YÜKSEK DÜZEYLİ MİMARİNİN, KOŞUT İŞLEM GEREKSİNİMİ OLAN BİLİMSEL BENZETİM UYGULAMALARI İÇİN UYGUNLUĞUNUN İNCELENMESİ

INVESTIGATION OF SUITABILITY OF THE HIGH LEVEL ARCHITECTURE (HLA) FOR SCIENTIFIC SIMULATION APPLICATIONS THAT REQUIRE PARALLEL PROCESSING

TÜLİN ERÇELEBİ

Hacettepe Üniversitesi

Lisansüstü Eğitim-Öğretim ve Sınav Yönetmeliğinin

BİLGİSAYAR MÜHENDİSLİĞİ Anabilim Dalı için Öngördüğü

YÜKSEK LİSANS TEZİ

olarak hazırlanmıştır.

2008

Fen Bilimleri Enstitüsü Müdürlüğü'ne,

Bu çalışma jürimiz tarafından **BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI'nda YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Üye (Danışman) :.….....
(Yrd. Doç. Dr. Kayhan İMRE)

Üye :.….....
(Yrd. Doç. Dr. Harun ARTUNER)

Üye :.….....
(Yrd. Doç. Dr. Ali Ziya ALKAR)

ONAY

Bu tez ...../...../..... tarihinde Enstitü Yönetim Kurulunca kabul edilmiştir.

...../...../.....

Prof. Dr. Erdem YAZGAN
FEN BİLİMLERİ ENSTİTÜSÜ MÜDÜRÜ

YÜKSEK DÜZEYLİ MİMARİNİN, KOŞUT İŞLEM GEREKSİNİMİ OLAN BİLİMSEL BENZETİM UYGULAMALARI İÇİN UYGUNLUĞUNUN İNCELENMESİ

Tülin Erçelebi

ÖZ

Yüksek Düzeyli Mimari (HLA-*High Level Architecture*), birden fazla benzetim uygulamasının dağıtılmış bir ortamda tek bir benzetim uygulaması gibi çalışmasını sağlayan bir yazılım mimarisidir. Standartlaşmış (IEEE 1516) olan bu mimari, Amerikan Savunma Bakanlığı tarafından Amerika Birleşik Devletleri için geliştirilen savunma amaçlı benzetim uygulamalarının gerçekleştirilmesinde zorunlu tutulmaktadır. Türkiye'de de savunma amaçlı geliştirilecek benzetim uygulamalarının birçoğunda bu standarda uyumluluk şart koşulmaktadır.

Yüksek Düzeyli Mimarinin sivil uygulamalarda da kullanımı yaygınlaşmaktadır. Yüksek başarımlı işlem sığasına gereksinim duyan bilimsel benzetim uygulamaları da Yüksek Düzeyli Mimariyle gerçekleştirilmek için iyi birer aday olarak gözükmektedir.

Bu tez kapsamında yaygın kullanılan bilimsel benzetim uygulamaları incelenmiş ve bu uygulamalar Yüksek Düzeyli Mimari ile gerçekleştirilmek istenildiğinde elde edilebilecek yararların ve önümüze çıkabilecek engellerin neler olabileceği belirlenmiştir.

Anahtar Kelimeler: Benzetim, Yüksek Düzeyli Mimari (HLA)

Danışman: Yrd. Doç. Dr. Kayhan İMRE, Hacettepe Üniversitesi, Bilgisayar Mühendisliği Bölümü

INVESTIGATION OF SUITABILITY OF THE HIGH LEVEL ARCHITECTURE (HLA) FOR SCIENTIFIC SIMULATION APPLICATIONS THAT REQUIRE PARALLEL PROCESSING

Tülin Erçelebi

ABSTRACT

High Level Architecture (*HLA*) is a software architecture that ensures more than one simulation applications to run on a distributed environment as if they are a single simulation. This architecture, which has already been standardized (*IEEE 1516*), is obligatory for the implementations of the simulations developed by the US Department of Defense for the defense applications of the USA. It is also obligatory in Turkey for most of the defense-purposed simulations to be compatible with this standard.

Usage of High Level Architecture in civil applications becomes widespread as well. The scientific simulation applications, which require high performance computing capacity, are good candidates to be realized with High Level Architecture.

Within the scope of this thesis, the common simulation applications have been investigated, and it has been determined what would the benefits and obstacles be when these simulations are implemented with High Level Architecture.

Keywords: High Level Architecture (HLA), Simulation

Advisor: Associate Professor Kayhan İMRE, Hacettepe University, Department of Computer Science and Engineering.

Sevgili babam Dr. Hasan ERÇELEBİ ve

annem Ferah ERÇELEBİ'ye...

TEŞEKKÜR

Yazar, bu çalışmanın gerçekleşmesinde katkılarından dolayı, aşağıda adı geçen kişi ve kuruluşlara içtenlikle teşekkür eder.

Yrd. Doç. Dr. Kayhan İmre, tez konusunun belirlenmesini sağlamış, tez çalışmasının gerçekleştirilmesi ve tez metninin hazırlanması sırasında yol gösterici olmuştur.

Seda Güneş, Melisa Ceren Ertan ve Ersin Er her zaman iyi birer takım arkadaşları olmuş, önemli sorunları çözmüş ve desteklerini hiç esirgememişlerdir.

Nizam Ayyıldız, tezin gerçekleştirimi esnasında değerli bilgileriyle birçok problemi aşmama yardımcı olmuş ve tez metnini inceleyerek son düzenlemelerde yardımcı olmuştur.

Ayrıca sevgi ve desteklerini hiç esirgemeyen, bana her zaman güvenen, beni bugünlere getiren aileme sonsuz teşekkür ediyorum.

İÇİNDEKİLER DİZİNİ

ŞEKİLLER DİZİNİ

Sayfa

ÇİZELGELER DİZİNİ

Sayfa

SİMGELER VE KISALTMALAR

DMSO	Defense Modeling and Simulation Office
FED	Federation Execution Data
FIPA	Foundation for Intelligent Physical Agents
FOM	Federation Object Model
HLA	High Level Architecture
KQML	Knowledge and Query Manipulation Language
LBTS	Lower Bound Time Stamp
MD	Molecular Dynamics
MIMD	Multiple Instruction Multiple Data
MPI	Message Passing Interface
OMDT	Object Model Definition Tool
OMT	Object Model Template
RTI	Run Time Infrastructure
SOM	Simulation Object Model
TSO	Time Stamped Order

1. GİRİŞ

Benzetim, gerçek bir süreç veya sisteminin işletilmesinin zaman üzerinden modellenmesi için kullanılan bir araçtır.

Örneğin; bilgisayar üzerindeki bir araç simülatörü, trafiğin bazı kurallarının bir bilgisayar üzerinde öğretilmesi amacıyla kullanılan bir benzetim modelidir. Sürücünün aracı kullanırken göreceği görüntünün bir benzerini bilgisayar ekranında görmesi ve aracı kontrol etme işlemlerinin gerçek yaşamda olduğu gibi tasarlanması, bir benzetim olayıdır.

Benzetim sistemleri çoğu zaman dağıtılmış ortamlarda çalışan bileşenlerden oluşur. Bu bileşenlerin kolayca yeniden kullanılabilmesi ve birlikte çalışabilir olması, geliştirme sürecinin etkinliğini ve sistemin yeteneklerini önemli ölçüde artıracaktır. Benzetim bileşenlerinin birlikte çalışabilmeleri ve yeniden kullanılabilmeleri için önceden tanımlanmış birtakım kurallara uymaları gerekir. HLA (*High Level Architecture*) bu ihtiyaçları karşılamak için DMSO (*Defense Modeling and Simulation Office*) tarafından geliştirilmiştir [1][2]. HLA' da benzetim bileşenleri federe, benzetim sistemi de federasyon olarak adlandırılır. HLA temelde, federasyon kuralları, arayüz belirtimi ve nesne modeli şablonundan (OMT – *Object Model Template*) oluşur. Federasyon kuralları, federelerin ve federasyonun uyması gereken kurallardır. HLA'nın temel kurallarından biri federe etkileşimlerinin bir altyapı üzerinden sağlanmasıdır. Bu altyapı işletim zamanı altyapısı (RTI – *Runtime Infrastructure*) olarak adlandırılır. Arayüz belirtimi federelerin RTI üzerinden alacağı hizmetler ve RTI'nın federeler üzerinden kullanabileceği geri dönüş (*callback*) çağrılarından oluşur. HLA belirtiminde federelerin işletim zamanında kullanacakları nesneler ve gönderip alabilecekleri etkileşimler nesne modeli şablonları ile tanımlanır. Federasyondaki her federe için bir benzetim nesne modeli (SOM), federasyon genelindeki paylaşılan nesne tanımları için bir federasyon nesne modeli (FOM) bulunur. RTI'nın kullanacağı nesne modeli federasyon çalışma bilgisi (FED) olarak adlandırılır [3].

Tez metninin ilerleyen kesimlerinde öncelikle yaygın kullanılan bilimsel benzetim uygulamaları incelenecektir. Daha sonra bu uygulamalardan molekül dinamiği (*molecular dynamics*), akışkanlar dinamiği başlığı altında hava ve iklim tahminleri

(*weather and climate predictions*), son olarak da yapay zekanın bir alt kolu olan akıllı ajan (*intelligent agent*) incelenecek, sonraki kesimde de, bu bilimsel benzetim uygulamalarında HLA'nın uygunluğu teorik olarak sonuçlandırılacaktır.

Hava tahmin benzetimi ve molekül dinamiği benzetiminin HLA ile tasarlanmış bir örneğine rastlanmamıştır. Bu benzetimlerin ilk kez bu tezde HLA'nın uygunluğunun incelenmesiyle, teorik bir tasarım modelinin de sunuluyor olması bu tezde motive edici bir etken olmuştur.

Bu tezden çıkarılacak sonuçların, bir sonraki adım olan bilimsel benzetim sistemlerinin, HLA ile gerçekleştirimi konusunda önemli ölçüde fayda sağlayacağı düşünülmektedir.

