisteme dâhil edilen etmenlerdir. Sisteme bağlanan her etmen için (istemci veya işçi etmenler) bir temsilci etmen yaratılır.

Temsilci etmen sisteme bağlanan etmen için yönetim tarafında yapılması gereken tüm işleri (mesaj transferi, veri akışı v.b) yaparak yönetici etmenin yükünü hafifletirler. İlgili etmenleri sistemden ayrıldıktan sonra da yok edilirler.

3. **İstemci Etmenler:** Yürütülmesini istedikleri görevleri ve bu görevin gerçeklenmesi için gereken bazı özellikler varsa bunları da yönetici etmene bildirirler. Yönetici etmen ve temsilci etmenler, işçi etmenler arasında görevi yerine getirebilecek ve ilgili özellikleri sağlayan etmenleri bulup, çalıştırılmak üzere görevleri bu işçilere iletirler.
4. **İşçi Etmenler:** Sisteme bağlanarak görev alan etmenlerdir. Uygun olduklarında ve sahip oldukları özelliklere göre bir görev yönetici etmen tarafından kendilerine atandığında bu görevi yürütüp cevabını ilgili etmenlere bildirirler.

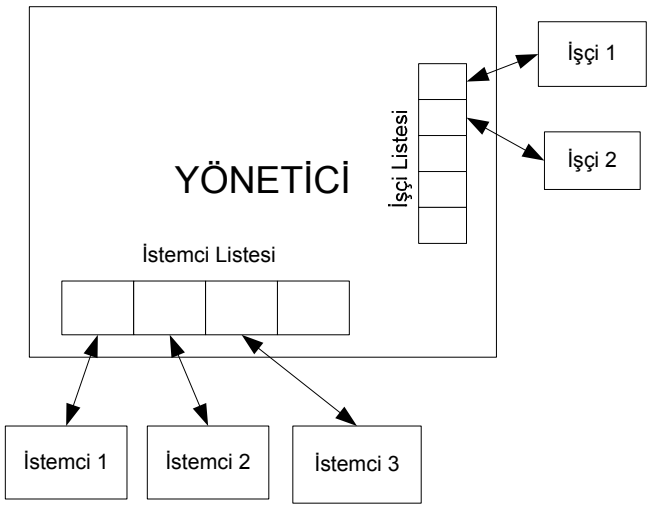
5.3.2 Sistemin versiyonları

Yazılım, nihai versiyonuna kadar üç ayrı versiyonla gerçeklenmiştir. Versiyonlar arasında sistemin tasarım ve analiz kararlarında bazı değişiklikler yapılmış ve bu değişikliklerin önerilen grid sisteminin güvenilirliğini, kararlılığını nasıl etkilediği gözlemlenerek sistemin iyileştirilmesi hedeflenmiştir.

İlk versiyonda bir tane yönetici etmen oluşturulmuştur. Hizmet alan (istemci) ve hizmet veren (işçi) etmenler sisteme bu yönetici etmene yolladıkları mesajlarla bağlanırlar. Sisteme bağlanma prosedürleri gerçekleştikten sonra istemci etmenler işletilmesini istedikleri görevleri yönetici etmene bildiriler. Yönetici etmen ise işçi etmenler arasında uygun bir etmen bulunca görevleri çalıştırılmak üzere ilgili işçi etmene yollar.

Bir işçi etmene bir anda tek bir iş yapacak şekilde iş dağıtımı gerçeklenmiştir İşçi etmen de sonucu belirli protokole göre istemciye iletir. Bu versiyonda iş atanması ve genel akışla ilgili mesajlaşma işlemleri hep yönetici etmen üzerinden yapıldığı için yöneticiye fazla yüklenilmesi durumu söz konusu olmaktadır ve bu durum sistemin çalışmasını olumsuz yönde etkileyecek sonuçlar doğurmaktadır.

Şekil 5.1'de birinci versiyona dair tasarımın genel mimarisi verilmiştir.



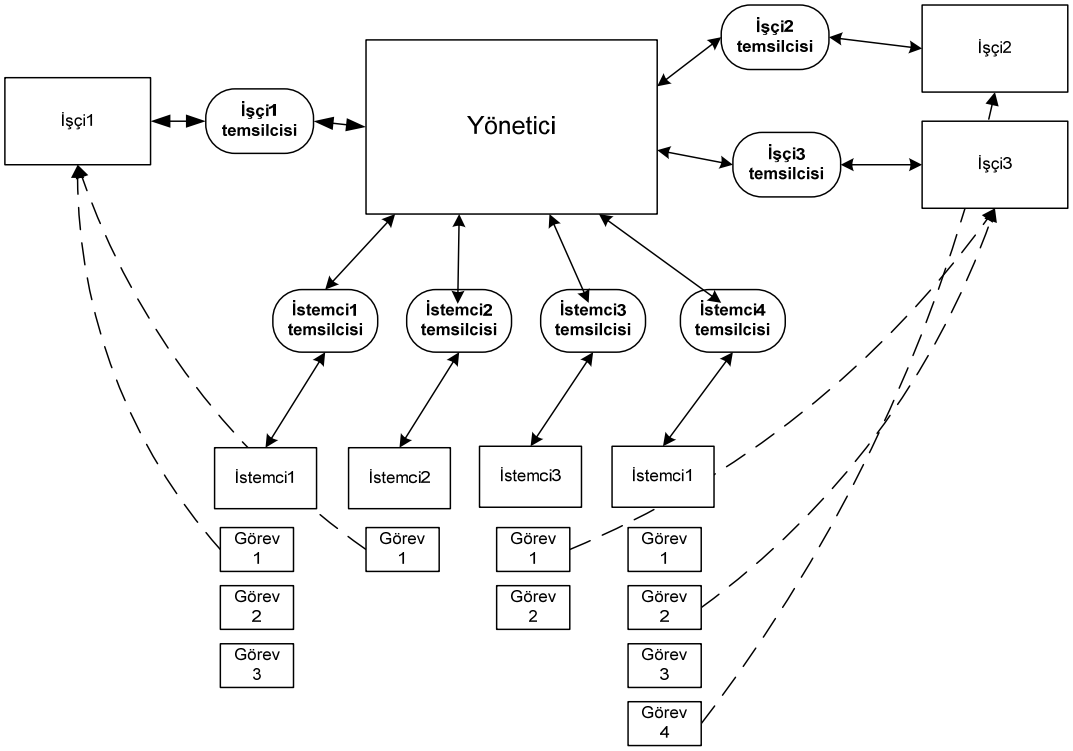
Şekil 5.1 : Birinci versiyonun genel mimarisi.

İkinci versiyonda ise birinci versiyonda oluşan aşırı yüklenme sorununu çözmek için her işçi ve istemci etmen için yönetici etmen bir temsilci etmen yaratır.

İşçi/istemci etmeler, yaşam döngülerinde uyguladıkları mesajlaşma protokollerini bu temsilci etmenler üzerinden yaparlar.

Aynı zamanda istemci etmenler işleri görev etmeni şeklinde yaratmaktadırlar. Normal bir yaşam döngüsüne sahip olan bu etmen, sisteme dahil olduktan sonra yürütülmek üzere taşınacağı hedef adresi bekler. İstemci etmen her bir görev için yönetici etmene talepte bulunur. Yönetici etmen sisteme bağlı işçi etmenler arasında uygun olan bir işçi etmeni tespit eder. İlgili görev etmenine işlemi yürütecek işçi etmenin adresini yollar ve işçi etmen kendisini uzaktaki bu hedef bilgisayara taşır. Taşınma işlemi bittikten sonra görev etmeni bu yeni konumunda görevi yürütmeye başlar.

Böylece etmenlerin taşınılabilirlik özelliğinin de sisteme eklenmesi hedeflenmiştir. Bir işçi makinesi iş durumuna göre birden fazla hizmeti aynı anda işletebilmektedir. Bu versiyon temsilci etmenler kullandığı için yönetici etmenin iş yükünü hafifletmektedir. Fakat hizmetlerin etmen olarak gerçeklenmesi ve işçi makinelere taşınması sistemin kararlılığı açısından sorun oluşturmaktadır. Herhangi bir sebepten ötürü işçi etmenin sistemden çıkması durumunda, bu etmenin yürütmekte olduğu işlerin kaybedilmesi durumu oluşmaktadır. Şekil 5.2'de bu versiyona ait tasarıma ilişkin diyagram verilmiştir.



Şekil 5.2 : İkinci versiyonun genel mimarisi.

Son versiyonda ise gene her bir işçi/istemci etmen için yönetici tarafından oluşturulan temsilci etmenle mesajlaşma protokollerini gerçeklemektedir. Aynı zamanda görev istemi ve atanması işleri de ilgili temsilci etmenin bağlamında halledildiğinden yönetici etmenin yük durumu dengelenmeye çalışılmıştır.

Bu versiyonda görevlerin etmen olarak gerçeklenmesine ve dolayısıyla taşınılabilirliğine son verilerek olası tutarsızlıklar ortadan kaldırılmıştır. Görevler normal bir sınıf şeklinde gerçeklenmişlerdir, hedef makineye görev atanması sırasında mesajın içeriğine eklenerek gönderilmektedirler. İşçi etmenler tek bir anda sadece bir istemciye ilişkin görevleri yerine getirmektedirler. Mesajlaşma alt yapısı tekrar yazılarak optimize edilen bu versiyonda etmenler arasındaki çeşitli protokoller güçlendirilerek sistemin kararlılığı ve güvenilirliği arttırılmıştır. Son versiyonun genel mimari yapısı Şekil 5.3'te verilmiştir.

5.3.3 Sistemin analizi

Etmen Sistemleri ve Gridler bölümünde de belirtildiği gibi çoklu etmen sistemleri ve grid sistemleri farklı tasarım kararları alınarak geliştirilmiş sistemlerdir. Bu çalışmada geliştirilen etmenli grid sisteminde her iki farklı bakışın melezi bir bakış açısı sağlanmıştır. Sisteme bağlanan istemcilerin bölümlendirip, sisteme verdikleri

işlerin uygun biçimde işletilip kendilerine bildirilmesi, etmenlerin yeterli ve gerekli esneklikle sisteme bağlanması ve sistemden ayrılması, hata durumlarının tespit edilip giderilmesi işlemlerine ilişkin yöntemler ve protokoller tanımlanmıştır. Yazılım çeşitli paketlere ayrıştırılarak her paketin belirli sınırlar içinde tanımlı işleri yerine getirmesine özen gösterilmiştir. Yazılım paketleri ve görevleri şu şekilde sıralanabilir.

- grid: Grid sisteminin yönetimsel sınıflarını içeren pakettir.
- client: İstemci etmenlere ilişkin sınıfları tutar.
- worker: İşçi etmenlere ilişkin sınıfları tutar.
- delegate: Temsilci etmenlere ilişkin sınıfların bulunduğu pakettir.
- message: Etmenler arası mesajlaşma işlemlerine ilişkin sınıfları içerir.
- task: İstemciler tarafından sisteme gönderilen ve işçi etmenlerce yürütülen görevlere ilişkin sınıflar bu pakettedir.
- property: Görevlerin ihtiyaç duydukları ve işçi etmenlerin sahip oldukları özellikleri gerçekleyen sınıfları barındırır.
- pooler: Delegelerin ilgili etmenin halen sisteme bağlı olup olmadığını test etmeleri gerekmektedir. Bu işlem için gereken protokol bu paketteki sınıflarca yürütülür.
- test: Test uygulamaları bu pakettedir.

Sistem analizi genel hatlarıyla yönetici sistemin analizi, etmenlerin yaratılması sisteme bağlanması/sistemden ayrılması prosedürünün ve görev atanması/görev sonuçlarının bildirilmesi protokollerinin analizi, mesaj iletişimiyle ilgili sistemin analizi olma üzere dört ana parçaya bölünmüştür. Bu bileşenler ve versiyonlar arasındaki iyileştirmeler izleyen alt bölümlerde verilmiştir.

Yönetici bileşenlerinin analizi

Gerçeklenen sistemde tüm akışın kontrolünü yapan bir yönetici etmen vardır. Klasik etmen sistemlerinde tüm sistemi kontrol eden merkezi bir yapı bulunmazken, [20] grid sistemlerinde akışı yöneten merkezi bir yapıya ihtiyaç vardır [3]. Tasarlamakta olduğumuz sistemde grid yapısının öne çıkıyor olması ve sisteme bağlı tüm işçi ve istemci etmenlerin uygun biçimde yönetilmesi gereğinden ötürü yönetici bir etmenin

gerekliliği açıktır. Tasarlanan sistemde bir ve yalnız bir tane yönetici etmen bulunmaktadır. Sunucu makinede grid sistemi yaratıldığında yönetici etmen de yaratılır. Yönetici etmeni görevleri şu şekilde sıralanabilir;

1. Sisteme bağlanan istemci ve işçi etmenlerin kayıtlarını tutmak ve onlar bağlandığında her biri için bir temsilci etmen yaratmak.