2. BENZETİM NEDİR?

Benzetim, gerçek bir dünya süreci veya sisteminin işletilmesinin zaman üzerinden taklit edilerek, sistem nesneleri arasında tanımlanmış ilişkileri içeren sistem veya süreçlerin bir modelidir.

Benzetim günümüzde mevcut olan ve daha önemlisi yarın da mevcut olabilecek işlemler hakkında nesnel bilgiler sağlar. Örneğin bilgisayar üzerindeki bir uçuş benzeticisi, uçuşun bazı kurallarının bir bilgisayar üzerinde öğretilmesi amacıyla kullanılan bir benzetim modelidir. Pilotun kokpitte göreceği ekranın bir benzerini bilgisayar ekranında görmesi ve uçuşu kontrol etme işlemlerini sanki gerçekten de uçaktaymış gibi yapması, bir benzetim olayıdır. Bu tez kapsamında incelenecek olan benzetim uygulamaları koşut ve dağıtılmış ortamda çalışabilecek olan uygulamalardır.

2.1. Koşut ve Dağıtılmış Benzetim (*Parallel And Distributed Simulation*)

Koşut ayrık olay benzetimi (*parallel discrete event simulation*), benzetim programlarının çok işlemcili ortamlarda çalıştırılması ile ilgilidir. Dağıtılmış benzetim ise benzetimlerin birbirlerine yerel alan ağı ya da geniş alan ağı ile bağlanmış, fiziksel konumları itibari ile dağıtılmış bilgisayarlar üzerinde çalıştırılması ile ilgilenmektedir. Her iki durumda da tek bir benzetim modelinin (bazen bu model birden fazla benzetimin bir araya gelmesiyle de oluşabilir) işletilmesi çok sayıda işlemci üzerinde gerçekleştirilir [5].

2.2. Benzetim Çeşitleri

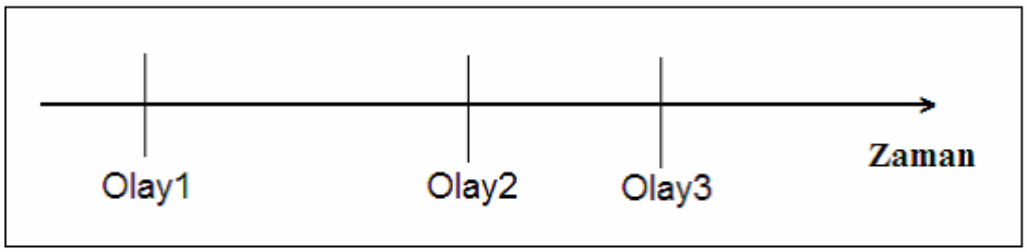
Dağıtılmış koşut benzetim uygulamaları kesikli ve sürekli olaylar benzetimi olmak üzere ikiye ayrılır.

2.2.1. Kesikli (*Discrete*) Olaylar Benzetim Yöntemi

Kesikli benzetim bir sistemin davranışındaki değişimleri yalnızca belirli bir anda inceleyen benzetim tekniğidir. Kesikli benzetim, çoğunlukla bekleme hattı dediğimiz kuyruk sistemlerinde kullanılır. Bu kuyruk sistemlerine örnek olarak banka vezneleri, trafik ve otoparklar verilebilir.

Sistemin kesikli olarak incelenmesindeki temel amaç, sistemin en son bilinen veya o anki durumunun, yalnızca sisteme yeni bir giriş veya çıkış olduğunda değişmesidir. Şekil 2.1'de kesikli olaylar benzetim yapısı görülmektedir.

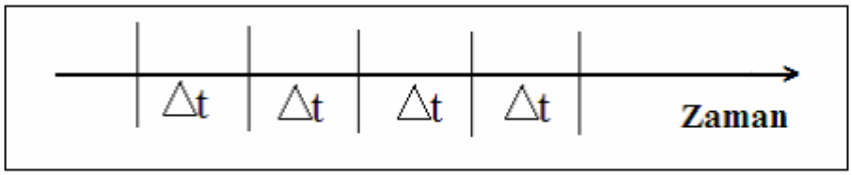
Tezin ilerleyen kısımlarında kesikli benzetim sisteminin özel bir uygulaması (standardı) olan HLA incelenecektir.



Şekil 2.1. Kesikli Olaylar Benzetim Yapısı

2.2.2. Sürekli (*Continuous*) Benzetim Yöntemi

Bazı sistemlerde olaylar tüm zaman sürecinde sürekli değişebilir ve bu değişme bazı kesikli olayların aynı zamanlarında olmayabilir. Örneğin bir su deposundaki su seviyesi, giren ve çıkan akışkan debisine bağlı olarak bütün zaman içinde değişebilir. Böyle bir olay için sürekli benzetim yapmak daha uygundur. Buna rağmen kesikli benzetim yöntemi de burada bir yaklaşım yöntemi olarak kullanılabilir. Şekil 2.2'de sürekli benzetim yapısı görülmektedir.



Şekil 2.2. Sürekli Benzetim Yapısı

2.2.3. Hibrit Benzetim Yöntemi

Hibrit benzetim yöntemi, kesikli olaylar benzetim yöntemi ve sürekli benzetim yöntemlerinin her ikisini de kapsayan yöntemdir.

3. KOŞUT İŞLEM

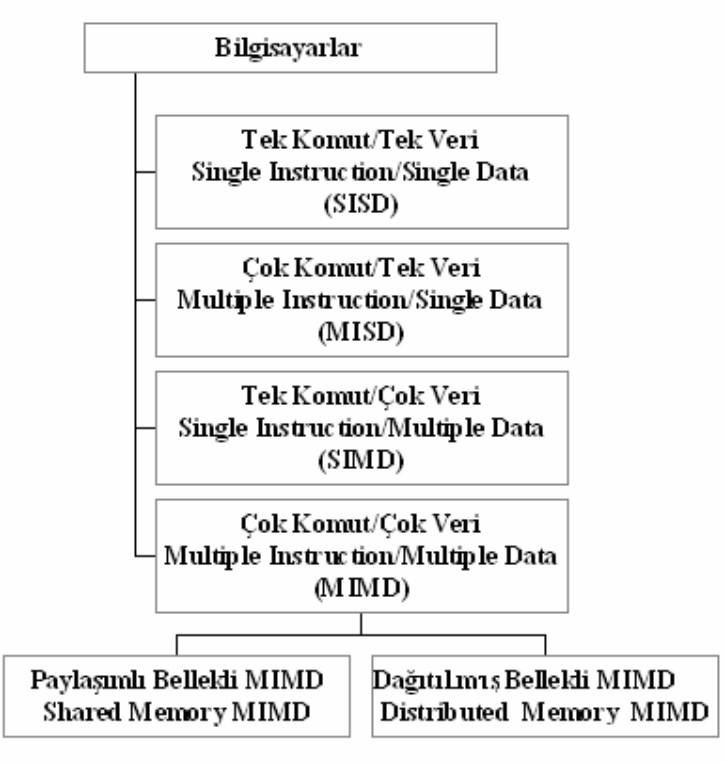
Günümüzde ağ teknolojisindeki hızlı gelişmelerin koşut programlama için kolay ulaşılabilir ve edinilebilir donanımlara olanak sağlaması ve yüksek başarım gerektiren uygulamaların toplam çalışma zamanının çok fazla olması koşut programlamanın önemini artırmaktadır.

Büyük uygulamalarda daha hızlı sonuç alabilmek için uygulamaya ait program parçalarının birden fazla işlemcide eş zamanlı çalıştırılmasına koşut programlama denir. Günümüzde koşut programlama kullanılarak geliştirilen en belirgin uygulamalar, akışkan dinamiği, hava tahmin benzetimi, kimyasal ve nükleer reaktör benzetimi ve hesaplanabilir kimya uygulamalarıdır.

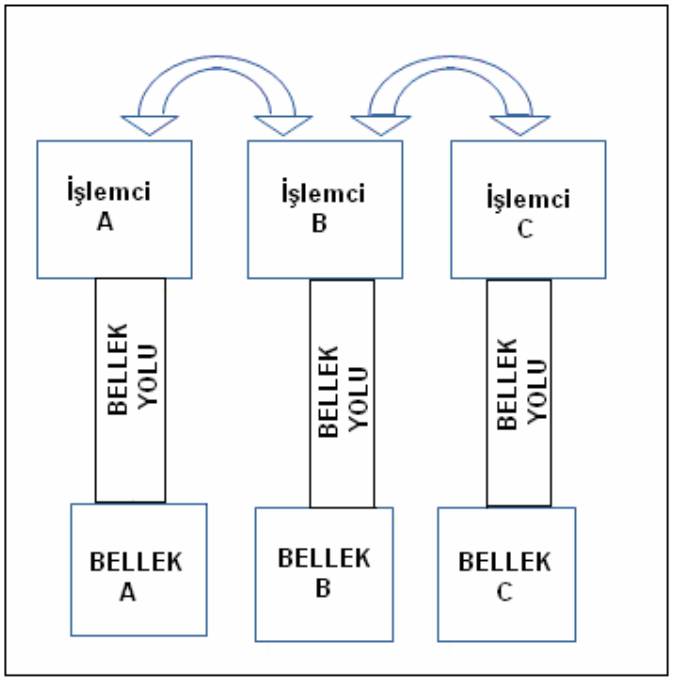
Koşut programlar en iyi verim alabilmek için gelişmiş mimarilere sahip koşut bilgisayar sistemleri üzerinde çalıştırılırlar.

Koşut bilgisayarların kendi aralarında değişik sınıflandırma biçimleri vardır. En yaygın kullanılan koşut bilgisayar sınıflandırması Michael J. Flynn'ın sınıflandırmasıdır. Bu sınıflandırma, komut ve veri akışlarına göre Şekil 3.1'de gösterilmiştir.

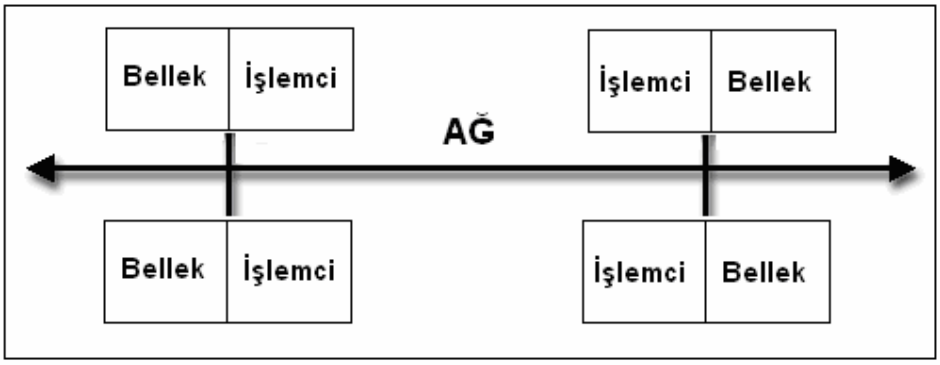
Şekil 3.1' de gösterilen bilgisayar sınıflandırmasına ek olarak, bilgisayarlar bellek çeşitlerine göre paylaşımlı bellek (*shared memory*) ve dağıtılmış bellek (*distributed memory*) olmak üzere ikiye ayrılır. Paylaşımlı bellekte işlemciler ortak bellek kullanırlar ve aynı zaman diliminde sadece bir işlemci belleğe ulaşabilir. Dağıtılmış bellekli mimarilerde ise her işlemcinin kendisine ait belleği vardır. İşlemciler arası veri paylaşımı ağ üzerinden yapılmaktadır. En sık kullanılan koşut bilgisayar çeşitlerinden biri olan dağıtılmış bellekli MIMD bilgisayarı ve dağıtılmış bellekli mimarinin genel görünümü Şekil 3.2 ve Şekil 3.3' de gösterilmiştir.



Şekil 3.1. Flynn'ın Sınıflandırması



Şekil 3.2. Dağıtılmış Bellekli MIMD Bilgisayar



Şekil 3.3. Dağıtılmış Bellek Mimarisi

3.1. Koşut Programlama Yaklaşımları

Yukarıdaki bölümde koşut bilgisayar mimarileri ve bellek mimarilerinden bahsedilmişti. Bu adımda donanım ve bellek mimarilerinden bağımsız olarak kullanılan koşut programlama yaklaşımlarından bahsedilecektir.

Bilinen koşut programlama yaklaşımlarından bazıları paylaşımlı bellek (*shared memory*), iş parçacıkları (*threads*), ileti geçirme (*message passing*), koşut veri (*data parallel*) ve hibrit (*hybrid*) yaklaşımlarıdır. Bu yaklaşımlar arasında en yaygın olarak kullanılan iki tanesi ileti geçirme ve koşut veri yaklaşımlarıdır.

3.1.1. İleti Geçirme (*Message Passing*)

İleti geçirme yaklaşımında işlemciler kendi yerel (*local*) belleklerini kullanırlar. İşlemciler arasındaki haberleşme ileti alıp-verme şeklinde gerçekleşir. Veri aktarımı bütün işlemcilerin işbirliği ile sağlanır.

İşlemciler arası veri değişimi ileti geçirme kütüphaneleri tarafından gerçekleştirilir. En yaygın olarak kullanılan ileti geçirme kütüphaneleri MPI (*Message Passing Interface*) ve PVM'dir (*Parallel Virtual Machine*).

3.1.2. Koşut Veri (*Data Parallel*)

Koşut veri yaklaşımında her işlemci veri yapısının farklı bir parçası üzerinde çalışır. İleti geçişleri programcıya gözükmeden yapılır.

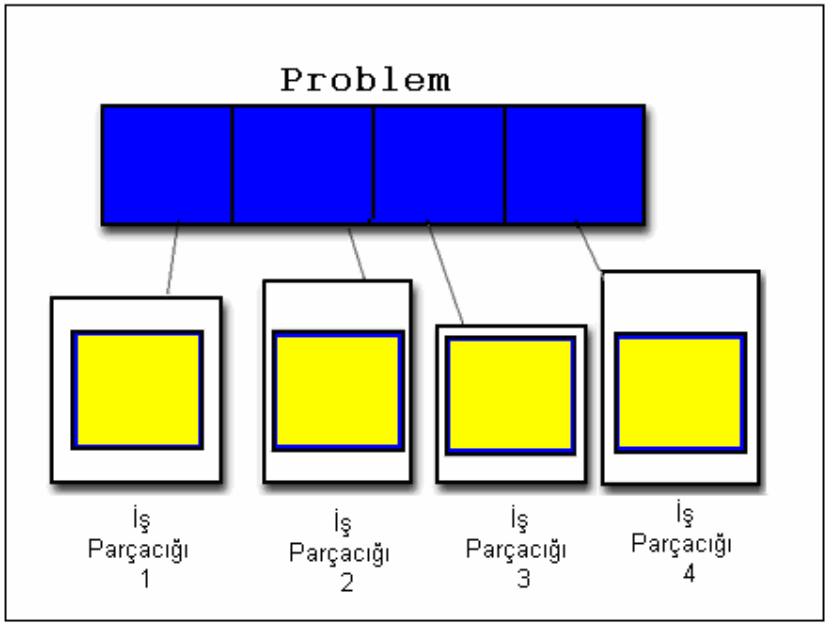
Koşut veri yaklaşımı kullanılmak istendiğinde, yazılan kod koşut veri derleyicinde derlenmelidir. Derleyici programı standart koda çevirip, veriyi işlemcilere dağıtabilmek için ileti geçirme kütüphanesini çağırır.

3.2. Problemlerin Koşut Programlama Yaklaşımı ile Çözümlenmesi

Bir problemi ya da bu problemi çözmek için geliştirilmiş olan ardıl bir programı koşutlaştırmak için tanımlanan ilk adımlardan biri, programı küçük parçalara bölmektir. Bu yönteme bölümleme (partitioning) denir. Bilinen en temel iki bölümleme yaklaşımı alan ayrıştırma (*domain decomposition*) ve işlevsel ayrıştırmadır (*functional decomposition*).

3.2.1. Alan Ayrıştırma (*Domain Decomposition*)

Alan ayrıştırma yönteminde her iş parçacığı verinin bir parçasıyla ilgilenir. Şekil 3.4'de alan ayrıştırma yöntemi gösterilmiştir.



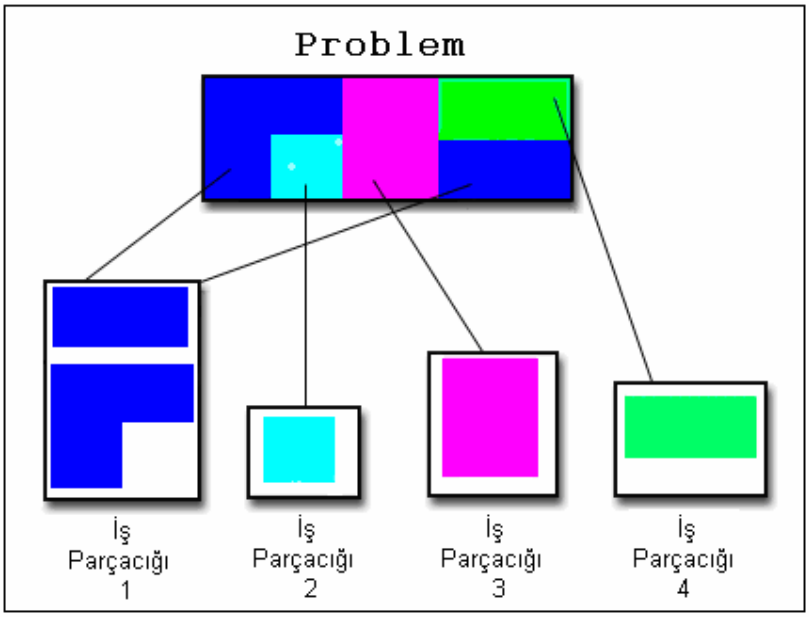
Şekil 3.4. Alan Ayrıştırma Yöntemi

3.2.2. İşlevsel Ayrıştırma (*Functional Decomposition*)

Alan ayrıştırma yönteminden farklı olarak işlevsel ayrıştırma yönteminde görevler yapılacak işin durumuna göre iş parçacıklarına bölünür. Eşit bir dağılım söz konusu değildir.

İşlevsel ayrıştırma, eko sistem modellemede, sinyal işleme uygulamalarında ve iklim modelleme alanlarında sıkça tercih edilen bir ayrıştırma yöntemidir.

Şekil 3.5'de problem değişik iş parçacıklarına bölünmüş, her iş parçacığına düşen görevler gösterilmiştir.



Şekil 3.5. İşlevsel Ayrıştırma

Bir programı koşutlaştırmak için kullanılan adımlardan ilki bölümlemeydi. İkinci adım ise haberleşmedir (*communication*). Problemin tanımına göre bölümlenen parçaların her birinin hangi parçalar ile ve nasıl haberleşeceği bu adımda irdelenir.

3.3. Koşut Bilgisayarların Bugünü ve Geleceği

Yüksek sayıda merkezi işlem birimi ve ilgili donanımlarının bir arada bulundurulduğu koşut bilgisayar sistemlerine süper bilgisayarlar (*supercomputers*) denilmektedir. Günümüzde hemen hemen her bilim dalı ve mühendislik alanlarında süper bilgisayarlar kullanılmaktadır. Örneğin, hava tahminlerinde, iklim modellemede, uçak tasarımında, savunma sanayiinde, çarpışma testlerinde sıkça kullanılmaktadır.

Dünyanın en hızlı bilgisayarlarını belirleyen Top500 süper bilgisayar listesinde birinciliği, saniyede 280 trilyon işlem yapabilen ve 131 bin 72 adet işlemcinin birleşiminden oluşan IBM Blue Gene/L isimli bilgisayar sürdürmektedir.

IBM'nin halen Amerika Savunma Bakanlığı'na bağlı çalışan İleri Savunma Araştırmaları Projeleri Ajansı (*DARPA-Defense Advanced Research Projects Agency*) için yaptığı çalışma tamamlandığında saniyede 10.000 trilyon (*10 petaflop*) işlem kapasitesini aşan bir makine üretileceği söylenmektedir [18].