2. Etmenlerin sistemden güvenli biçimde ayrılmalarını sağlamak.

3. İşçi etmenlerin üzerlerindeki işleri tamamlamadan sistemden ayrılmaları durumunda ilgili görevin tekrar atanmasını sağlamak

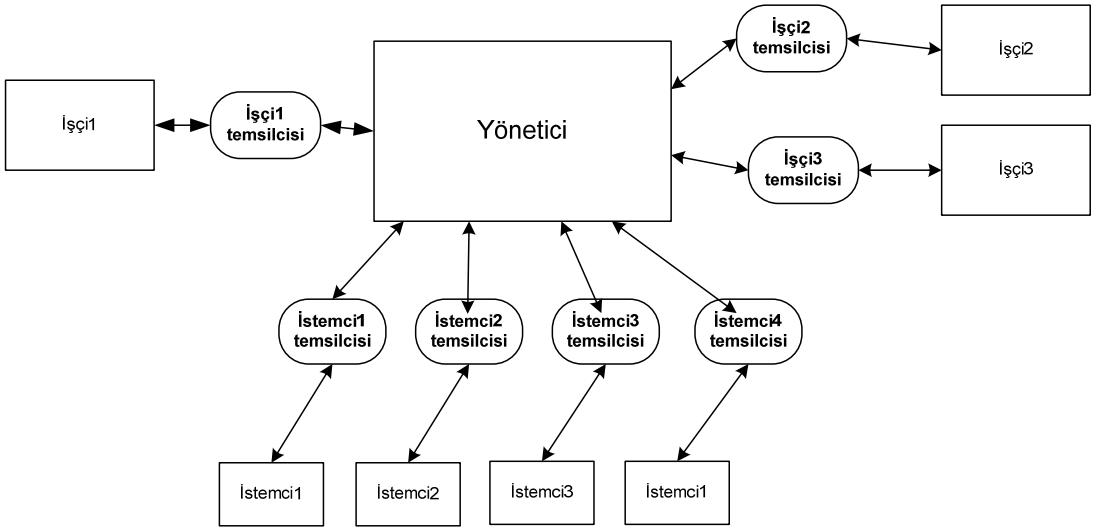
4. İstemci etmenlerin sistemden ayrılarak görevlerinin sonuçları istemedikleri durumunda ilgili işçi etmenlerin çalışmalarını sonlandırmak

Yönetici etmen bağlanan her bir etmen için (işçi/istemci) bir temsilci (delegate) etmen yaratır. Temsilci etmenler temsil ettikleri işçi ya da istemci etmenle olan tüm iletişimi ve protokol gerçeklemelerini yaparlar. Böylece yönetici etmenin üzerindeki yükü azaltmak hedeflenmiştir. Uygulamanın ilk versiyonunda bu temsilci yapısı kullanılmamıştı. Her katılımcı etmen yaşam döngüsü boyunca birçok kere yönetici etmenle mesaj alışverişinde bulunmak zorundadır. Özellikle sisteme bağlanan etmen sayısının artması durumunda yönetici etmenin yükünün çok arttığı ve buna bağlı olarak zaman zaman yakalanamayan mesajların olduğu gözlemlenmiştir. Bir grid sisteminin en önemli tasarım hedefinin güvenilirlik ve kararlılık olduğu daha önce belirtilmişti. Bu nedenle ikinci versiyondan sonra temsilci etmenler yapısı kullanılmıştır. Şekil 5.3'te bu yapıya genel bir bakışı verilmiştir.

Etmenlerin analizi

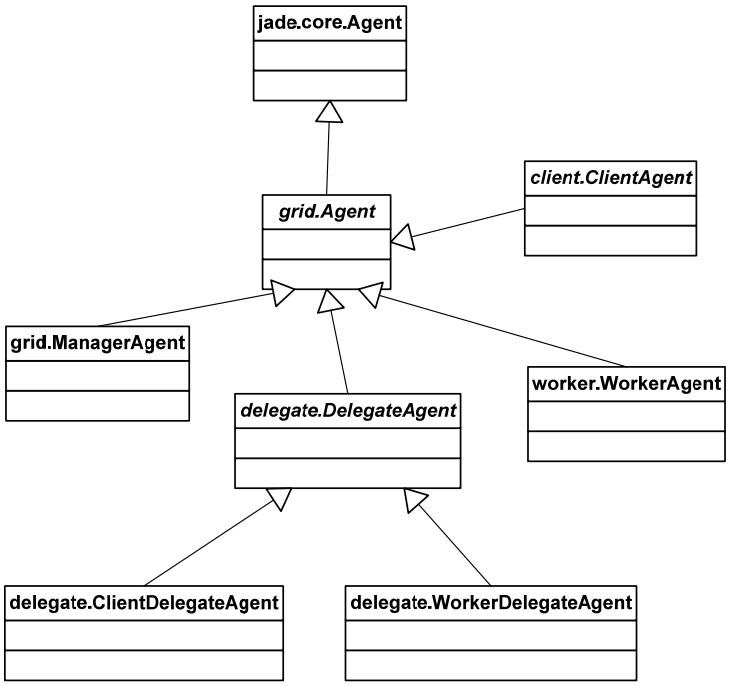
Şekil 5.3'te de görüldüğü gibi sistemde Yönetici, istemci, istemci temsilcisi, işçi ve işçi temsilcisi olmak üzere dört çeşit etmen bulunmaktadır. Yönetici etmenin görevi yukarıda anlatılmıştır.

Diğer etmenler hakkında ayrıntılı bilgi ise bu bölümde anlatılacaktır. Şekil 5.4 etmenler arasındaki hiyerarşik yapının genel bir şeklini göstermektedir.



Şekil 5.3 : Son versiyonun genel mimarisi.

jade.core.Agent sınıfı kullanıcıların yazdıkları tüm etmen sınıfları için ortak ana sınıftır [38]. Grid isimli pakette bulunan grid.Agent sınıfı da bu sınıftan türetilmiş ve grid sisteminde bulunan tüm etmenlerin ana sınıfı rolünü üstelenmiştir. Bu sınıfta bulunan yardımcı metotlar etmenlerin sisteme bağlanmasını ve sistemden ayrılmasını düzenler.



Şekil 5.4 : Etmenler arası hiyerarşik yapı.

Etmenler yaşam döngülerinde yapmaları gereken işleri ve uymak zorunda oldukları protokolleri gerçeklerken birçok yardımcı sınıfı kullanmaktadırlar. Bunlarla ilgili bilgiler ileriki bölümlerde verilecektir

İstemci Etmenler: İstemciler sistemden hizmet alan etmenlerdir. İstemci etmenler sistemin kendilerini tanıması için gereken bilgileri yönetici etmene yollayarak gride bağlandıktan sonra yürütmek istedikleri görevleri bildirirler. Grid sistemi kendi yaşam döngüsü içinde bu görevleri yürütüp sonuçlarını ilgili istemcilere bildirirler.

İstemci Temsilcileri: Daha önce de belirtildiği gibi her istemci için yönetici etmen tarafından bir temsilci etmen yaratılır. Bu temsilci etmen temsil ettiği istemci ile grid yönetimi arasındaki iletişimi koordine eder.

İşçi Etmenler: İşçi etmenler gönüllü olarak grid sistemine bağlanan ve sahip oldukları kaynakları gridin kullanımına sunan etmenlerdir. Önerilen sisteme yönelik yapılan çalışmada işçi etmenler işlemci gücüne dair kaynakları paylaşmaktadır. İşlemci gücünün yanında diğer hesaplama birimlerinin (grafik kartı v.b.) kaynakları da işçiler tarafından paylaşılabilinir. Ancak oluşturulan işçilerin depolama alanı gibi kaynaklarını paylaşımı çalışmanın kapsamının dışında tutulmuştur. İşçi etmenler sahip oldukları donanım-yazılım özelliklerine ve yeteneklerine uygun olarak kendilerine atanan görevleri işletip sonuçlarını ilgili etmenlere bildirirler. İşçi etmenler aynı zamanda çalıştıkları sistemin yük durumunu ölçerek sadece iş durumları elverişli olduğu durumlarda görev üstelenecek şekildedirler.

İşçi Temsilcileri: İstemci ve temsilcileri arasındaki ilişkiye benzer bir ilişki işçi etmenleri ve işçi temsilci etmenleri arasında da bir ilişki vardır. İşçi etmenleri grid yönetimiyle olan tüm iletişimi kendisi için yaratılıp sisteme dahil edilen temsilci etmenler üzerinden yapar.

Mesajlaşma yapısının analizi

Mesajlaşma dağıtık sistemlerin önemli noktalarından birisidir. Çünkü sistemin tüm koordinasyonu uzak noktalarda bulunan katılımcılar arasındaki mesajlaşmalara dayanmaktadır. Hızlı, kararlı ve güvenilir bir mesajlaşma alt yapısı önerilen sistem için anahtar bir tasarım kararı olarak ön plana çıkmıştır. Sistemde tanımlanan

protokollerin uygulanması için iyi tasarlanmış bir mesajlaşma alt yapısına gerek duyulmuştur.

Mesaj tipleri gerçeklenen protokollere sıkı biçimde bağlıdırlar. Bu nedenle sistemin versiyonları arasındaki değişimlerden en fazla etkilenen bileşenlerin başında mesajlaşma bileşeni gelmektedir.

Sistemin yaşam döngüsü boyunca en fazla çalışan modülü mesaj alışverişiyle ilgili kısımları olmuştur. Bu paketin sistem yükünü kaldıracak biçimde tasarlanması ve mesajların mesaj kuyruğundan akıllı biçimde seçilmesi çok büyük önem taşımaktadır. Uygulamanın ilk versiyonunda her etmen kendisine gelen mesajları belirli bir döngü içinde sürekli kontrol etmekteydi. Bu yöntemin bazı sorunlar doğurduğu görülmüştür. Döngünün kontrol frekansının çok büyük olması sistemin performansına olumsuz yönde etki etmekteydi, döngü frekansının küçültülmesi durumunda ise bazı mesajların iletiminde sorunlar doğurmakta ve bazı mesajların iletilmeden sistemden kaldırılmasına neden olmaktaydı. Bundan ötürü ikinci versiyonda mesajlaşma yapısı gözden geçirilmiş ve yoğun mesajlaşma iletişimini sağlayacak şekle getirilmiştir. Tüm mesajlaşma yapısında FIPA standartlarıyla tam uyumlu olan JADE ACL modülü kullanılmıştır.

Mesaj Tipleri

Sistemde kullanılan mesaj tipleri ve kullanım amaçları şu şekildedir.

1. CLIENT_REGISTER: İstemci etmenin sisteme bağlandığını belirtir.
2. CLIENT_ACCEPTED: İstemci etmenin grid tarafından kabul edildiğini ve çalışmaya hazır olduğunu bildirir.
3. CLIENT_INFO: İstemci etmen sisteme bağlandıktan sonra kendisi hakkındaki bilgileri bu mesajla yollar.
4. CLIENT_RECONNECT: Sistemden ayrılan istemcinin tekrar sisteme bağlandığını bildirmek için kullanılır
5. CLIENT_UNREGISTER: İstemci etmenin artık sistemden ayrıldığın bildirir.
6. CLIENT_DONE: İstemci etmenin sistemden tamamen ayrıldığını ve temsilcisinin de sistemden çıkartılabileceğini bildirir.
7. WORKER_REGISTER: İşçi etmenin sisteme bağlandığını bildirir.

8. WORKER_ACCEPTED: İşçi etmenin grid tarafından kabul edildiğini ve çalışmaya hazır olduğunu bildirir.
9. WORKER_INFO: İşçi etmen sisteme bağlandıktan sonra kendisi hakkındaki bilgileri bu mesajla yollar.
10. STOP_WORKER: İşçi etmene elindeki işi sonlandırması gerektiğini bildirir.
11. WORKER_STOPPED: İşçi etmenin çalışmasının sonlandığını bildirir.
12. WORKER_UNREGISTER: İşçi etmenin artık sistemden ayrıldığın bildirir.
13. WORKER_FREE: İşçi etmenin çalıştığı platformun uygun durumda olduğunu bildirir.
14. WORKER_BUSY: İşçi etmenin çalıştığı platformun meşgul durumda olduğunu bildirir.
15. TASK_REQUEST: İstemci tarafından gönderilen göreve ilişkin mesajdır.
16. TASK_REQUEST_FAILED: İstemci tarafından gönderilen görevin işçi etmenlere atanmasında ve yürütülmesinde bir hata olduğunu bildirir.
17. TASK_REQUEST_ACCEPTED: İstemci tarafından gönderilen görevin uygun bir işçi etmen tarafından kabul edildiğini ve yürütülmeye başlandığını bildirir.
18. TASK_COMPLETED: Görevin başarıyla tamamlandığını belirtir.
19. I_AM_ALIVE: İşçi ve istemci etmenler belirli aralıklarla kendi temsilcilerine hala hayatta oldukları ve grid sistemine bağlı olduklarını bu mesaj tipiyle gönderirler.

Mesaj gönderme ve alma işlemleri

JADE ile mesaj gönderimi jade.lang.acl.ACLMessage sınıfı ile yapılmaktadır. Bu sınıfın FIPA standartlarına uygun bazı nitelikleri vardır [23, 38]. Etmenler aşağıda tanımları verilen bu nitelikleri göndermek istedikleri mesaj türüne göre belirleyip göndermektedirler.

- Edimsel (Performative): Mesajın türüne dair bilgi içerir. Yukarıda tanıtılan mesaj tiplerinin her biri kendisi için bir edimsel tanımlamışlardır.
- Ontoloji (Ontology): Mesajın içeriğine dair bilgi içerir. Sistemde kullanılan mesaj tiplerinin her biri kendileri için bir ontoloji tanımlamışlardır.

- İçerik (Content): Mesajın içeriğidir. FIPA standartlarına göre mesajların içerikleri metin tabanlıdır. Ancak JADE Java dilinin bir yapısı olan serileştirilebilir (Serializable) nesneleri de mesaj içeriği olarak kullanabilmektedir.