Bu teknolojik gelişmeler karşısında, her geçen gün süper bilgisayarlara olan ilgi artmakta ve yaygın olarak kullanılmaya başlanmaktadır. Yakın gelecekte hemen hemen her bilim dalında koşut bilgisayar kullanımının oldukça yaygınlaşacağı öngörülmektedir.

4. HLA İLE İLGİLİ TEMEL KAVRAMLAR

4.1. HLA Nedir?

HLA, benzetim sistemleri geliştirilirken kullanılabilecek bir belirtimdir. Benzetim sistemi geliştiricilerine uygulamalarını tanımlayıp gerçekleştirebilecekleri bir uygulama çatısı sunar.

4.2. HLA Nelerden Oluşur?

HLA belirtimi üç kesimden oluşur:

- HLA kuralları
- Arayüz belirtimi (*Interface Specification*)
- Nesne modeli şablonu (*Object Model Template*)

4.3. HLA Kuralları

HLA kuralları, federasyonu oluşturan federeler arası etkileşimin düzgün bir biçimde gerçekleşmesini sağlamak için federasyonun ve federelerin görevlerini tanımlar. HLA kuralları, ilk beşi federe, geriye kalan beşi de federasyon kuralı olmak üzere toplam on kuraldan oluşur. Federelerin ve federasyonların HLA uyumlu sayılmaları için bu kurallara uymaları gerekir.

4.3.1. Federasyon Kuralları

- Federasyonun HLA nesne modeli şablonuna (OMT) uygun bir federasyon nesne modeli (FOM) olması gerekir.
- FOM'daki nesnelerin tanımları RTI tarafından değil, federeler tarafından yapılmalıdır.
- Federasyon çalışması esnasında tüm FOM bilgi değişimi RTI üzerinden olmalıdır.
- Federasyon çalışması esnasında federelerle RTI arası iletişim HLA arayüz belirtimine uygun olarak yapılmalıdır.

- Federasyon çalışması esnasında bir olgu niteliğinin sahipliği aynı anda sadece bir federede olmalıdır. Sahiplik, çalışma zamanında federeler arasında el değiştirebilir.

4.3.2. Federe Kuralları

- Federenin HLA nesne modeli şablonuna (OMT) göre düzenlenmiş bir benzetim nesne modeli (SOM) olmalıdır.
- Federeler SOM'da tanımlanan kurallara uyarak olgu niteliklerinin değerlerini günleyip / yansıtabilirler, etkileşim gönderip / alabilir.
- Federeler SOM'da tanımlanan kurallara uyarak federasyon çalışması esnasında olgu niteliklerinin sahipliğini kabul edebilir / devredebilir.
- Federeler SOM'da tanımlanan kurallara uyarak sahip oldukları olgu niteliklerinin değerlerini günleyerek federasyonun akışını değiştirebilir.
- Federeler yerel zamanlarını federasyon üyesi diğer federelerle veri değişimini yönetebilecek şekilde kontrol edebilir.

4.4. Arayüz Belirtimi ve RTI

Arayüz belirtimi altı yönetim alanından oluşur

- Federasyon yönetimi (*Federation Management*)
- Tanımlama yönetimi (*Declaration Management*)
- Nesne yönetimi (*Object Management*)
- Sahiplik yönetimi (*Ownership Management*)
- Veri dağıtım yönetimi (*Data Distribution Management*)
- Zaman yönetimi (*Time Management*)

Bu tez kapsamında bizim asıl çalışma alanımıza giren 3 yönetimden bahsedilecektir. Bunlar nesne yönetimi, veri dağıtım yönetimi ve zaman yönetimidir.

4.5. Nesne Modeli Şablonu (*Object Model Template*)

HLA'nın 1. kuralı her federasyonun bir federasyon nesne modelinin (FOM), 6. Kuralı da her federenin bir benzetim nesne modelinin (SOM) olması gerektiği şeklindedir. SOM ve FOM'un düzenlemelerini içine alan taslağa OMT denir. OMT benzetimler arası veri değişiminin ve federasyon üyeleri arasındaki eşgüdümün nasıl sağlanacağına dair işleyişi belirler. Benzetimler ve federasyonlar arasında haberleşmede kullanılacak nesne sınıflarıyla, bunlara ait nitelik ve etkileşimlerin hangi biçimde toplanacağı ve aktarılacağı da yine OMT ile tanımlanır.

4.5.1. Federasyon Nesne Modeli (FOM)

Federasyona katılan üyelerin aralarındaki haberleşme ortak bir protokol çerçevesi içinde olmalıdır. FOM federeler arasındaki veri değişiminin standart bir biçimde tanımlanmasını sağlar. FOM, benzetimin başında, federelerin bir arada çalışmasından sorumlu olan RTI'ya sunulur ve federeler arası iletişim FOM esaslarına dayalı olarak sürdürülür.

4.5.2. Benzetim Nesne Modeli (SOM)

Bir federenin federasyona katılımının uygunluğu SOM aracılığıyla belirlenir. Bir federe federasyona katılırken, ortak çalışma esnasında kendisinden istenebilecek nesne, nitelik ve etkileşimleri sağlayabilmelidir. SOM, federenin federasyona sağlayacağı ve federasyondan talep edeceği verilerin özelliklerinin toplandığı doküman olarak da tanımlanabilir.

4.5.3. Nesne Modeli Şablonu Bileşenleri

HLA nesne modelleri, bir grup nesne sınıfları ve onların nitelikleri ile etkileşimler ve parametrelerinden oluşurlar.

OMT aşağıda sayılan ondört bileşenden oluşur:

→ Nesne modeli kimlik çizelgesi(*Object Model Idendification Table*)

→ Nesne sınıfı yapı çizelgesi (*Object Class Structure Table*)

→ Etkileşim sınıfı yapı çizelgesi (*Interaction Class Structure Table*)

→ Nitelik çizelgesi (*Attribute Table*)

→ Parametre çizelgesi (*Parameter Table*)

→ Boyut çizelgesi (*Dimension Table*)

→ Zaman gösterim çizelgesi (*Time Representation Table*)

→ Kullanıcı tanımlı etiketler çizelgesi (*User-supplied Tag Table*)

→ Zamanuyumlama çizelgesi(*Synchronization Table*)

→ Nakil türü çizelgesi (*Transportation Type Table*)

→ Anahtarlar çizelgesi (*Switches Table*)

→ Veritürü çizelgesi (*Datatype Table*)

→ Notlar çizelgesi (*Notes Table*)

→ FOM/SOM sözlüğü(*FOM/SOM lexicon*)

OMT belirtimine göre, federasyonlar ve federeler tanımlanırken tüm OMT bileşenleri sağlanmalıdır. Bununla birlikte, ilgili çizelgeler boş veya alana özel bilgi içermiyor olabilir. Örneğin federasyonlar, federeleri arasında etkileşime izin vermesine rağmen, bazı federelerin etkileşimleri olmayabilir. Bu durumda federe için oluşturulacak SOM'da etkileşim çizelgesinde sadece HLA tarafından ihtiyaç duyulan etkileşim sınıfı bulunur ve parametre çizelgesi boş olur. Çoğu zaman federelerin federasyon çapında kullanılan nesne nitelikleri olur, fakat eğer federasyon içindeki tüm veri değişimi etkileşimler ile gerçekleşiyorsa, bu durumda nesne sınıf yapı çizelgesi ve nitelik çizelgesi sadece HLA'nın ihtiyaç duyduğu bilgiyi içerir [8].

4.6. Nesne Yönetimi (*Object Management*)

Nesne Yönetimi, genel anlamda nesne ve etkileşimlerin (*interactions*) yaratılması, günlenmesi ve silinmesini sağlayan bir yönetimdir. Daha ayrıntılı değinecek olursak;

- Federe bir nesne yaratır ve nesneyi federasyona kaydeder.

(*Register Object Instance*)

- Yeni bir nesne federasyona katıldığında, RTI bunu bildirir.

(*Discover Object Instance*)

- Federe zaman adımı sonunda veri günlemesi yapar.

(*Update Attribute Values*)

- Federe o zaman adımında diğer federelerin yaptığı günlemeleri alır.

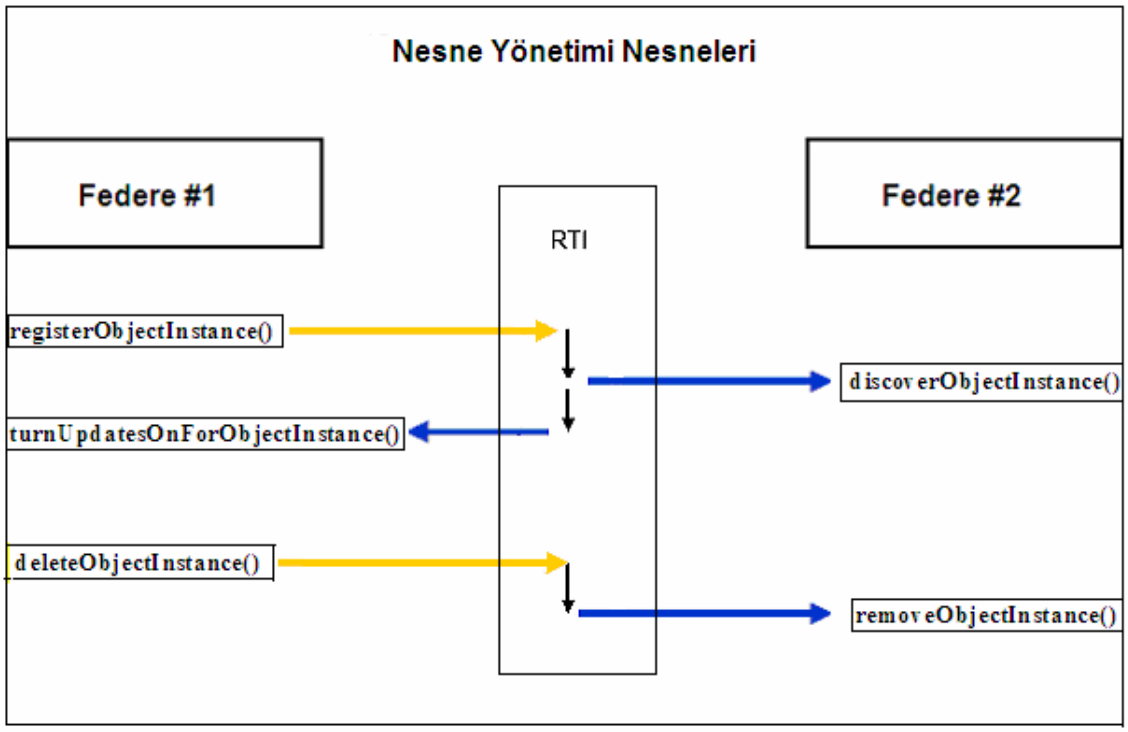
(*Reflect Attribute Values*)

- Federe arasıra diğer federelere ileti gönderebilir (*Send Interaction*) ve arasıra diğer federelerden ileti alabilir (*Receive Interaction*).