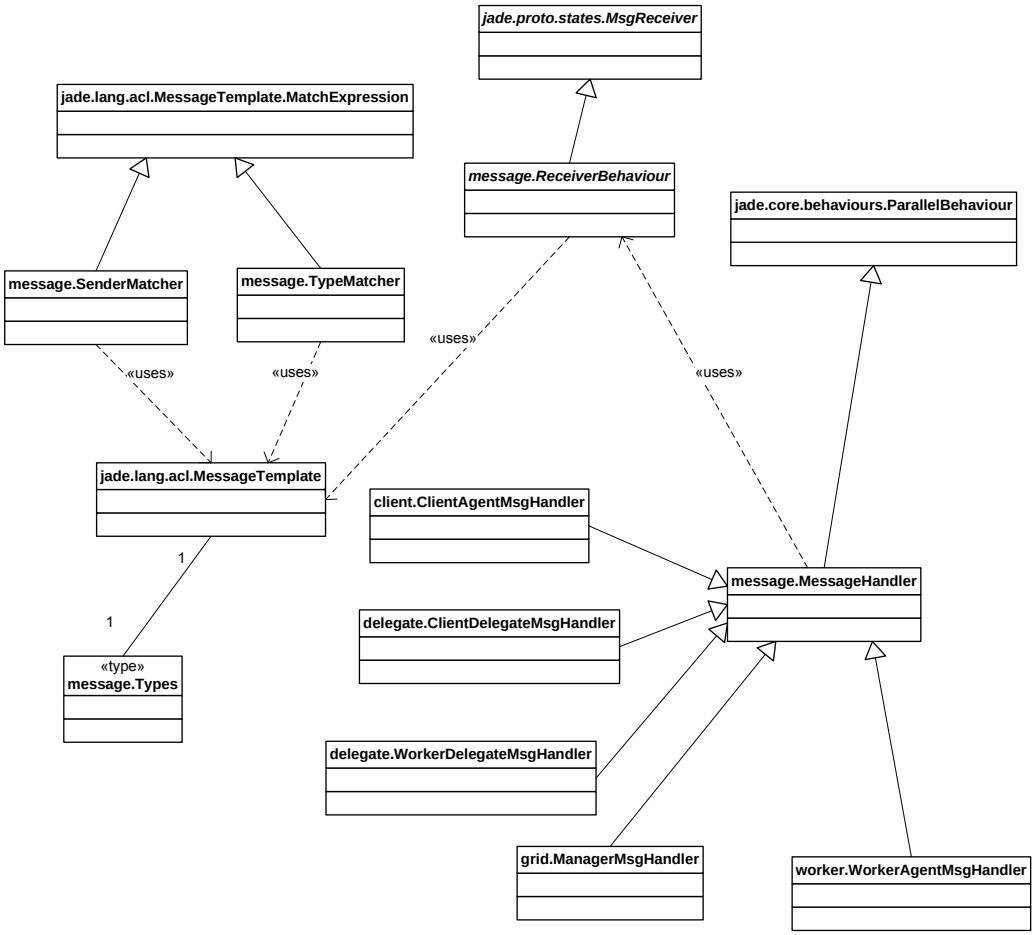
Daha önce de tartışıldığı gibi etmenlerden sisteme çok fazla yük olmadan kendilerine gönderilen mesajı kolayca ve hızlıca algılayıp gerekeni yapması beklenmektedir. Bu işlem için JADE tarafından sunulan jade.lang.acl.MessageTemplate ve jade.lang.acl.MessageTemplate.MatchExpression sınıfları kullanılmıştır. MessageTemplate sınıfı mesaj türleri için belirli bir kalıp belirtmektedir. MatchExpression sınıfı ise mesajın belirtilen kalıba uygun olup olmadığını anlamakta kullanılmaktadır. Bunların yanında kullanılan jade.proto.states.MsgReceiver sınıfı belirli bir mesaj kalıbında uygun mesaj alınması durumunda sistem tarafından otomatikman çalışan yapıları oluşturmaktadır.

Efektif bir mesaj alt yapısı oluşturmak için JADE tarafından sunulan bu yapılar kullanılmıştır. Buna göre verilen tasarım kararı şu şekilde verilmiştir.

- Mesajın tipine göre seçilebilmesi için MatchExpression sınıfından türetilmiş bir sınıf oluşturulacaktır (TypeMatcher)
- Mesajın gönderenine göre seçilebilmesi için MatchExpression sınıfından türetilmiş bir sınıf oluşturulacaktır (SenderMatcher)
- Mesaj alımı işlerini MsgReceiver sınıfından türetilmiş bir sınıf üstlenecektir (ReceiverBehaviour). Bu sınıf MessageTemplate türünden sınıfları belirli kalıplara göre mesaj bekleme işini yapmaktadır.
- Farklı kalıplara göre mesaj bekleme işinin paralel yürütülebilmesi için bir sınıf yaratılacaktır (MessageHandler).
- Etmenler kendi gerçeklemelerine göre MessageHandler sınıfından türetecekleri yeni sınıflarla ve bu sınıfa ekleyecekleri ReceiverBehaviour türünden sınıflarla mesaj işleme işlerini yapacaklardır.

Şekil 5.5'de mesajlaşma yapısını oluşturan sınıflar arasındaki ilişki görülmektedir.

Önerilen bu mesajlaşma alt yapısıyla etmenlerin sistemi meşgul durumda bekletmeden (busy waiting) kendilerine iletilen mesajı seçerek alması ve işlemesi sağlanmıştır.



Şekil 5.5 : Mesajlaşmada kullanılan sınıflar.

Görevlerin analizi

Grid sisteminde görevler ve özelliklerinin esnek biçimde gerçeklenmesi gerekmektedir.

Önerilen sistemde görevlerin gerçeklemeye ve uygulamaya göre özelleştirilebilmesi sağlanmıştır. Bazı görevlerin yerine getirilmesi için özel donanımlar (Ekran kartı, belirli bir giriş/çıkış kartı) ya da hazır yazılımların bulunması gerekebilir Yönetici etmen görevi yönetecek işçi etmeni seçerken bu ekstra özelliklere de dikkate almak zorundadır.

Görevler soyut task.Task sınıfıyla gerçeklenmiştir. Java dilinin yetenekleri içinde her amaca yönelik görev tanımlamak mümkündür. task.Task sınıfından türetilen sınıflar soyut olan *execute* yordamını yapmak istedikleri işe göre gerçeklerler. Görevlerin istedikleri özellikler de property.Property ve property.PropertyList sınıflılarıyla gerçeklenmiştir. Görevler yürütülmek için ihtiyaçları olan özellikleri (Property) bir liste halinde (PropertyList) tutarlar. İşçi etmenlerde sahip oldukları özellikleri aynı

yolla tutarlar. Bir görev bu şekilde belirli özelliklere ihtiyaç duyuyorsa soyut olan *properties* yordamını gerçekleyerek bu özellikleri belirtir. Yönetici etmen görev atamasında işçi ve görevin özelliklerinin uyumlu olmasına dikkat eder. İşçi etmen bir görevi aldığında bu görevin *execute* yordamını çağırarak görevin yerine getirilmesini sağlar.

5.3.4 Protokoller

Grid sisteminin genelinin ve her bir etmenin yaşam döngüsü içinde gerçekleştirdiği eylemler belirli protokollere bağlı kalmak zorundadırlar. Bu protokoller hangi işin hangi sırayla ve nasıl gerçekleneceğini açıklar. Bu bölümde bu protokoller ayrıntılı biçimde tanıtılacak ve gerek analizi gerekse gerçeklenmesi sürecinde oluşan problemler ve bu problemlerin nasıl çözüldüğü anlatılacaktır.

Etmenlerin sisteme bağlanma protokolü

Etmenler JADE ana çatısına bağlandıklarında kendilerine bir kimlik numarası (AID) tahsis edilir ve böylece sistemde çalışmaya başlarlar. Ancak etmenlerin JADE sistemine dâhil olması önerdiğimiz grid sistemi için yeterli değildir. Yönetici etmenin bağlanan etmenin türü ve özelikleri hakkında bilgi sahibi olması gerekmektedir. Aynı zamanda yönetici etmen bağlanan etmenin gride dâhil edilip edilmeyeceğine de karar vermelidir (örneğin belirli güvenlik ve kimlik doğrulama yöntemleriyle). Böyle bir karar yapısı bu tezin konusu dışında kalmakla beraber sistemin ileride böyle desteklere sahip olabilmesi için protokol esnek tasarlanmıştır.

Sisteme bağlanabilecek iki tür etmen vardır; istemci ve işçi etmenler. Yönetici etmen grid sistemi oluşturulduğunda sistemin bir parçası olarak oluşturulur. Temsilci etmenler ise sisteme bağlanan her bir işçi ya da istemci etmen için yönetici etmen tarafından yaratılarak sisteme dâhil edilirler ve dolayısıyla söz konusu bağlanma protokolüne tabii değildirler. İşçi ve istemci etmenlerin sisteme bağlanma prosedüründe ufak farklılıklar bulunmaktadır. Bu nedenle bu iki prosedür çok benzer olmalarına karşın ayrı ayrı ele alınacaktır.

İstemci etmenin sisteme bağlanma protokolü

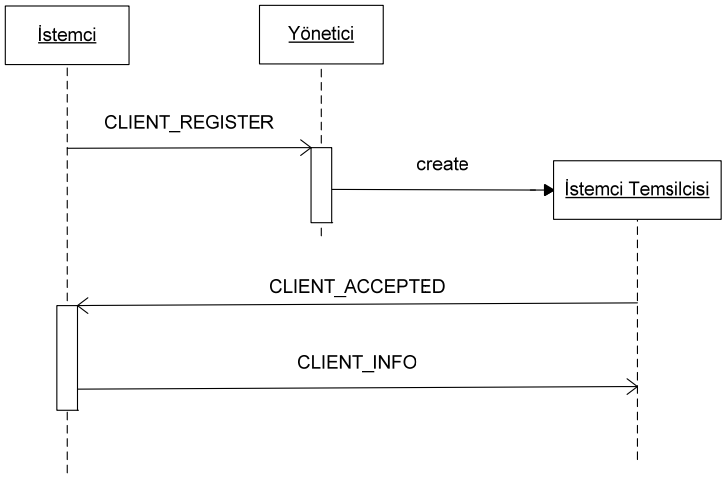
İstemci etmen sisteme bağlanıp sistemden çıkabilir ve bir süre sonra tekrar bağlanma talebi gönderebilir. Böylece istemci etmenin yürütülmek üzere sisteme görevleri göndermesi, tüm görevleri gönderdikten sonra sistemden ayrılması ve bir süre sonra

sisteme yeniden bağlanarak tamamlanan görevlerin sonuçlarını alabilmesi sağlanmıştır. Yeniden bağlanma protokolü ayrıca açıklanacaktır

Bağlanma protokolün işleyişi şu şekildedir.

1. İstemci etmen sisteme bağlanma isteğini yönetici etmene CLIENT_REGISTER mesajı yollayarak bildirir.
2. Yönetici etmen bu mesajı aldıktan sonra bu istemcinin sisteme daha önce dâhil olup olmadığını araştırır. Eğer bu istemci sisteme ilk defa dâhil oluyorsa istemci için bir temsilci etmen yaratır ve istemci ve temsilcisinin kayıtlarını tutar.
3. Yaratılan bu temsilci istemci etmene CLIENT_ACCEPTED mesajı yollar.
4. İstemci bu mesajı alınca temsilcisine kendisine ilişkin bilgileri içerik olarak barındıran CLIENT_INFO türünden bir mesaj yollar. Böylece istemci ve temsilcisi eşleştirilmiş olur ve istemci etmenin grid içindeki yaşam döngüsü başlar.

Şekil 5. 6'da istemci etmenin sisteme bağlanışı sırasında oluşan mesajlaşma sırasını ve aktörlerini görebilirsiniz. Bu şekilde de görüldüğü gibi istemci etmen son aşamada kendisine ait bilgileri CLIENT_INFO mesajıyla temsilcisine bildirir. Bu mesaj türünün içeriği client.ClientInfo türünden bir nesnedir İstemci kendisi hakkındaki bilgileri ve yürütülmesini istediği görevleri bu mesajla temsilcisine gönderir.



Şekil 5.6 : İstemci etmenin sisteme bağlanma prosedürü

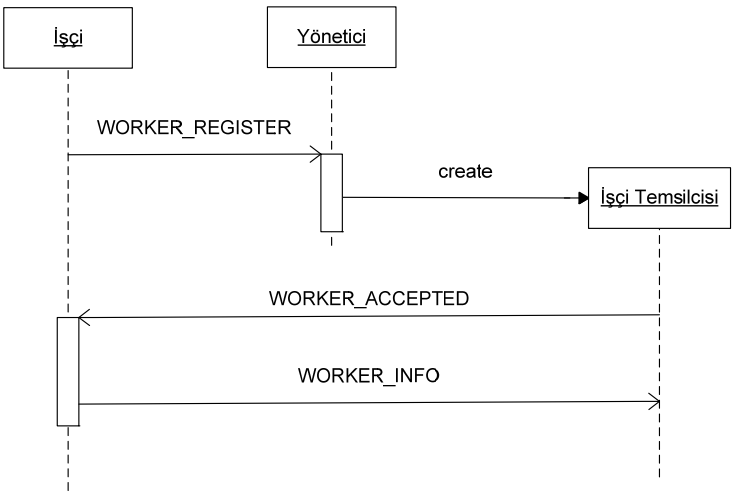
Gönderilen mesajın içeriği istemciye ait ClientInfo türünden bir mesajdır. ClientInfo sınıfı java.io.Serializable sınıfından türetildiği için JADE ile gönderilen mesajların içeriği olarak kullanılabilmektedir. Böylece temsilci yönetici etmenle çalışarak bu işleri uygun işçilere gönderir. Bu mesajla aynı zamanda istemci etmen isterse zaman aşımı bilgisi de gönderir. Temsilci etmen görevlerin belirtilen zaman aşımı içinde yapılmazsa istemciyi griden çıkartması sağlanır. Görev atanmasına ilişkin detaylı bilgi ilgili kısımda tekrar anlatılacaktır.

İşçi etmenin sisteme bağlanma protokolü

İşçi etmenler istemci etmenlere benzer bir protokolle bağlanırlar. Ancak farklı olarak işçi etmenlerin sisteme tekrar bağlanma durumu söz konusu olmamasından ötürü yönetici etmen her bağlanan işçi etmen için bir temsilci etmen yaratır.

Protokolün akışı şu şekildedir;

1. İşçi etmen sisteme bağlanma isteğini yönetici etmene WORKER_REGISTER mesajı yollayarak bildirir.
2. Yönetici etmen bu mesajı aldıktan sonra bir işçi temsilci etmen yaratır ve işçi ve temsilcisinin kayıtlarını tutar. Yaratılan bu temsilci işçi etmene WORKER_ACCEPTED mesajı yollar.
3. İşçi etmen bu mesajı alınca temsilcisine kendisine ilişkin bilgileri içerik olarak barındıran WORKER_INFO türünden bir mesaj yollar. Böylece işçi ve temsilcisi eşleştirilmiş olur.



Şekil 5.7 : İşçi etmenin sisteme bağlanma prosedürü.

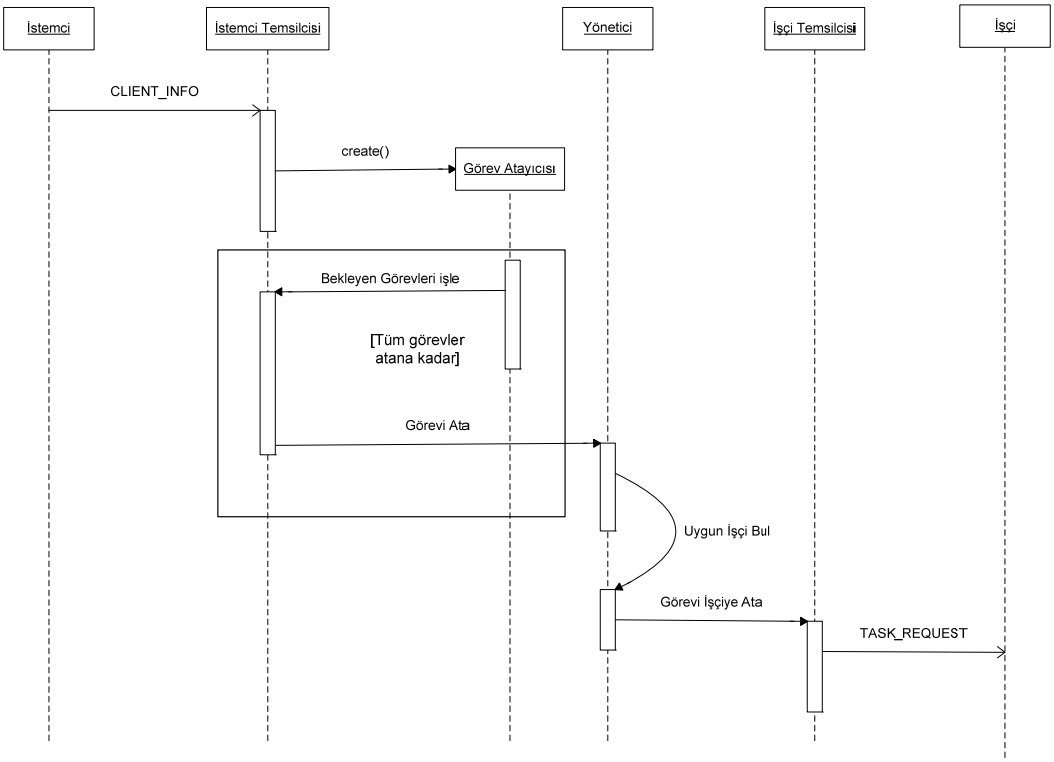
Şekil 5.7'de işçi etmenin sisteme bağlanışı sırasında oluşan mesajlaşma sırasını ve aktörlerini görülebilir. İşçi etmenin yolladığı WORKER_INFO mesajının içeriği worker.WorkerInfo türünden bir nesnedir. Bu mesajla işçi etmen temsilcisine sahip olduğu nitelikleri bildirir. Böylece işçi etmene görev atanması esnasında görevin gerektirdiği özelliklere sahip olan işçi etmenin seçilmesi sağlanır.

Görev atama protokolü

Grid sistemlerinde görevlerin uygun biçimde atanması ve olası hataların uygun bir şekilde giderilmesi hayatı bir önem taşımaktadır. Önerilen sistemde uygulana protokol şu şekildedir.

1. Etmenler sisteme bağlanma protokolünde CLIENT_INFO mesajının içeriği olarak gönderdiği bilgilerden birisi de yürütülmesini istediği görev bilgileridir.
2. Temsilci etmen istemciye ilişkin bilgileri (ClientInfo) aldığında istemci tarafından gönderilen işleri de alır. İstemci temsilcisinde bir işçiye atanmayı bekleyen ve bir işçi tarafından yürütülmesine devam edilen işler listesi olmak üzere iki ayrı liste bulunur. Temsilci tüm görevleri bekleyen görevler listesine ekler.
3. Temsilci etmen belirli aralıklarla bekleyen işler listesini kontrol edip yönetici etmene görevin atanması için istekte bulunur. Belirli aralıklarla belirli işleri yinelemek için JADE tarafından sunulan jade.core.behaviours.TickerBehaviour sınıfı kullanılır [38]. Görevlerin kontrolü için bu sınıftan türetilmiş delegate.TaskScheduler sınıfı kullanılmıştır
4. Yönetici etmen sahip olduğu etmenler arasında meşgul olmayan ve özellikleri bu göreve uygun işçi etmenleri bulur.
5. Uygun bir işçi etmen varsa bu işçinin temsilci etmeni görevi TASK_REQUEST mesajıyla bu işçi etmene iletir.
6. İşçi etmen bu mesajı alınca görevi yürütmeye başlar.
7. İşçi temsilcisi görevin atandığını yönetici etmene bildirir
8. Yönetici etmen de görevin atandığını ilgili istemci temsilcisine bildirir.

9. İstemci temsilcisi atanan görevi bekleyen görevler listesinden çıkartıp çalıştırılan görevler listesine ekler.



Şekil 5.8 : Görev atama prosedürü.

Görevlerin atanması protokolünde rol alan aktörleri ve mesajların akış sırası Şekil 5.8'de verilmiştir.

İşçi etmenleri yük durumunun ölçülmesi

Görev atama sürecinin önemli noktalarından birisi de işçi etmenlerin üzerindeki iş yükünün ölçülmesi ve çok yoğun işçi etmenlerine görev atanmamasıdır.

Bunun için uzak makinede bulunan her bir etmen belirli aralıklarla bulunduğu bilgisayarın işlemci kullanım oranını test eder. Bu oran belirli bir değerin üzerine çıkarsa (uygulamada %90 olarak belirlenmiştir) işçi etmen durumunu meşgul olarak belirtirken yük durumu bu oranın altında olursa işçi etmen uygun duruma geçer. Sisteme bağlanan her işçi etmenin varsayılan durumu meşgul olarak belirlenmiştir. Daha sonra gerçek yükü ölçülerek uygun duruma geçirilmesi sağlanmıştır.

Önerilen sistemin ilk iki versiyonunda yönetici etmen işçi etmenler ilişkin iki adet liste tutuyordu. Birinci listede uygun olan ikinci listede de meşgul olan etmenlerin kaydı vardı. İşçi etmenler temsilcileri aracılığıyla yönetici etmene durumlarını

bildirdiklerinde yönetici bu listelerde uygun düzenlemeyi yapıp, görev atanması sırasında yönetici etmen sadece uygun listesinde bulunan etmenler üzerinde arama yapıyordu. Ancak bu yapı çok fazla senkronizasyon sorunu doğurmakta idi. Ayrıca görev atanma prosedürünü oldukça karmaşıklaştırmakta idi. Son versiyonda bu bilgi her işçi etmene karşılık oluşturulan işçi temsilcisi tarafından tutulmaktadır. Böylece temsilci işçinin meşgul olup olmadığına karar verebilmektedir. Yönetici etmen işçi etmenlerinin değil işçi temsilcilerinin bir listesini tutmaktadır. İşçi temsilcileri ise ilgili işçiye ait tüm bilgileri (yük durumu, adresi-konumu, özellikleri v.b.) tutar. Görev atama sırasında yönetici etmen listesindeki her bir işçi temsilcisine temsil ettiği etmenin söz konusu görevi yerine getirip getiremeyeceğini sorar. Temsilci etmen işçinin yük durumuna ve görevin istediği özelliklerle işçinin sahip olduğu özelliklerin uyumuna bakarak yöneticiye cevap verir. Böylece görev için uygun işçi bulunur.

Yük durumunun ölçülmesinde worker.SystemAnalyzer sınıfı kullanılmıştır. jade.core.behaviours.TickerBehaviour sınıfından türetilen bu sınıf java.lang.management.OperatingSystemMXBean sınıfın kullanarak Java dilinin özelliklerini kullanarak belirli aralıklarla sistemin yükünü ölçmektedir.

Bu ölçümler sonucu işçi etmen koştuğu bilgisayarın çok meşgul olduğu sonucuna varırsa temsilcisine WORKER_BUSY mesajını, aksi yönde bir kanaat getirirse de WORKER_FREE mesajını yollar. İşçi ve temsilcisi arasındaki mesaj trafiğini çok fazla arttırmamak için sadece durum değişimi olursa mesaj gönderilmiştir.

Görev yürütülmesi ve sonuçlarının bildirilmesi

İşçi etmen görev atama protokolüne göre yürütmesi gereken görevi aldıktan sonra bu işi yerine getirmek için çalışmaya başlar. Önerilen sistemde bir işçi etmen bir anda sadece bir görevi çalıştırmakla yükümlüdür. Uygulamanın ikinci versiyonunda bir işçi etmen farklı farklı iş parçaları (Thread) yaratarak bir anda birden fazla görevi paralel olarak yönetebilecek şekilde tasarlanmıştı. Bu yapıdan son versiyonda vazgeçilmiştir. Bu yapı her işçi etmenin birden fazla iş parçası yaratmasından ötürü bu iş parçaları arasında senkronizasyonu problemleri çıkartmaktaydı. Aynı zamanda görev atama sırasında dengesizlik olmakta idi. Bir işçi birden fazla görevi aynı anda yürütebildiğinden bazı durumlarda bazı işçilerin yükü çok fazla iken bazı işçilerin boşta beklemeleri söz konusu olabiliyordu. Bununla beraber bir işçi etmenin bir hatadan ötürü veya başka bir nedenle sistemden çıkması durumunda, birden fazla

görev ve dolayısıyla birden fazla istemci etmenin durumdan etkilenmesi sorunu doğmakta idi Bu sorunlar iş atama algoritmasında yük dengeleme kullanarak, görevlerin işçiler arasında mümkün olduğunca dengeli bir biçimde dağıtılmasıyla giderilmiştir. Ancak bu durumda iş dağıtma algoritması karmaşıklaşmış ve esnekliğini yitirir duruma gelmiştir. Bu nedenle önerilen son versiyonda her işçi bir anda bir tek görevi yerine getirecek şekilde tasarlanmıştır.

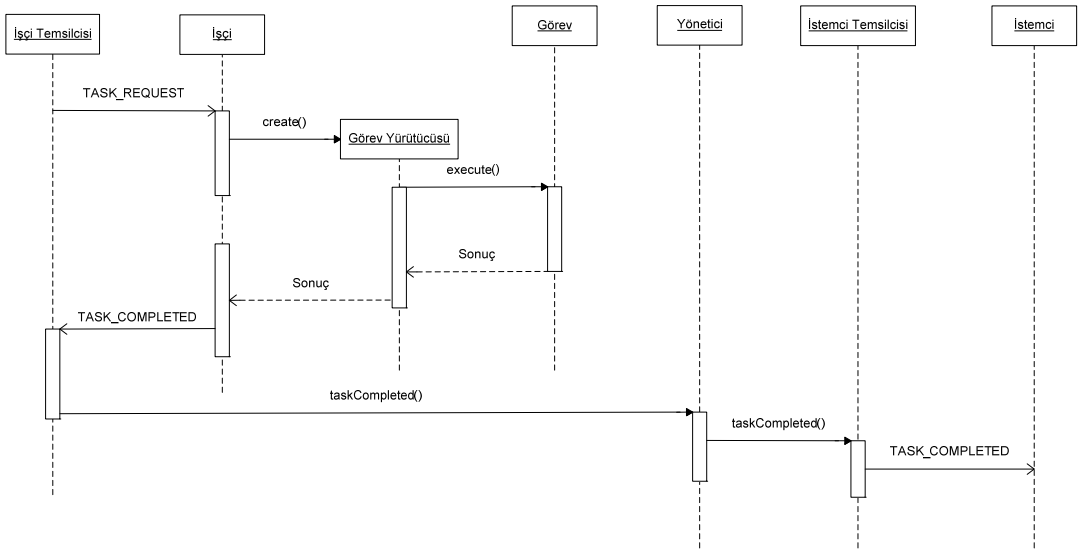
Görevin yürütülmesine dair protokol şu şekildedir.