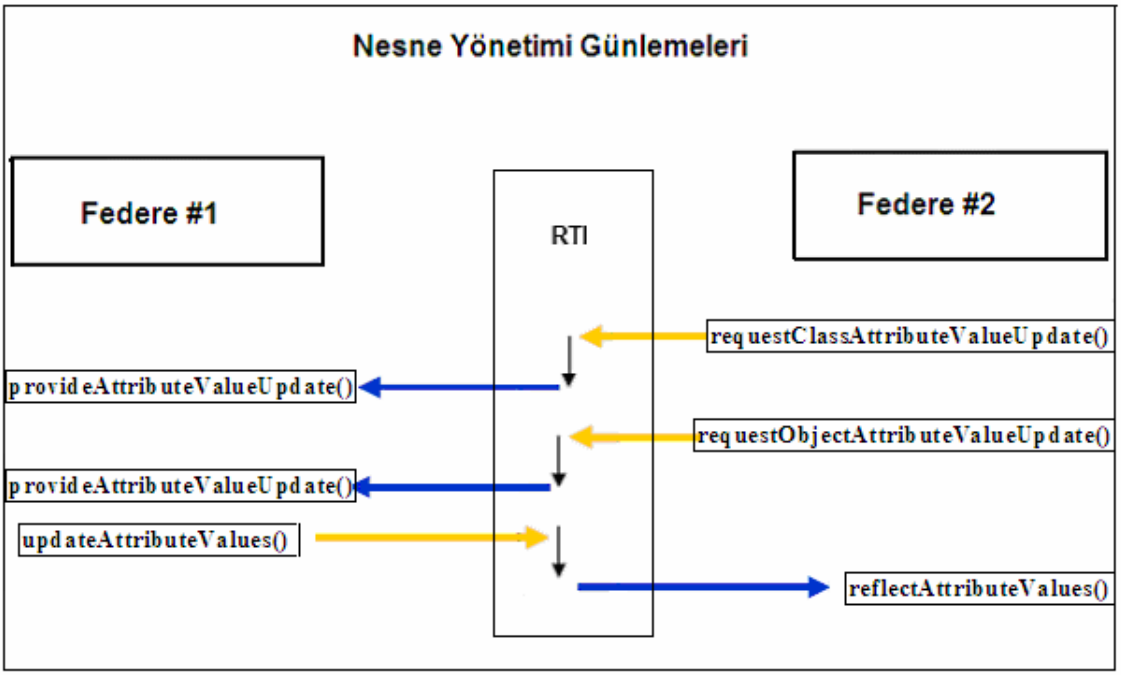
- Federenin verileriyle ilgilenen başka federe yoksa RTI federeyi uyarır.

(*Turn Updates off for Object Instance*)

Şekil 4.1 ve Şekil 4.2'de nesne yönetimi nesneleri ve nesne yönetimi günlemeleri ayrıntılı olarak gösterilmiştir.



Şekil 4.1. Nesne Yönetimi Nesneleri



Şekil 4.2. Nesne Yönetimi Günlemeleri

4.7. Veri Dağıtımı Yönetimi (*Data Distribution Management*)

Veri dağıtımı yönetimi, federeler arasındaki ilgi alanlarının yayınlanması ve bunlara üye olunması işlemlerini düzenlerken, aynı zamanda federelerin ilgilenmedikleri verilerin gereksiz yere iletilmesini ve alınmasını engeller. Veri dağıtımı yönetiminin sağladığı işlevler Çizelge 4.1'de görülmektedir.

Çizelge 4.1. Veri Dağıtımı Yönetimi

İŞLEV
Bölge Oluştur (*Create Region*)
Bölgeyi Değiştir (*Modify Region*)
Bölgeyi Sil (*Delete Region*)
Varlığı Bölgeye Kaydet (*Register Entity w/Region*)
Denetim Güncellemeleri (*Control Updates*)

Veri iletiminde en sık kullanılan veri iletim yöntemlerinden biri bütün federelere iletiyi gönderme (*broadcast*) yöntemidir. Diğeri ise belirli federelere veya gruplara (*multicast*) veri iletme yöntemidir. Bütün federelere ileti gönderme yönteminde ileti bütün federelere iletileceğinden hem ağ gereksiz yere bilgiyi aktarmış olacaktır hem de aktarılan bilginin benzetim bileşenleri tarafından filtrelenmesi gerekecektir. Bu durumda kaynaklar boş yere meşgul edilmektedir.

İletilmek istenen verilerin yalnızca, o veri ile ilgilenen federelere iletilmesi birincil amaçtır. Bu amaç kapsamında, hangi nesnelere ait bilgilerle ilgilenildiğinin belirtilmesi için bölge tanımları kullanılır. Her federe hangi bölgelerdeki güncellemelerle ilgilendiğini ve hangi bölgeleri güncellediği bilgilerini belirtir [5]. Güncellemelerle ilgilenme işine abone olma (*subscription)*, güncellenecek bölgenin bildirilmesi işine ise yayınlama (*publication*) adı verilir.

4.8. Yayınlama (*Publish*) ve Abone Olma (*Subscribe*) Mekanizması

Federasyonlar arasındaki iletişim, iletilerle sağlanır. Bu iletiler hem günlemeleri, hem de etkileşimleri kapsar. Federeler, federasyona hangi verileri yayınlayacaklarını ve hangi verilerle ilgilendiklerini bildirir. Eğer federasyon oluşturulmayacaksa federeler hangi verileri yayınlayıp hangi verilere abone olacaklarını kendileri bildirirler. Bu bildirime göre diğer federelerle iletişime geçilir ve gerekli bilgi alış-verişi sağlanır.

→ Nesne Yayınlama / Abone Olma

→ Publish Object Class

→ Subscribe Object Class

→ Nesne Kaydı

→ Register Object Instance

→ Discover Object Instance

→ Olgu Kaydının Denetlenmesi

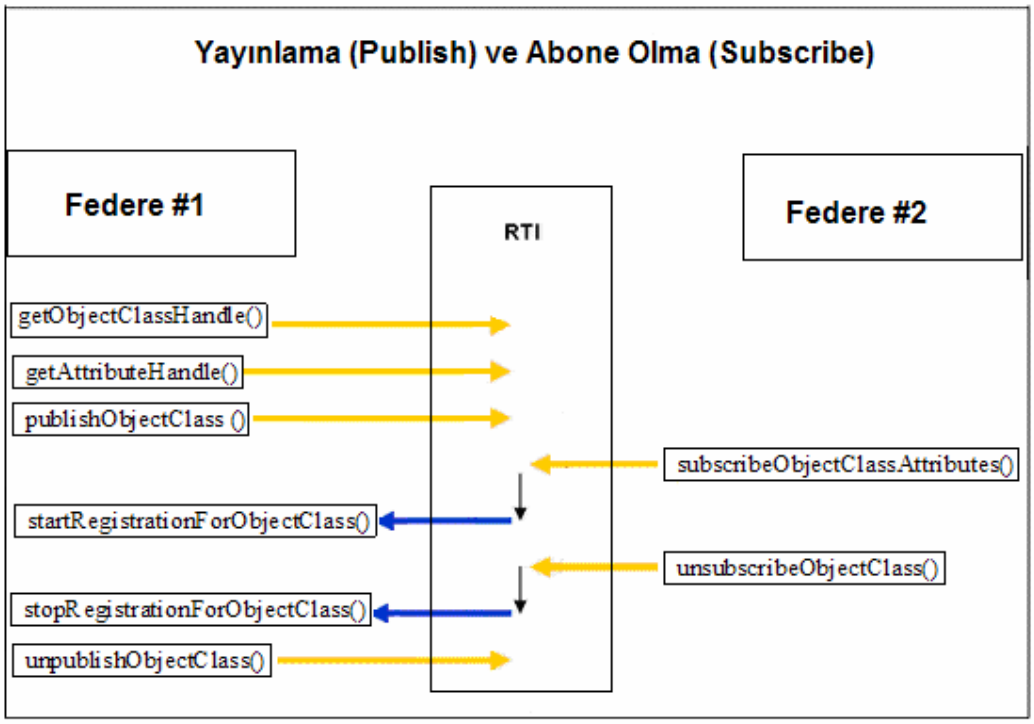
→ Enable / Disable Class Relevance Advisory Switch

→ Start / Stop Registration for Object Class

→ Etkileşimler Yayınlama / Abone Olma

→ Publish Interaction Class

→ Subscribe Interaction Class



Şekil 4.3. Yayınlama ve Abone Olma Mekanizması

4.8.1. Veri Dağıtımı Yönetimi Algoritmaları

Veri dağıtımı yönetimi, federelerin yayınlayacağı ve alacağı verilerin istenmeyen verilerden arındırılıp sadece ilgilenen verilerin aktarımını sağlayarak, sistemin mesaj trafiği yükünü ve ayrıca federelerin veri ayıklama yükünü hafifleten servis birimidir. Veri dağıtımı yönetiminde kullanılacak olan algoritma, veri dağıtımı yönetiminin başarımını belirleyen en önemli unsurlardandır. Aşağıda veri dağıtımı algoritmalarından önemli bir kaçı anlatılmaktadır. Bu algoritmaların açıklamalarına geçmeden önce bazı tanımların bilinmesi gerekmektedir:

İşlem uzayı (*routing space*) : Federelerin veri alışverişinde bulundukları çok boyutlu koordinat sistemidir.