1. İşçi delegesi görevi TASK_REQUEST mesajı ile işçi etmenine bildirir.
2. İşçi etmeni bu mesajı alınca TASK_REQUEST_ACCEPTED mesajını cevap olarak delegesine yollar.
3. Daha sonra işçi etmen görevi yürütecek iş parçasını (Thread) oluşturacak olan worker.TaskRunner sınıfından bir nesne yaratır.
4. TaskRunner nesnesi görevin *execute* yordamını yeni oluşturduğu bir iş parçasında çağırarak görevin işlevini yerine getirir.
5. Görevin yürütülmesi bitince TaskRunner nesnesi sonucu işçi etmene bildirir.
6. İşçi etmen sonucu TASK_COMPLETED mesajıyla işçi temsilcisine bildirir
7. İşçi temsilcisi görevin sonucunu yönetici etmene bildirir.
8. Yönetici etmen bu göreve ait istemci temsilcisini listesinden bulur ve görevin sonucunu istemci temsilcisine bildirir.
9. İstemci temsilcisi sonucu TASK_COMPLETED mesajıyla istemciye bildirir. İlgili görevi çalışan görevler listesinde çıkartır.

Şekil 5.9'da bu protokolün akışı görülebilir.

Görev yürütülmesinde oluşabilecek hatalar

Görevin işçi etmen tarafından yürütülmesi sırasında herhangi bir sebepten ötürü işçi etmen sistemden ayrılabilir. Etmenlerin sistemden ayrılmaları durumu temsilcileri tarafından sezilerek ilgili işlemler yapılır. Bu konu etmenlerin sistemden ayrılmaları bölümünde tartışılacaktır. İşçinin sistemden ayrılması durumunda yürütülen görevin tekrardan sisteme dâhil edilip görev atama protokolüne tabii tutulması için bir takım işler yapılmalıdır. Benzer bir durum görev atanmasında oluşacak hatalardan da doğabilir.



Şekil 5.9 : Görevlerin yürütülmesi prosedürü.

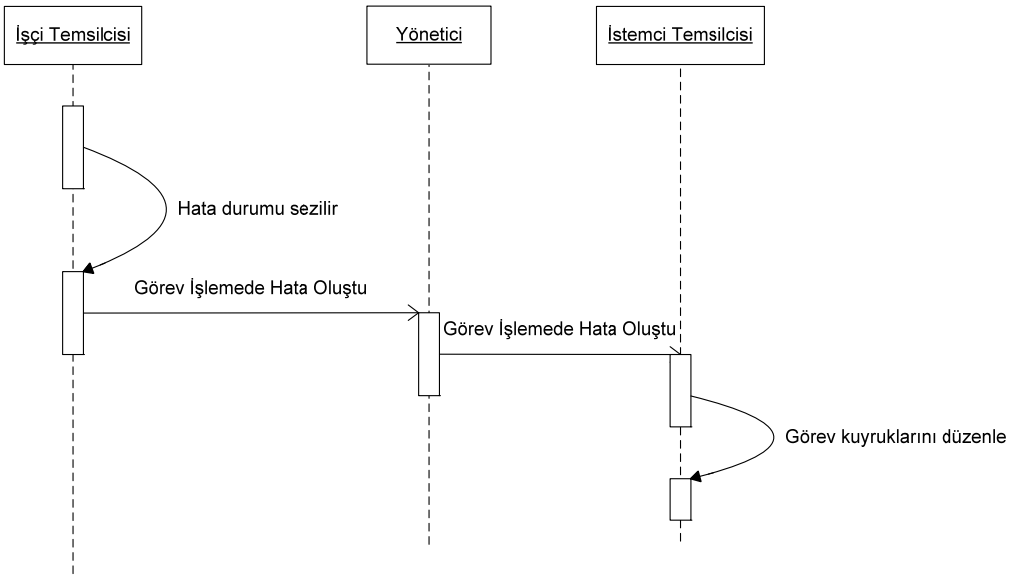
Görev atanması protokolünde bazı durumlarda işçi etmenlerin TASK_REQUEST mesajını almalarında ya da bu mesaja cevap olarak gönderdikleri TASK_REQUEST_ACCEPTED mesajının temsilci etmen tarafından alınması durumunda çeşitli sebeplerden ötürü (ağ da gecikme ya da hata v.b.) sorunlar olabilir. Görevin atanmasında bu durumun sorun teşkil etmesinin önüne geçilmesi gerekmektedir. Bu yüzden görev atama protokolün bu kısmına hata sezme prosedürü eklenmiştir. Bu prosedüre göre işçi temsilcisi TASK_REQUEST mesajını yolladıktan sonra karşılığında bir süre gelecek TASK_REQUEST_ACCEPTED mesajını bekler, şu andaki gerçeklemede temsilci bir dakika boyunca bu mesajın gelmesini bekler. Eğer bu bekleme süresince mesaj gelmezse görev atama işleminde hata olduğuna kanaat getirir. Böyle bir durumda da ilgili görevin istemci temsilci tarafından tekrar beklenen işler kuyruğuna atılması gerekmektedir.

Bu iki hatadan birisinin oluşması durumunda görevin tekrar işlenmek üzere bekleyenler görevler listesine atanması işlemine dair protokol şu şekildedir;

1. İşçi temsilcisi hata durumu sezer.
    a. İşçi temsilci etmeni belirtilen sürede TASK_REQUEST_ACCEPTED mesajını alamaz ve hata olduğunu algılar.
    b. İşçi temsilcisi işçi etmenin sistemden çıktığını anlar.

2. İşçi temsilcisi söz konusu görevin atanmasında hata olduğunu yönetici etmene bildirir.
3. Yönetici etmen bu göreve ait istemci temsilcisini listesinden bulur ve görevin atanmasında bir hata olduğunu istemci temsilcisine bildirir.
4. İstemci temsilci etmeni bu görevi çalışan görevler kuyruğundan çıkartıp bekleyen görevler kuyruğuna koyar. Böylece bir sonraki görev atama protokolünde görev tekrar atanmaya çalışılacaktır.

Şekil 5. 10 bu protokole ilişkin akışı göstermektedir.



Şekil 5.10 : Görevlerin yürütülmesindeki hataların sezilmesi.

Etmenlerin sistemden ayrılma protokolleri

Önerilen grid sisteminde yönetici etmen diğer tüm etmenlerin kaydını tutmakta, yaşam döngülerini gözlemekte kontrol etmektedir. Hatta ilgili temsilci etmenleri de yaratıp sisteme dâhil etmekle de görevlidir. Yönetici etmenin bu kontrolü sağlayabilmesi için etmenlerin sistemden ayrıldıklarını kesin bir şekilde sezebilmesi gerekmektedir. Sistemden ayrıldığı sezilemeyen etmenlerin olmasının sistemin kararlılığı ve güvenilirliğini negatif yönde etkileyeceği açıktır. Bazı görevlerin yerine getirilememesi ya da olmayan işçilere görev atanması ve buna bağlı olarak birçok hata düzeltme protokollerinin gereksiz yere tekrar tekrar çalıştırılması söz konusu olabilir.

Sistemden ayrılabilen etmenler sadece istemci ve işçi etmenlerdir. Temsilci etmenler sadece yönetici etmen tarafından yaratılıp sisteme dâhil edilir ve sadece onun tarafından sistemden çıkartılırlar.

Uygulamanın versiyonları arasında etmenlerin sistemden ayrılmalarına ilişkin farklı öneriler sunulup gerçeklenmiştir. Bu önerilen şu şekilde sıralanabilir;

- Birinci versiyonda etmenler sistemden ayrılacakları durumda yöneticiye isteklerinin bildirir bir mesaj yollar ve yönetici de ayrılma prosedürünü işletirdi. Ancak bu bakış açısı büyük sorunlar doğurmaktadır. Çünkü bazı durumlarda etmenler kendi iradeleri dışında sistemden ayrılmak durumunda kalmaktadırlar. Örneğin etmenin çalıştığı uzak makinedeki olası bir sorun, elektrik kesintisi ya da bağlantıda oluşabilecek bir sorun etmenin söz konusu mesajı gönderemeden sistemden ayrılmasına sebep olabilmektedir.
- İkinci versiyonda ise temsilcilerle etmenler arasında oluşan karşılıklı bir mesaj alışverişi sonucunda etmenin sistemden ayrılıp ayrılmadığına karar verilmiştir. Bu yöntemde temsilciler belirli aralıklarla gönderdikleri mesajlarla etmenlere hala sisteme bağlı olup olmadıklarını sormaktadırlar. Etmenler sisteme bağlı iseler bu mesaja karşılık olarak sistemde bağlı olduklarını bildirir cevap mesajı yollamaktadırlar. Temsilci belirli bir süre boyunca mesajlarına karşılık alamazsa temsil ettiği etmenin sistemden ayrıldığı sonucuna karar verip ilgili protokolü işlemeye başlar. Bu bakış açısında ise etmenler ve temsilcileri arasında sürekli oluşan iki yönlü mesaj trafiği sisteme oldukça yük olmaktaydı. Özellikle sisteme bağlanan etmen sayısının çok artması durumunda bu trafik tüm gridi etkileyen bir boyuta ulaşabilmektedir.
- Son versiyonda ise bu iki bakış açısının karışımı olan melez bir bakış açısı ele alınmıştır. Temsilciler ve etmenler arasında mesajlaşmaya dayana protokol tek yönlü olacak şekilde değiştirilmiştir. Buna göre her etmen belirli aralıklarla temsilcisine bir mesaj göndererek hala sisteme bağlı olduğunu bildirir. Bu mesaj gönderme frekansı da çok küçük olmayacak biçimde ayarlanarak sisteme fazlaca yük olmasına engel olunmuştur. Bunun yanında etmenlerin sistemden düzgün biçimde yani kendi iradeleri ile ayrılmaları durumunda temsilcilerine sistemden ayrıldıklarına dair bir mesaj

göndermeleri ön görülmüştür. Böylece etmenlerin sistemde ayrılmaları güvenli ve hızlı biçimde algılanabilmektedir.

İşçi ve istemci etmenlerinin sistemden çıkma protokolü arasında bazı farklılıklar bulunmaktadır. İstemci etmenlerin sisteme bağlanıp görevleri temsilcisine ilettikten sonra sistemden çıkabileceği ve bir süre sonra tekrar sisteme bağlanabileceği düşünülmüştür. Ancak işçi etmen için böyle bir durum söz konusu değildir.

İstemci etmenlerin sistemden ayrılması

İstemci etmenler için iki çeşit sistemden ayrılma protokolü tanımlanmıştır. Birinci protokolde istemci sisteme yeniden bağlanmak üzere sistemden ayrılırken ikinci protokolde temelli sistemden ayrılmaktadır. İstemci etmen için ayrıca tekrar bağlanma protokolü de tanımlanacaktır.

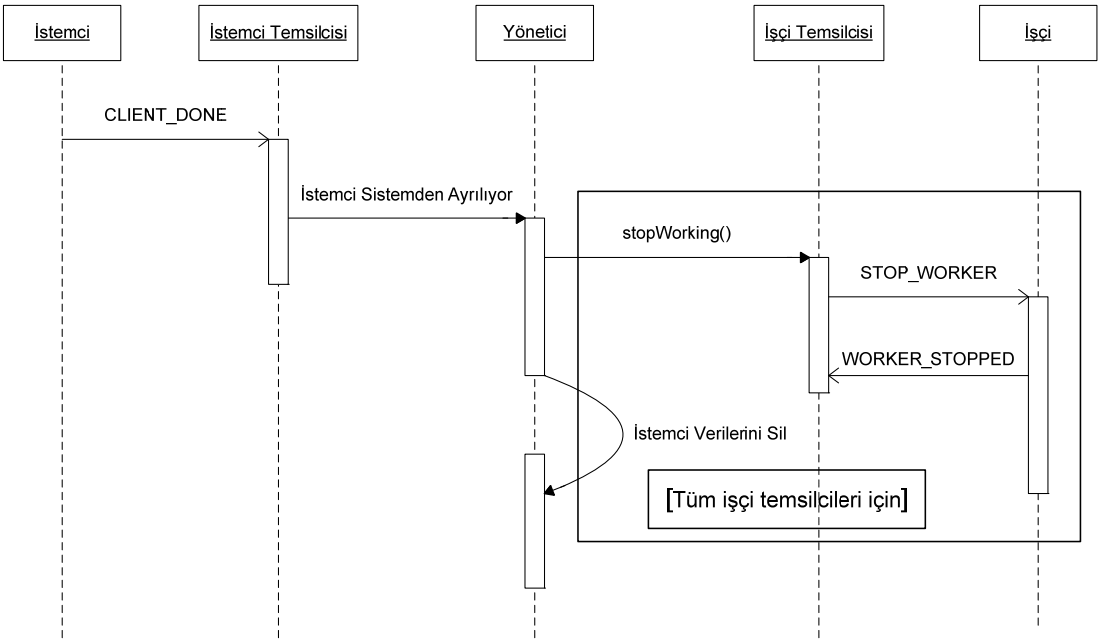
Aşağıda tekrar bağlanmak üzere ayrılma durumunda uygulanacak protokol verilmiştir.

1. İstemci temsilcisi istemci etmenin sistemde çıktığına karar verir. Bunun için aşağıdaki olaylardan birisi gerçekleşir.
    a. İstemci temsilci etmene CLIENT_UNREGISTER mesajı yollar.
    b. İstemci temsilci etmeni belirtilen süre boyunca istemci etmenden I_AM_ALIVE mesajı almamıştır.
2. İstemci temsilcisi bu noktadan itibaren istemciden gelen I_AM_ALIVE mesajını beklemekte kullandığı sınıfı yapısından çıkartır, böylece artık istemcinin sisteme bağlı olup olmadığını kontrol etmez.

İstemci etmen bir şekilde sistemden temelli çıkma kararı verebilir. Örneğin istemcinin tüm görevleri tamamlanıp, sonuçlarını alması durumunda ya da istemcinin gerçeklemesine göre kullanıcı ara yüzünden bu bağlamda bir mesaj alınması durumunda, istemcinin sistemden çıkması istenebilir. Bu durumda yürütülen protokol şu şekildedir

1. İstemci etmen temsilcisine CLIENT_DONE mesajı yollar.
2. Temsilci, istemci etmenin sistemden çıkmak istediğini algılar ve yönetici etmene durumu bildirir.
3. Yönetici etmen istemcinin atadığı görevleri yürüten işçi etmenlerin temsilcilerini bulur.

4. Yönetici etmen her işçi temsilcisinden çalışma işini sonlandırmasını ister (stopWorking)
5. İşçi temsilcisi temsil ettiği işçi etmenine STOP_WORKER mesajını yollar.
6. İşçi etmen görevi sonlandırır ve temsilcisine WORKER_STOPPED mesajı yollar, böylece işçi etmen yeni görevler alabilecektir.
7. Tüm görevleri yürüten işçi etmenler için 4. 5. ve 6. adımlar yinelenir.
8. Yönetici etmen istemci temsilcisini sonlandırır ve istemci etmene ve temsilcisine dair tuttuğu kayıtları siler.



Şekil 5.11 : İstemci etmenin sistemden ayrılması.

Şekil 5.11, istemci etmenin sistemden temelli ayrılmasına ilişkin protokolün akışı göstermektedir.

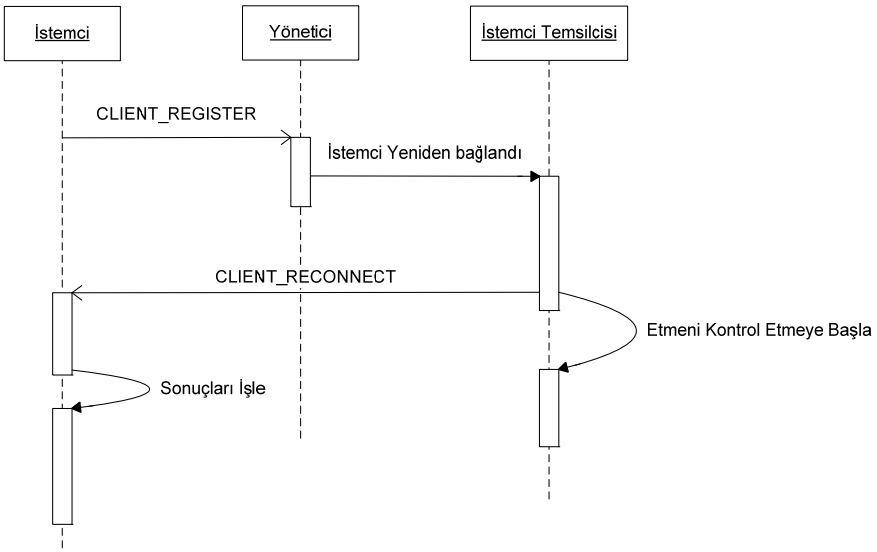
İstemci etmenlerin sisteme yeniden bağlanması

Daha önce de belirtildiği gibi istemci etmenlerin yürütülmesini istedikleri görevleri temsilcisine bildirdikten sonra sistemden ayrılma ve belirli bir süre sonra sisteme yeniden bağlanabilme yeteneklerinin olmasına karar verilmiştir. Yeniden bağlanma işlemi ilk bağlanma işlemine oldukça benzese de farklı yanları da vardır. Yeniden bağlanma protokolünde istemci etmen kendisine dair bilgileri tekrar temsilcisine göndermez, çünkü ilk bağlanma protokolünde bu işlem zaten yapılmıştır. Aynı

zamanda istemci temsilcisi istemci etmene yeniden bağlanma anına kadar alınan tüm sonuçları bildirir. İlgili protokol şu şekildedir.

1. İstemci etmen yönetici etmene CLIENT_REGISTER mesajı yollar.
2. Yönetici etmen bu mesajı alınca istemcinin daha önce sisteme bağlanıp bağlanmadığını kontrol eder. Daha önce bağlandığı sonucuna varırsa bu istemci için yaratılan temsilci etmeni bulur.
3. Bulunan temsilci etmene istemcisinin sisteme tekrar bağlandığını bildirir.
4. İstemci temsilcisi istemci etmene CLIENT_RECONNECT mesajını yollar. Bu mesajın içeriğine de o ana kadar tamamlanan görevlerin sonuçlarını ekler.
5. Aynı zamanda istemci etmenden I_AM_ALIVE mesajını beklemeye, yani tekrar istemcinin sistemden ayrılıp ayrılmayacağını düzenli olarak kontrol etmeye başlar.
6. İstemci etmen CLIENT_RECONNECT mesajını alınca mesaj içeriğindeki her bir sonucu işlemeye başlar.

Bu protokolün akışı Şekil 5. 12'de verilmiştir

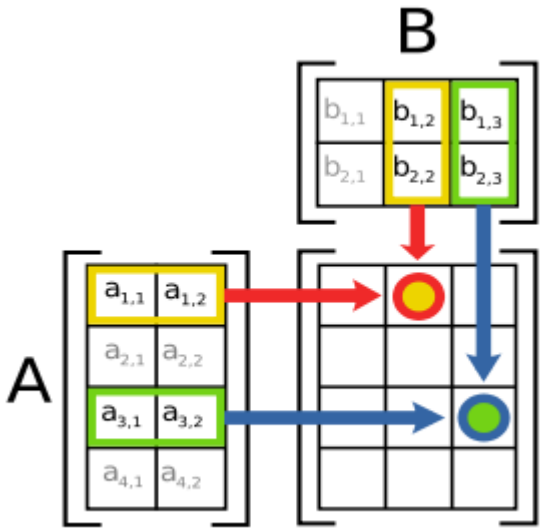


Şekil 5.12 : İstemci etmenin sisteme yeniden bağlanması.

5.4 Sistemin Testi ve Değerlendirilmesi

Önerilen sistemin testi için paralel görevlere bölünebilecek bir örneğe ihtiyaç vardır. Paralel programla ve dağıtık sistemlerin popüler bir örneği olan paralel matris çarpımı problemi ele alınmıştır.

Matrislerin çarpımı özünde basit bir işlem olsa da matrislerin boyutlarının artması durumunda hesaplama süresi oldukça uzamaktadır. Matris çarpımı en kaba haliyle satır ve sütunların kendi aralarında çarpılarak toplanması işlemidir. Şekil 5.13'de de görülebildiği gibi matris çarpımı işlemi birbirinden bağımsız satır ve sütünların çarpılıp sonuç matrisindeki hedef alana yazılması işlemidir. Bu nedenle paralelleştirilmeye oldukça musait bir işlemdir.



Şekil 5.13 : Matris Çarpımı.

Matris çarpımını gerçeklemek için test paketinde MatrixMultiplication sınıfı yazılmıştır. Bir tür istemci olan bu sınıf belirtilen boyutlarda iki tane matrisi rastgele değerler ile doldurarak yaratır. Birinci matrisin her satırına karşı ikinci matrisin her sütununu eşleyen görevler yaratır. Bu görevler de gene test paketinde bulunan MatrixMultiplicationTask sınıfıyla gerçeklenmektedir. Bu sınıftan nesneler yaratılırken birinci matristen ilgili satırın ve ikinci matristen de ilgili sütunun değerlerini alırlar. Herhangi bir işçi etmen tarafından yürütüldüklerinde yani *execute* yordamları çağırıldığında (Bkz. Şekil 5.9) bu satır ve sütun üzerinde karşılıklı değerleri çarpıp, her çarpımın sonucunu toplar, böylece sonucu oluşturur.

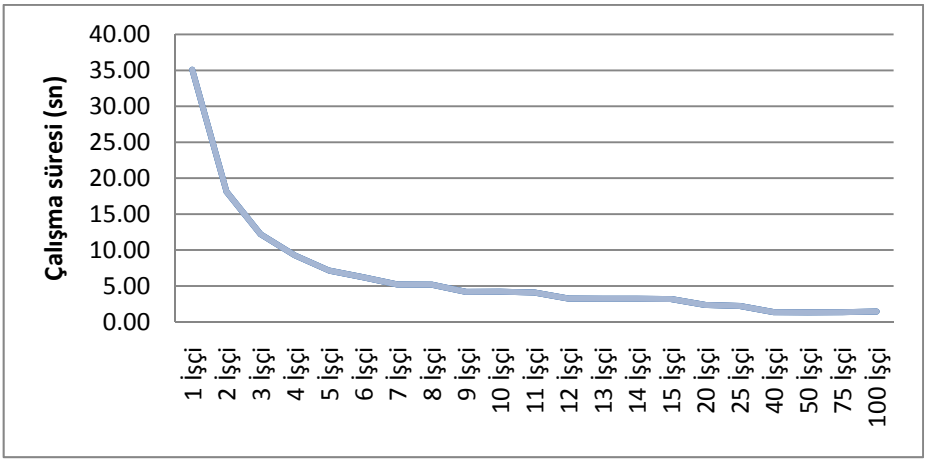
Böylece matris çarpımındaki bir parçayı yerine getirmiş olur. Daha sonra da görevlerin yürütülmesi ve sonuçlarının bildirilmesi protokolüne göre sonuçlar ilgili istemciye bildirilir.

Testler iki hedef bilgisayar üzerinde gerçekleşmiştir. Bir bilgisayar ana grid sistemini oluşturmaktadır İstemci ve işçi etmenler ise bu iki bilgisayar arasında ayrı ayrı dağıtılmışlardır.

Test durumu olarak 5x6 ve 6x7 boyutlarında iki matrisin çarpılması ele alınmıştır. Sırasıyla istemci etmenin ve işçi etmenlerin sayıları değiştirilerek sistemin çalışması değerlendirilmiştir. Ölçüm kriteri olarak görevlerin ortalama tamamlanma suresi, en kısa ve en uzun tamamlanma süreleri verilmiştir. Her istemci birinci matrisin her satırı için ikinci matrisin sütunu kadar görev yaratacaktır. Bu nedenle bir istemcinin yaratacağı görev sayısı bu örnek için 35 olacaktır.

5.4.1 İşçi Sayısındaki Artışın Sonuçlarının Ölçülmesi

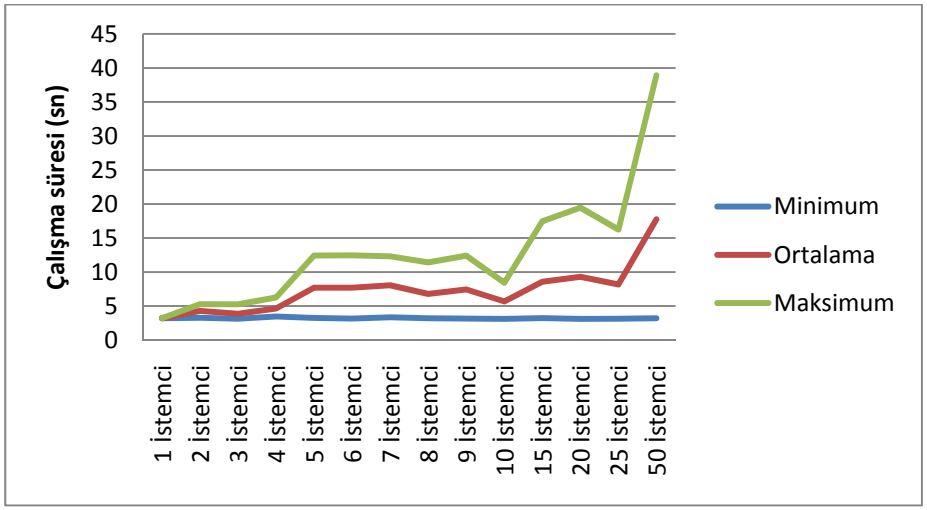
İlk testte bir tane istemci için işçi sayısı birden yüze kadar arttırılmıştır. Şekil 5.14'de yapılan testin sonuçları verilmiştir. Şekilde de görüldüğü gibi işçi sayısının artması çalışma süresini düşürmektedir. İşçi sayısının artışı bir noktadan sonra çalışma süresini azaltmamaktadır. Yaklaşık 40 işçi sayısına ulaşılınca artık 35 görevin hepsi işçilere dağıtılmış olacaktır ve hepsi de ivedilikle sonucu dönmektedir. Bu noktadan sonra işçi sayısının artması çalışma zamanı değiştirmemektedir, çünkü geçen süre mesaj iletiminde geçmektedir. İletilen mesaj sayısı her çalışmada aynı olduğu için bir noktadan sonra işçi sayısının artışının sistem performansına olumlu bir etkisinin olmadığı sonucu çıkartılabilir.