Boyut (*dimension*) : Koordinat sisteminin bir paçasıdır. Örneğin üç boyutlu uzayda enlem (*longitude*), boylam (*latitude*), yükseklik (*altitude*) kavramlarının her biri bir boyutu ifade etmektedir. İşlem uzayı ve boyutlar FED (*Federation Execution Data*)'de tanımlanmaktadır.

Aralık (*range*) : İşlem uzayında, bir boyutun alt ve üst sınırı ile tanımlanan kısmıdır.

Bölge (*region*) : İşlem uzayı koordinat sisteminin tüm boyutlarında aralığı tanımlanmış olan alandır.

Yayınlama Bölgesi (*Update Region*) : Bir federenin ilgilendiği nesnelerin güncellemelerini yaptığı bölgedir.

Abonelik Bölgesi (*Subscription Region*) : Bir federenin almak istediği verilerin güncellemesinin yapıldığı bölgedir.

Kesişim Bölgesi (*Intersection Region*) : Yayınlama ve abonelik bölgelerinin kesişim kümesidir.

Sınıf (*class*) : Ortak nitelik, ilişki ve mantık özellikleri taşıyan varlıklar kümesidir.

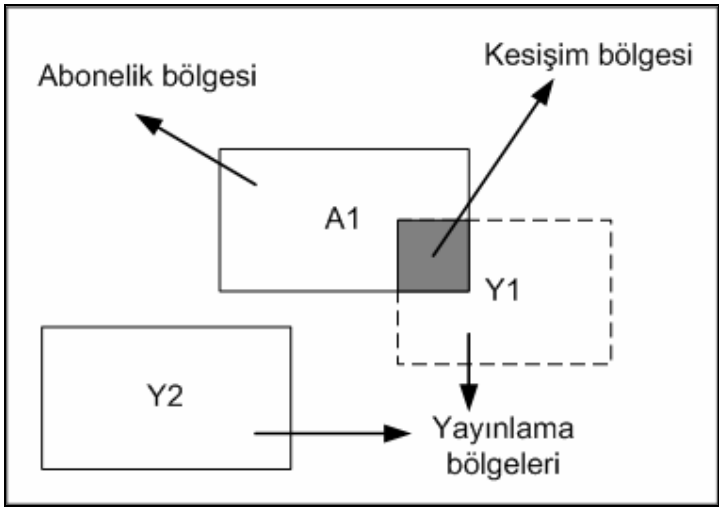
4.8.2. Sınıf Tabanlı (*Class-based*) Veri Dağıtımı Yönetimi

Sınıf tabanlı veri dağıtımı yönetiminde, federelerin abone oldukları yapılar sınıflardır. Dolayısıyla bir federe abone olduğu sınıfa ait tüm nesnelerin güncelleme bilgilerini alır. Bu da gerekli olmadığı halde bazı verilerin alınması anlamına gelmektedir. Gereksiz bilgilerin federelere taşınması hem mesaj trafiğini, hem de gerekli verileri tüm verilerin içerisinden ayırt edecek olan federenin işlem yükünü artırmaktadır. Bu nedenlerle verimliliği düşük bir veri dağıtımı yönetimi algoritmasıdır.

4.8.3. Bölge Tabanlı (*Region-based*) Veri Dağıtımı Yönetimi

Bölge tabanlı veri dağıtımı yönetiminde, federeler yayınlama yapacakları bölgeleri ve abone oldukları bölgeleri belirlerler. Bir federenin yayınlama bölgesiyle bir başka federenin abonelik bölgesi kesişirse, yayınlama yapan federenin yayınladığı tüm güncellemeler, o federenin yayınlama bölgesiyle kesişen federeye gönderilir. Bu yönetim biçiminde, veri alımında bulunan federe, kesişim bölgesinin dışında kalan verileri de almak zorunda kalmaktadır. Şekil 4.4 yayınlama, abonelik ve

kesişim bölgelerini ve maruz kalınan gereksiz veri alımını görsel olarak daha açık bir şekilde göstermektedir.



Şekil 4.4. Bölge Tabanlı Veri Dağıtımı

Şekil 4.4'deki A1 bölgesi bir federenin abonelik bölgesini, Y1 ve Y2 bölgeleri de başka federelerin yayınlama bölgelerini göstermektedir. A1 ve Y2 bölgeleri arasında bir kesişme olmadığı için Y2 bölgesine ait veriler hiç bir şekilde A1 bölgesine abone olan federeye gönderilmeyecektir.

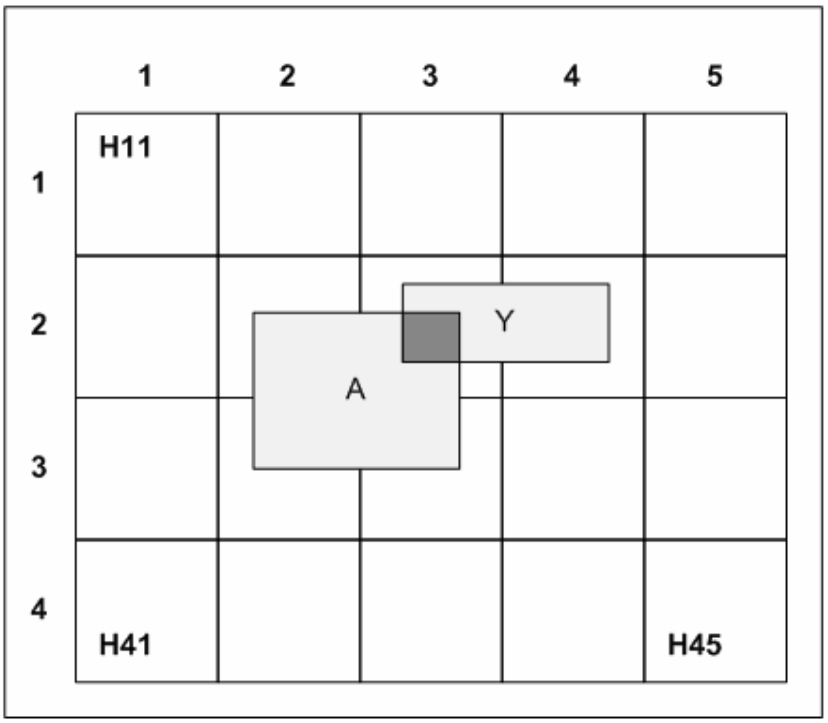
Fakat A1 ve Y1 bölgeleri arasında bir kesişim vardır, ve kesişim bölgesi dışındaki bölge de dahil olmak üzere, tüm güncellemeler A1 bölgesine abone olan federeye gönderilecektir. Bu durumda A1 bölgesine abone olan federe Y1 bölgesindeki A1 ve Y1 bölgelerinin kesişimi dışında kalan ihtiyaç duymadığı güncellemeleri de alacaktır.

4.8.4. Izgara Tabanlı (*Grid-based*) Veri Dağıtımı Yönetimi

Izgara tabanlı veri dağıtımı yönetimi, bölge tabanlı veri dağıtımı yönetimindeki aktarılan gereksiz veri miktarını azaltmak için geliştirilmiştir.

Bu yöntemde işlem uzayı ızgara şeklinde parçalara bölünmüştür. Her bir hücre için, o hücreye isabet eden yayınlama bölgeleri ve abonelik bölgeleri iki ayrı liste halinde tutulur. Eğer nesneler hareket halindeyse ya da yayınlama ve abonelik bölgeleri zaman içerisinde değişiyorsa bu listeler de zamana bağlı olarak

güncellenmektedir. Kesişim bölgeleri hangi hücreye denk geldiyse, abone olan federeye yayınlama bölgesinin o hücre kısmındaki veri güncellemeleri aktarılır. Bu sayede bölge tabanlı veri dağıtımı yönetimine oranla daha az gereksiz veri aktarımı sağlanmış olur. Şekil 4.5'de ızgara tabanlı veri dağıtımı yönetimi bir örnekle açıklanmaktadır.



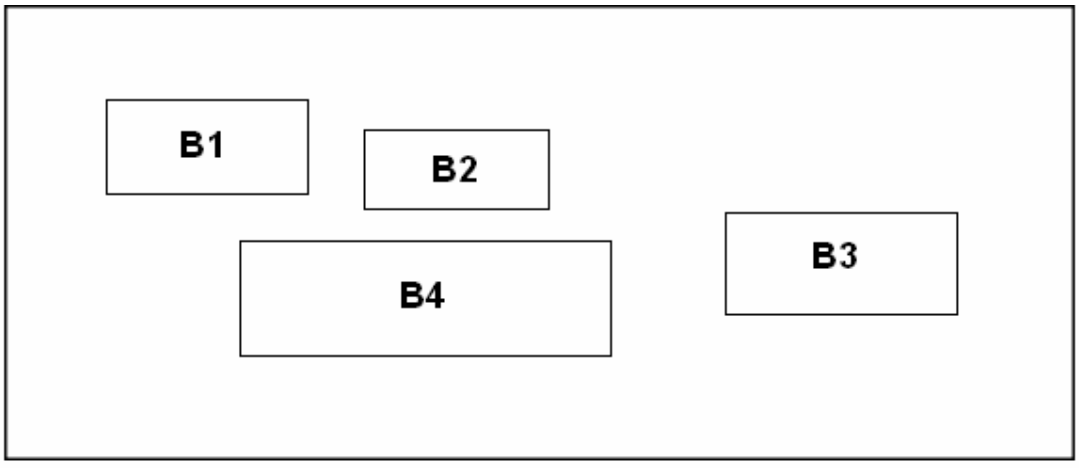
Şekil 4.5. Izgara Tabanlı Veri Dağıtımı

Şekil 4.5'de iki boyutlu bir işlem uzayı, 4x5'lik bir ızgara biçiminde hücrelere bölünmüştür. A bölgesi bir federenin abonelik bölgesini, Y bölgesi de başka bir federenin yayınlama bölgesini göstermektedir. Kesişim bölgesi H23 hücresi içerisindedir. Bu durumda A bölgesine abone olan federe Y bölgesinin H23 hücresine giren veri güncellemelerini alacaktır. Bölge tabanlı veri dağıtımı yönetimi uygulanmış olsaydı A bölgesine abone olan federe H24 hücresindeki verileri de alacaktı. Izgara tabanlı veri dağıtımı yönetimiyle bölge tabanlı veri dağıtımı yönetimine göre daha az gereksiz veri akışı sağlanmakta, ancak kesişim bölgesi dışında kalmasına rağmen kesişimin gerçekleştiği hücrede (H23) bulunan verilerin aktarılması engellenmemektedir.