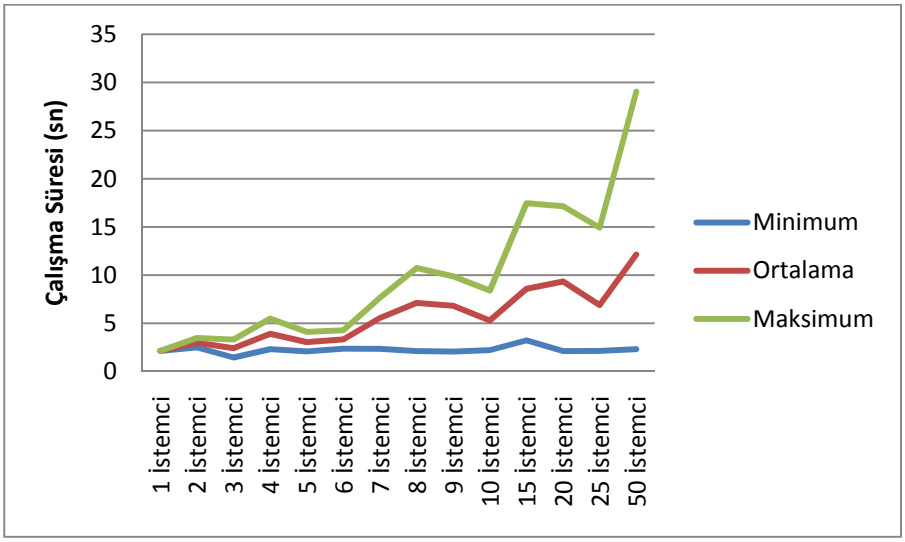
Şekil 5.14 : Bir İstemci İçin Test Sonuçları.

5.4.2 İstemci Sayısındaki Artışın Sonuçlarının Ölçülmesi

İkinci testte ise işçi sayısı sabit tutulup istemci sayısı arttırılmıştır. Böylece görev sayısındaki artışın çalışma süresine etkisi araştırılmıştır. Şekil 5.15 ve Şekil 5.16'da sırasıyla 15 ve 30 işçi katılımına karşın değişken sayıda istemci etmenin sisteme bağlanması durumunda gözlemlenen minimum, maksimum ve ortalama çalışma süreleri verilmiştir.



Şekil 5.15 : 15 İşçi – Değişken Sayıda İstemci İçin Test Sonuçları.

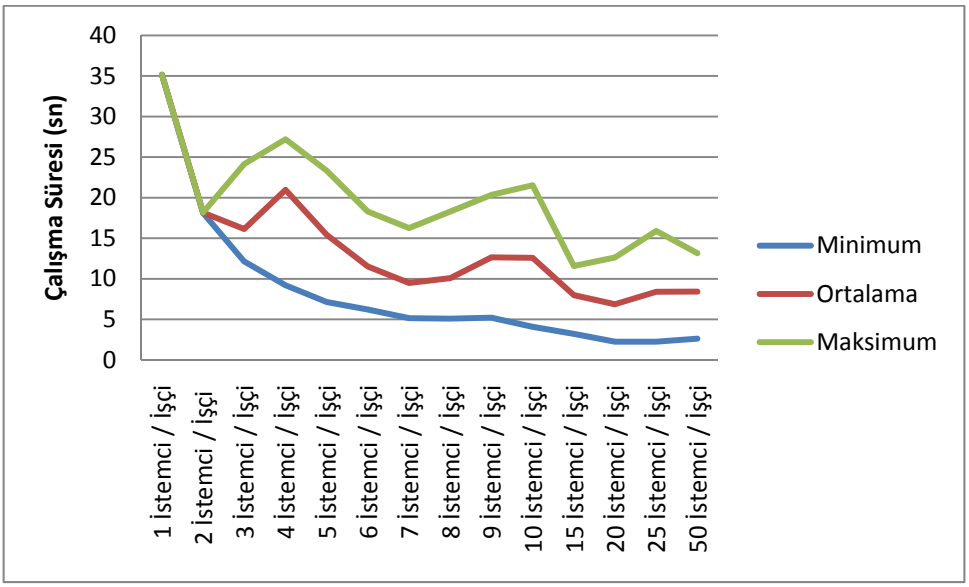


Şekil 5.16 : 30 İşçi – Değişken Sayıda İstemci İçin Test Sonuçları.

Minumum çalışma süresi en kısa zamanda sonuç verilen istemcinin çalışma süresini belirtirken maksimum çalışma süresi de sonuç almak için en fazla bekleyen istemciye dair çalışma süresidir. Ortalama değer ise tüm katılımcı etmenlerin çalışma sürelerinin ortalamasıdır. Görüldüğü gibi istemci sayısının ve dolayısıyla görev sayısının artması maksimum ve çalışma süresini arttırmıştır.

5.4.3 İstemci ve İşçi Sayısındaki Paralel Artışının Sonuçlarının Ölçülmesi

Bir diğer test ise sistemin katılımcı istemci ve işçi etmenlerinin sayısını aynı anda arttırarak yapılmıştır.



Şekil 5.17 : İşçi – İstemci Sayısının Beraber Arttırılmasına Dair Test Sonucu

Bu testin sonuçlarının verildiği Şekil 5.17'den de anlaşılacağı gibi istemci yani görev sayısının artması çalışma süresini arttırsa da, işçi sayısının artması sistemin cevap süresini esas etkileyen etken olarak öne çıkmaktadır.

5.4.4 Katılımcı Bilgisayarın Etkisi

Grid sistemine katılımcı olacak bilgisayarın etkisi, bu bilgisayarın sistemde sahip olacağı role göre değişmektedir. Yapılan testlerden de anlaşıldığı gibi beklendiği üzere sisteme daha çok istemci etmenler ekleyen katılımcı bilgisayarlar ortalama cevap süresini arttırırken, daha çok işçi etmen ekleyenler ise ortalama cevap süresini düşürme eğiliminde etki etmektedirler.

6. SONUÇ VE ÖNERİLER

Gerçeklenen etmen tabanlı grid sistemiyle esnek ve dinamik bir yapı olan grid sistemlerinin güçlü, kararlı ve kendini ispatlamış kavramlarıyla etmen sistemlerinin zeki, esnek ve dinamik yapısının bir araya getirilmesiyle kaynak paylaşımı ve dağıtılmasına yönelik bir etmen sistemi geliştirilmiştir.

Önerilen sitem grid sistemlerinin başlıca hedefi olan kaynak paylaşımı ve özellikle işlemci kaynağının paylaşımına yoğunlaşırken grid sistemlerin yöntemlerini kullanmıştır. Aynı zamanda sistemin aktörlerinin FIPA standartlarına tam uyumlu etmenler şeklinde gerçeklenmiş olması sisteme gereken esnekliği sağlamaktadır. Aynı zamanda FIPA standartlarına uygun yazılmış ve sistemin protokollerini uygulayan harici etmenler platformdan bağımsız bir şekilde sisteme bağlanıp hizmet alıp verebilmektedirler.

Kullanılan etmen sistemi ana çatısı olan JADE'e ait araçlarla sistem yöneticilerine görsel ortamlar sunulmaktadır. JADE'in sahip olduğu topluluk desteği sayesinde gelecekte olası FIPA standartlarındaki değişikliklerin saydam bir şekilde geçilebileceği düşünülmektedir.

Yapılan testler sistemin cevap hızının makul sınırlar içinde kaldığını göstermektedir. Testler sonucu elde edilen verilere göre sistem kararlı ve önceden sezilebilir davranışlar sergilemektedir. Çok sayıda etmenin aynı anda sisteme bağlanıp hizmet alabilmesine ve güvenli bir şekilde sistemden ayrılabilmesine olanak sağlamaktadır. Kullanıcılar istemci ve görev sınıflarında kendi yapacakları gerçeklemelere göre sistemin çalışma şekline ve amacına istedikleri gibi müdahale edebilmektedirler.

Geleneksel grid sistemlerinde genelde görevler arasında bir iletişim yapısı bulunmaz. Fakat önerilen bu yapıda görev sınıfları da etmen sistemlerinin gelişmiş mesajlaşma alt yapılarını kullanarak iletişime geçebilirler. Böylece daha güçlü ve esnek görev yürütme yapıları oluşturulabilinir.

6.1 Öneriler ve İleriki Çalışmalar

Bu çalışmanın kapsamı dışında kalmış olmasına rağmen dengeli bir yük dengeleme yapısı kurulmaya çalışılmıştır. Ancak sisteme eklenebilecek esnek ve kullanışlı bir yük dengeleme algoritması sisteme katkıda bulunabilir.

Ayrıca güvenlik konusu tezin kapsamı içine alınmamış bir konudur. Ancak sisteme bağlanma protokollerinde yapılacak değişiklerle kolaylıkla bir kimlik doğrulama yapısı eklenebilir. Açık kapalı anahtar şifreleme yöntemleriyle mesaj akışına etkili bir güvenlik prosedürü eklenebilir. Güvenlikle ilgili bir diğer iyileştirme ise görevlerin yürütülmesi alanında düşünülebilinir. Görevler uzaktaki bilgisayarlardan alınıp işçi bilgisayarlarda yürütülmektedirler. Yürütülen bu kod parçasının işçi etmenlerin bulunduğu sistemlere zarar vermemesi gerekmektedir. Java dilinin sağladığı bazı kod güvenliği yapıları bu alanda kullanılabilinir.

KAYNAKLAR

[1] **I. Foster, C. Kesselman, J. Nick, S. Tuecke,** 2002. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. *Open Grid Service Infrastructure WG, Global Grid Forum*, June 22, 2002.

[2] **I. Foster, Nicholas R. J., C. Kesselman,** 2004. Brain Meets Brawn: Why Grid and Agents Need Each Other. *Proceedings of 3rd Int. Conf. on Autonomous Agents and Multi-Agent Systems (AAMAS 2004),* New York, USA 2004

[3] **Jacob, B. Brown, M. Fukui, K. Trivedi N.,** 2005. Introduction to Grid Computing. *IBM*

[4] **Cicerre, F. R., Madeira, E. R., and Buzato, L. E.,** 2006. Structured process execution middleware for Grid computing: Research Articles. *Concurr. Comput. : Pract. Exper.* 18, 6 (May. 2006), 581-594

[5] **The EDUCAUSE Learning Initiative,** 2006. 7 Things You Should Know About Grid Computing, *ELI 7 Things You Should Know, EDUCAUSE Learning Initiative*

[6] **I. Foster, C. Kesselman, S. Tuecke,** 2001. The Anatomy of the Grid: Enabling Scalable Virtual Organizations**.** ***International J. Supercomputer Applications*****,** 15(3), 2001

[7] **Athanaileas, T. E., Tselikas, N. D., Tsoulos, G. V., and Kaklamani, D. I.,** 2007. An agent-based framework for integrating mobility into grid services. In *Proceedings of the 1st international Conference on Mobile Wireless Middleware, Operating Systems, and Applications* (Innsbruck, Austria, February 13 - 15, 2008). MOBILWARE, vol. 278. ICST (Institute for Computer Sciences Social-Informatics and Telecommunications Engineering), ICST, Brussels, Belgium, 1-6.

[8] **Jia Y., Rajkumar B.,** 2005. Taxonomy of Workflow Management Systems for Grid Computing. *Journal of Grid Computing, Volume 3, Numbers 3-4, Pages: 171-200,* Springer Science+Business Media B.V., New York, USA, Sept. 2005.

[9] http://www.globus.org, alındığı tarih 29.04.2009.

[10] **J. Frey, T. Tannenbaum, I. Foster, M. Livny, and S. Tuecke,** 2001**.** Condor-G: A Computation Management Agent for Multi-Institutional Grids. *Proceedings of the Tenth IEEE Symposium on High Performance Distributed Computing (HPDC10)* San Francisco, California, August 7-9, 2001

[11] **A. Natrajan, A. Nguyen-Tuong, M. Humphrey, M. Herrick, B. P. Clarke, A. S. Grimshaw,** 2002**.** The Legion Grid Portal. *Concurrency and Computation: Practice and Experience 14(13-15): 1365-1394* (2002)

[12] **The Legion Group,** 2001. Legion 1.8 Basic User Manual

[13] http://www.globus.org/ogsa, alındığı tarih 29.04.2009.

[14] **Luck, M., McBurney, P., Preist, C.,** 2003. Agent Technology: Enabling Next Generation Computing (A Roadmap for Agent Based Computing). *AgentLink. ISBN 0854 327886*

[15] **Shen, Z., Miao, C., Gay, R., and Li, D.,** 2006. Goal-Oriented Methodology for Agent System Development. *IEICE - Trans. Inf. Syst.* E89-D, 4 (Apr. 2006), 1413-1420.

[16] **Jennings, N. R., Sycara, K., and Wooldridge, M.,** 1998. A Roadmap of Agent Research and Development. *Autonomous Agents and Multi-Agent Systems* 1, 1 (Jan. 1998), 7-38.

[17] **Franklin, S. and Graesser, A.,** 1997. Is it an Agent, or Just a Program?: A Taxonomy for Autonomous Agents. In *Proceedings of the Workshop on intelligent Agents Iii, Agent theories, Architectures, and Languages* (August 12 - 13, 1996). J. P. Müller, M. Wooldridge, and N. R. Jennings, Eds. Lecture Notes In Computer Science, vol. 1193. Springer-Verlag, London, 21-35.

[18] **Henderson-Sellers, B., Gorton, I.,** 2002. Agent-Based Software Development Methodologies. *White Paper of the Workshop on Agent-Oriented Methodologies at OOPSLA2002*. Seattle, 2002

[19] **Mundle, S., Giri, N., Ray, A., and Bodhe, S.,** 2009. JADE based Multi Agent system for mobile computing for cellular networks. *Proceedings of the international Conference on Advances in Computing, Communication and Control* (Mumbai, India, January 23 - 24, 2009). ICAC3 '09. ACM, New York, NY, 467-473.

[20] **Wooldridge, M.,** 1997. Agent-Based Software Engineering. *IEE Proceedings on Software Engineering* 1997 26-37

[21] **Yoon, Y., Lee, G., Choi, K., and Shin, D.,** 2008. Design of Agent Service System to Manage Services among Heterogeneous Multi-agent Systems. *Proceedings of the 2008 IEEE international Workshop on Semantic Computing and Applications - Volume 00* (July 10 - 11, 2008). IWSCA. IEEE Computer Society, Washington, DC, 123-125.

[22] **Finin, T., Fritzson, R., McKay, D., and McEntire, R.,** 1994. KQML as an agent communication language. *Proceedings of the Third international Conference on information and Knowledge Management* (Gaithersburg, Maryland, United States, November 29 - December 02, 1994). N. R. Adam, B. K. Bhargava, and Y. Yesha, Eds. CIKM '94. ACM, New York, NY, 456-463

[23] **Foundation for Intelligent Physical Agents,** 2002. FIPA ACL Message Structure Specification

[24] **Rao, A. S.,** 1996. AgentSpeak(L): BDI agents speak out in a logical computable language. In *Proceedings of the 7th European Workshop on Modelling Autonomous Agents in A Multi-Agent World : Agents Breaking Away: Agents Breaking Away* (Einhoven, The Netherlands). W. Van de Velde and J. W. Perram, Eds. Springer-Verlag New York, Secaucus, NJ, 42-55.

[25] **Morley, D. and Myers, K.,** 2004. The SPARK Agent Framework. *Proceedings of the Third international Joint Conference on Autonomous Agents and Multiagent Systems - Volume 2* (New York, New York, July 19 - 23, 2004). International Conference on Autonomous Agents. IEEE Computer Society, Washington, DC, 714-721

[26] **van Riemsdijk, M. B., de Boer, F. S., Dastani, M., and Meyer, J. C.,** 2006. Prototyping 3APL in the Maude term rewriting language. *Proceedings of the Fifth international Joint Conference on Autonomous Agents and Multiagent Systems* (Hakodate, Japan, May 08 - 12, 2006). AAMAS '06. ACM, New York, NY, 1279-1281.

[27] **Busetta, P., Rönnquist, R., Hodgson, A., Lucas, A.,** 1999. JACK Intelligent Agents - Components for Intelligent Agents in Java. *AgentLink News (2)*. 2-5.

[28] **Howden, N., Rönnquist, R., Hodgson, A., and Lucas, A.,** 2001. JACK intelligent agents - summary of an agent infrastructure. *Proceedings of the 5th International Conference on Autonomous Agents (Agents '01).*

[29] **Ingrand, F. F., Georgeff, M. P., and Rao, A. S.,** 1992. An architecture for Real-Time Reasoning and System Control. *IEEE Expert: Intelligent Systems and Their Applications* 7, 6 (Dec. 1992), 34-44.

[30] **Karen L. M.,** 1996. A Procedural Knowledge Approach to Task-level Control.*Proceedings of the 3rd International Conference on Artificial Intelligence Planning Systems (AIPS-96)* 158-165

[31] **d'Inverno, M., Kinny, D., Luck, M., and Wooldridge, M.,** 1998. A Formal Specification of dMARS. *Proceedings of the 4th international Workshop on intelligent Agents Iv, Agent theories, Architectures, and Languages* (July 24 - 26, 1997). M. P. Singh, A. S. Rao, and M. Wooldridge, Eds. Lecture Notes In Computer Science, vol. 1365. Springer-Verlag, London, 155-176.

[32] **Poggi, A., Tomaiuolo, M., and Turci, P.,** 2004. Extending JADE for Agent Grid Applications. *Proceedings of the 13th IEEE international Workshops on Enabling Technologies: infrastructure For Collaborative Enterprises* (June 14 - 16, 2004). WETICE. IEEE Computer Society, Washington, DC, 352-357.

[33] **P. Mitkas and A. Symeonidis and D. Kechagias and I. Athanasiadis and G. Laleci and G. Kurt and Y. Kabak and A. Acar and A. Dogac**, 2002. An Agent Framework for Dynamic Agent: Retraining Agent Academy. *Stanford-Smith, B., Chiozza, E., & Edin, M. (Editors), Challenges and Achievements in e-business and e-work* Prague, Czech Republic, IOS Press, October 2002, 757-764

[34] **Rita C. N., and Andries B.,** 2007. A Generic Agent Framework to Support the Various Software Project Management Processes. *Interdisciplinary Journal of Information, Knowledge, and Management Volume 2, 2007*

[35] **Henri A., Analía A.,** 2000. A Java Framework for Multi-agent Systems. *SADIO Electronic Journal of Informatics and Operations Research vol. 3, no. 1.* 1-12

[36] http://sourceforge.net/projects/fleeble alındığı tarih 29.04.2009.

[37] http://www.lostwax.com/agents/framework/ alındığı tarih 29.04.2009.

[38] **Fabio B., Giovanni C., Tiziana T. Giovanni R.,** 2007. JADE Programmer's Guide

[39] **B.J. Overeinder, N.J.E. Wijngaards, M. van Steen, and F.M.T. Brazier**, 2002. Multi-Agent Support for Internet-Scale Grid Management. *AISB'02 Symposium on AI and Grid Computing* 18-22

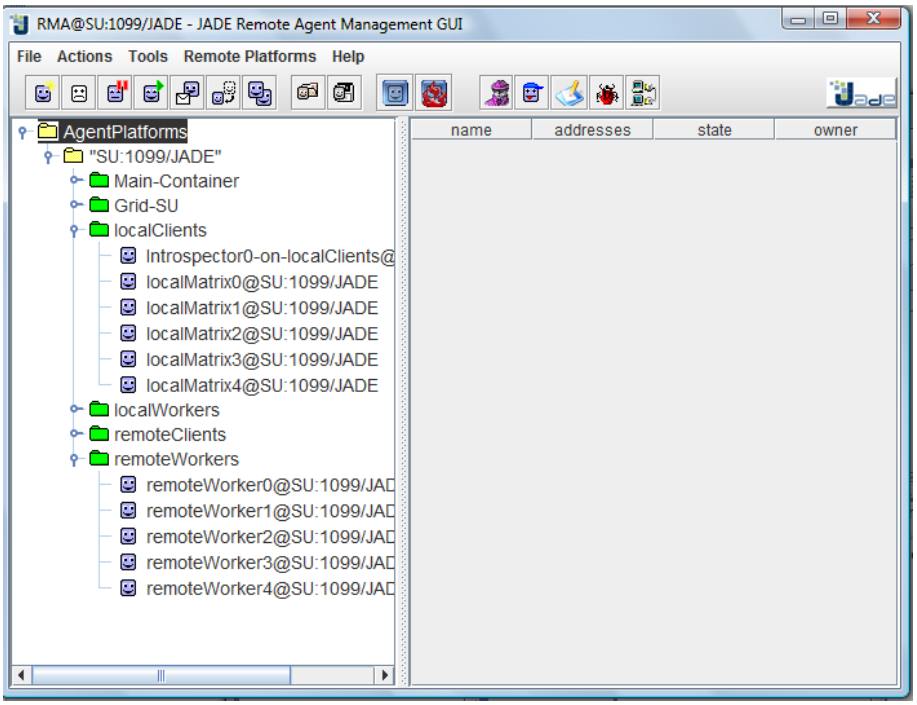
[40] **Schmorrow D.,** 2002. The DARPA Control of Agent Based Systems (CoABS) Program and Challenges for Collaborative Coalitions

[41] **Munehiro F., Duncan S.,** 2006. UWAgents: A Mobile Agent System Optimized for Grid Computing. *Proceedings of the 2006 International Conference on Grid Computing & Applications, GCA 2006, USA.*

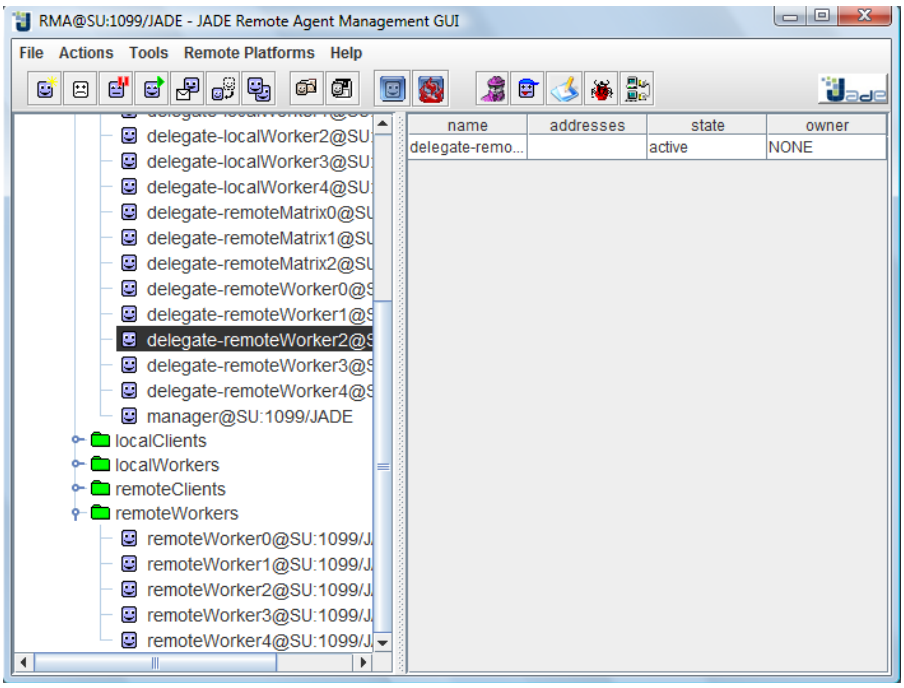
[42] http://jade.tilab.com alındığı tarih 29.04.2009.

EKLER

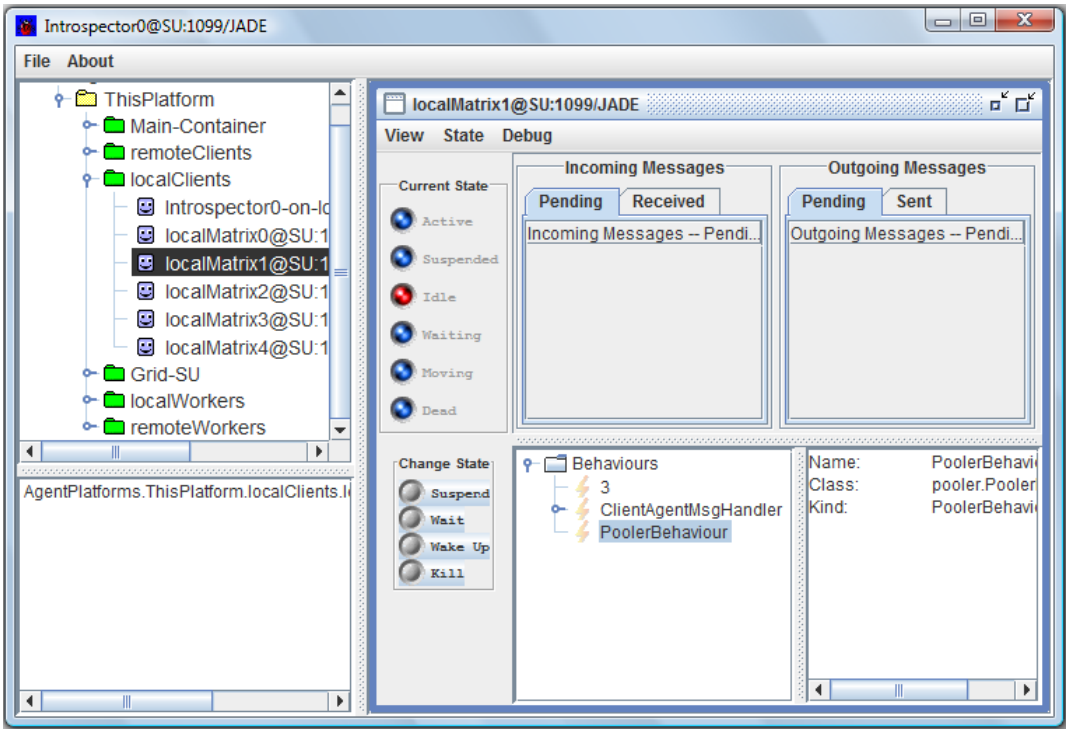
EK A.1 : Program Çalışma Görüntüleri



Şekil A.1 : Yerel ve Uzak Katılımcıların Bulunduğu bir Sistem



Şekil A.2 : Yönetici ve Temsilci Etmenler



Şekil A.3 : Introspector Etmeni (localMatrix1'in yaşam döngüsü)

ÖZGEÇMİŞ

Ad Soyad:	Uygar GÜMÜŞ
Doğum Yeri ve Tarihi:	Kayseri / 16.08.1983
Lisans Üniversite:	İstanbul Teknik Üniversitesi / Bilgisayar Mühendisliği