Hücre boyutları ne kadar büyükse gereksiz yere aktarılan veri miktarı da o kadar fazla olacaktır. Bu sebeple gereksiz veri aktarımının daha da aza indirgenebilmesi için hücre boyutlarının daha da küçültülmesi akla gelen ilk yöntemdir. Ancak bu uygulama sonucunda hücre sayısının artmasıyla bu hücrelerin yönetilmesinin maliyeti artacaktır. Uygun hücre büyüklüğü ve sayısının seçilmesi yapılacak maliyet-başarım analizi yapılarak seçilebilir.

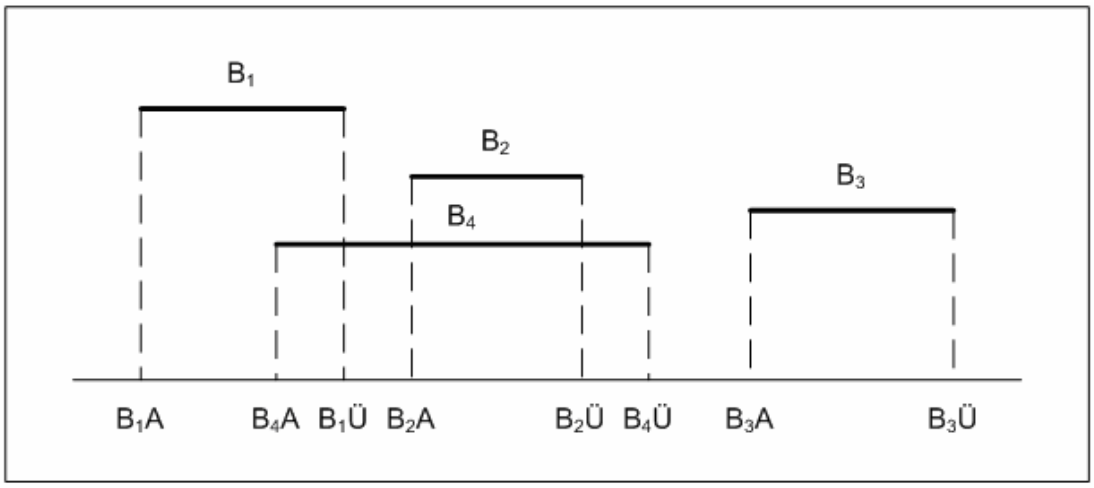
4.8.5. Sıra Tabanlı (*Sort-based*) Veri Dağıtımı Yönetimi

Sıra tabanlı veri dağıtımı yönetiminde, abonelik ve yayınlama bölgelerinin, içinde bulundukları işlem uzayı koordinat sisteminin eksenlerine izdüşümleri alınır. Kesişimler boyutlar bazında incelenmektedir. Aşağıdaki örnek iki boyutlu uzayda algoritmanın nasıl işlediğini göstermektedir. Şekil 4.6 bir işlem uzayındaki çeşitli yayınlama ve abonelik bölgelerini göstermektedir.

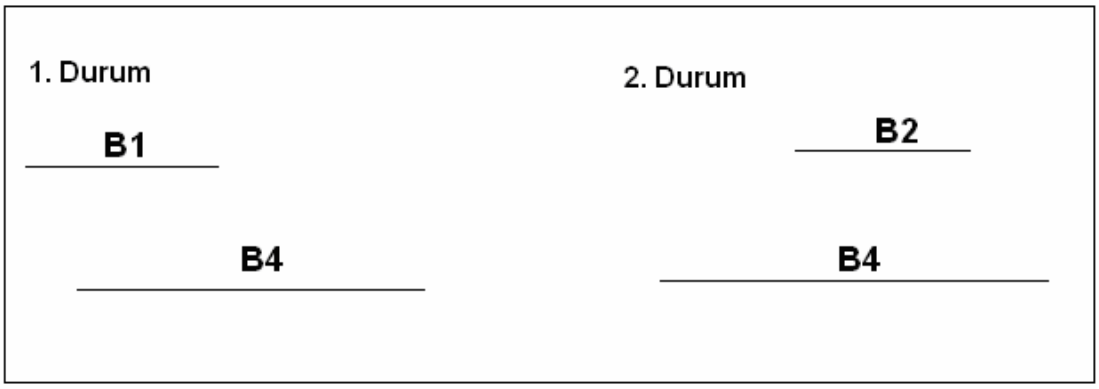


Şekil 4.6. Sıra tabanlı veri dağıtımı yönetimi için temel senaryo

Şekil 4.7'de ise Şekil 4.6'da gösterilen bölgelerin x ekseni üzerindeki izdüşümleri görülmektedir. İzdüşümler alındığında iki durumla karşılaşmak olasıdır. Bunlardan birincisi, izdüşümlerin bir bölümünün kesişmesi, ikincisi ise izdüşümlerden birinin diğerini kapsamasıdır. (Şekil 4.8)



Şekil 4.7. Şekil 4.6'daki bölgelerin x ekseni üzerine izdüşümleri



Şekil 4.8. İzdüşümler alındıktan sonra gözlemlenecek olası durumlar

Bölgelerin x ekseni üzerine izdüşümleri alındıktan sonra izdüşümlerin alt ve üst sınırları işaretlenir. Burada B1A, B1 bölgesinin altı sınırını B1Ü de üst sınırını göstermektedir. X ekseni soldan sağa doğusu tarandığında izdüşümlerin alt ve üst sınırları sırasıyla karşılaşılacaktır. Örneğimizde ilk olarak B1 bölgesinin alt sınırıyla karşılaşılacaktır. Daha sonra ise B4 bölgesinin alt sınırıyla karşılaşılacaktır. B1 bölgesinin üst sınırıyla karşılaşılmadan B4 bölgesinin alt sınırıyla karşılaşmak B1 bölgesiyle B4 bölgesinin kesiştiğini göstermektedir. B4 bölgesinin üst sınırıyla karşılaşılmadan B1 bölgesinin üst sınırıyla karşılaşılması da kesişimin 1. türden bir kesişim olduğunu gösterir. B4 ve B2 bölgelerinde ise sırasıyla B4 bölgesinin alt sınırı, B2 bölgesinin alt sınırı, B2 bölgesinin üst sınırı ve en son olarak da B4 bölgesinin üst sınırıyla karşılaşılmaktadır. Bu durumda B4 bölgesinin B2 bölgesini

kapsadığını gösterir. B3 bölgesinin alt sınırıyla karşılaşıldıktan sonra başka bir bölgenin ne alt ne de üst sınırıyla karşılaşılmadan B3 bölgesinin üst sınırıyla karşılaşılması da B3 bölgesinin hiçbir bölgeyle kesişmediğinin göstergesidir.

Yukarıda tek bir boyut için anlatılan algoritma koordinat sisteminin diğer boyutları için de uygulanır ve tüm boyutlarda kesişimi olan bölgeler, aralarında veri transferi olacak olan bölgeler olarak tanımlanırlar. Bu yöntem hücre boyutu ve sayısından bağımsız olarak kesişen bölgeleri hızlı bir biçimde belirleyebilmektedir.

4.9. Zaman Yönetimi (*Time Management*)

Dağıtılmış benzetim sistemlerinde en önemli yönetimlerden biri zaman yönetimidir. Farklı işlemci hızlarına sahip bilgisayarlar üzerinde koşan dağıtılmış benzetim bileşenlerinin ortak bir zaman kavramı oluşturması ve iletişim amaçlı gönderilen iletilerin ağ üzerindeki olası yoğunluk ve problemlerden bağımsız olarak istenen sırada gönderilebilmesi gerekmektedir [6].

Zaman yönetimiyle ilgili bazı önemli kavramlar bilinmelidir. Aşağıda bunlarla ilgili tanımlara yer verilmiştir.

Mantıksal Zaman (*Logical Time*): Bir federasyonda, her bir federenin kendine ait bir yerel zamanı vardır. Bu yerel zamana mantıksal zaman denir. Federelerin benzetime katıldıkları ilk anda mantıksal zamanları sıfır olarak varsayılır.

Zaman-Damgası Sıralı İletiler (*Time Stamped Order, TSO*): Belirli bir zaman damgasına göre sıralı olarak işlenen iletilere zaman-damgası sıralı iletiler denir.

İletilerin tesliminin zaman sıralı olması için, tüm federelerin zamanları birbirleriyle uyumlu olmalıdır (synchronized). Bu nedenle zaman-damgalı ileti alan federeler için zaman ilerletme açısından kısıtlar vardır. Zaman-damgası sıralı iletileri alan federelere **zaman-kısıtlı (*time constrained*)** federe denir.

Zamanı yöneten servislerin zaman-kısıtlı bir federeyi T zamanına ilerletmesi için, veriyi yayınlayan federelerin hepsi T zamanına kadar olan iletilerini göndermeyi bitirdiklerinde, RTI bundan haberdar edilmelidir. Bu nedenle zaman-damgalı ileti gönderen federeler zamanı düzenlemektedir. Zaman-damgası sıralı iletileri gönderen federelere **zaman-düzenleyen (*time regulating*)** federe denir [8].

- Bir federe TSO iletilerini zaman-damgalı sırada almak için, kendisini ***Time Constrained*** olarak bildirmelidir. ***(Enable Time Constrained → Time Constrained Enabled)***

- Bir federe TSO iletilerini gönderebilmek için, kendisini ***Time Regulating*** olarak bildirmelidir.

  (Enable Time Regulation → Time Regulation Enabled)

- Genel olarak bir federe;

  → Zaman-düzenleyen,

  → Zaman-kısıtlı,

  → Hem zaman düzenleyen hem zaman-kısıtlı,

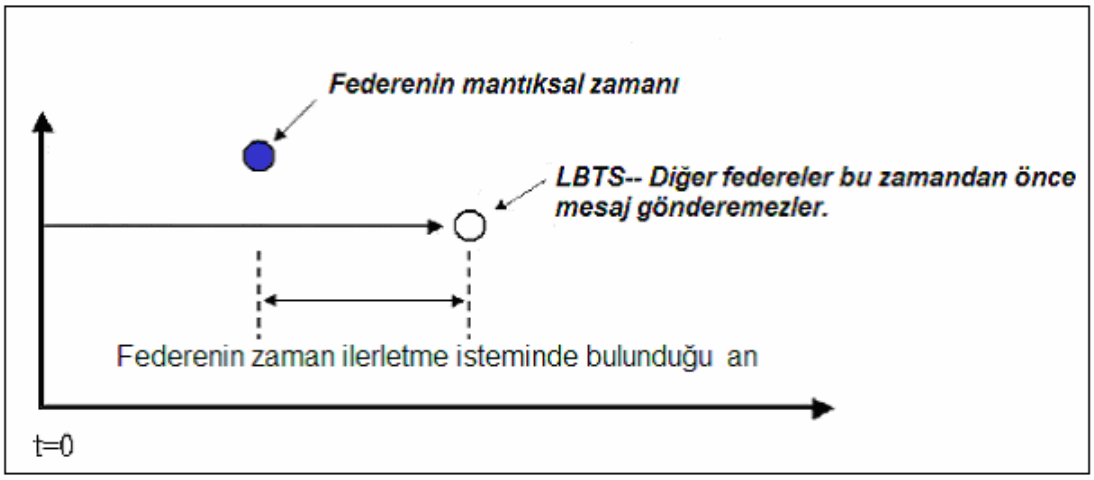
  → Ne zaman düzenleyen ne zaman-kısıtlı,

  → Zaman yönetimi servisini kullanmayan olur.

- Varsayılan olarak, federeler ne zaman-kısıtlıdır ne de zaman düzenleyendir.

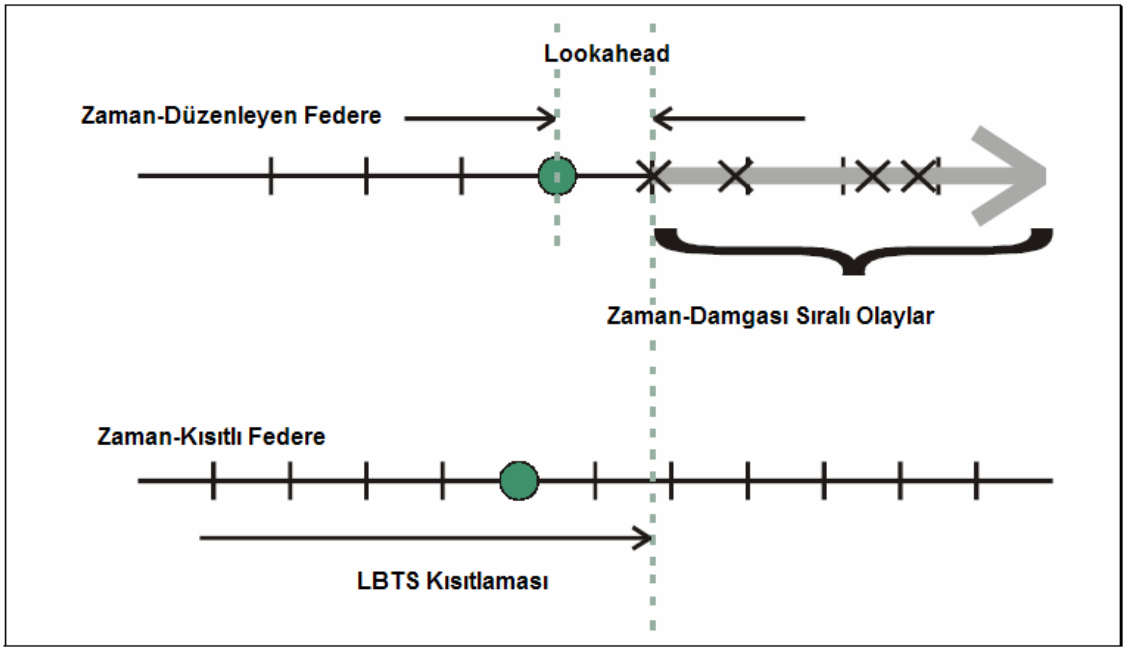
- Bir federasyonda birden çok çeşitte yukarıda sayılan federelerden bulunabilir.

Zaman Damgası Alt Sınırı (*Lower Bound Time Stamp, LBTS*): Zaman kısıtlı bir federe, zamanını ilerletebilmek için, geçmişten (mantıksal zamanından küçük değerli zaman damgalı) bir ileti gelmeyeceğini garanti etmelidir. Bu garanti RTI tarafından verilmektedir. LBTS değeri, diğer bütün zaman-düzenleyen federelerin en erken ileti gönderecekleri zamana bakılarak düzenlenir. Şekil 4.9'da zaman kısıtlı federenin mantıksal zamanı ve LBTS değeri görülmektedir.



Şekil 4.9. Zaman-Kısıtlı Federenin Mantıksal Zamanı ve LBTS Değeri

Lookahead: Zaman-düzenleyen bir federe için, göndereceği TSO iletilerinin zaman-damgalarının en küçük değerine "lookahead" değeri denir. Her zaman-düzenleyen federenin lookahead değeri vardır ve federenin göndereceği bir TSO iletisinin zaman-damgası, "*mantıksal zaman + lookahead"* değerinden küçük olamaz.

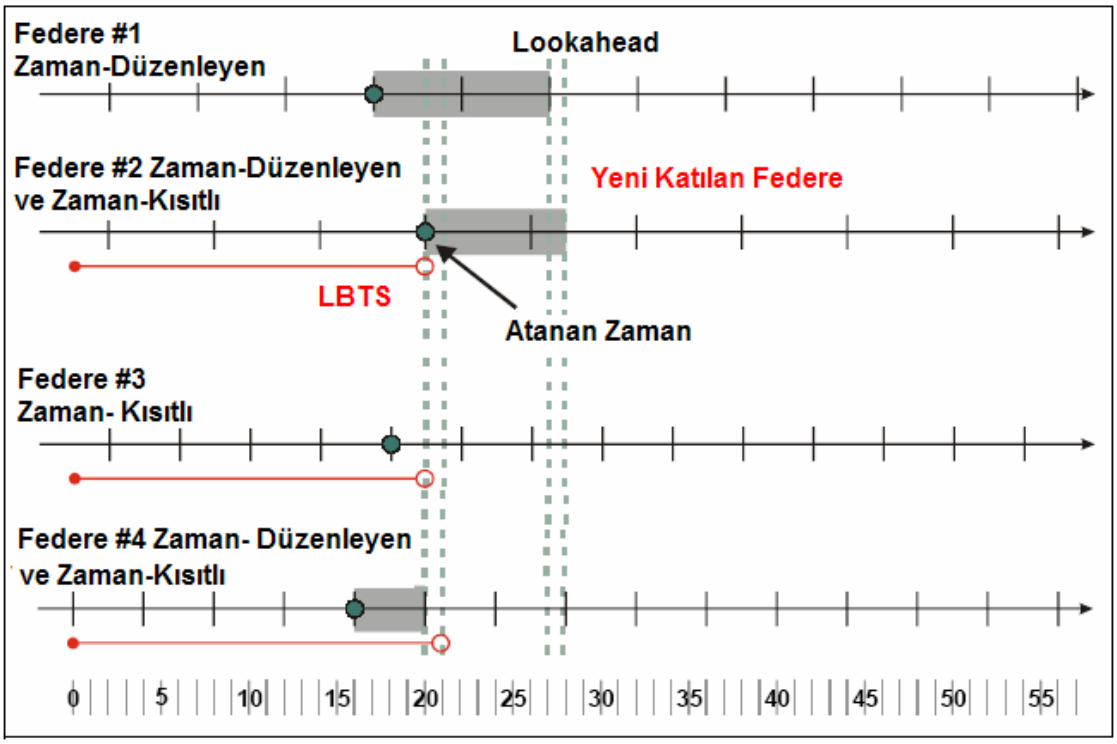


Şekil 4.10. Zaman-Düzenleyen Federenin Lookahead Değeri

Şekil 4.10'da görüldüğü gibi, zaman-düzenleyen federe, lookahead değeri süresince herhangi bir ileti gönderemez. Zaman-kısıtlı federe de LBTS kısıtlaması sürecinden sonra zaman-düzenleyen federenin gönderdiği iletiyi alabilir.

Bir federe hem zaman-düzenleyen hem de zaman-kısıtlı (*regulating and constrained*) olabilir. Bu durumda federe, hem ilerleyeceği zaman olarak LBTS ile kısıtlıdır hem de göndereceği iletilerin "mantıksal zaman + lookahead" değerinden önce gönderilmeyeceğini garanti etmek durumundadır.

Federasyona yeni/geç katılacak federeler için durum biraz daha farklıdır. Bu durumda öncelikle federenin zaman yönetimi servisi açısından ne tür bir federe olduğu bilinmelidir. Örneğin; federenin hem zaman-kısıtlı hem de zaman-düzenleyen olduğu varsayılsın. Bu federe federasyona katıldığı anda mantıksal zaman olarak, diğer bütün federelerden en küçük LBTS değerine sahip olanın zamanı atanır. Bu durum Şekil 4.11'de daha detaylı gösterilmiştir.



Şekil 4.11. Federasyona Yeni Katılan Federenin LBTS Değeri

Şekil 4.11'de görüldüğü gibi federasyona sonradan katılan federenin mantıksal zamanı en küçük LBTS değeri olan "20"ye atanmıştır.

5. BİLİMSEL BENZETİM UYGULAMALARI

Birçok bilim dalında ve mühendislik alanlarında koşut programlama kullanılarak benzetim geliştirmeye yönelik uygulamalar yapılmaktadır. Özellikle üniversitelerde, koşut programlama kullanılarak gerçekleştirilen önemli bazı benzetim uygulamaları şunlardır:

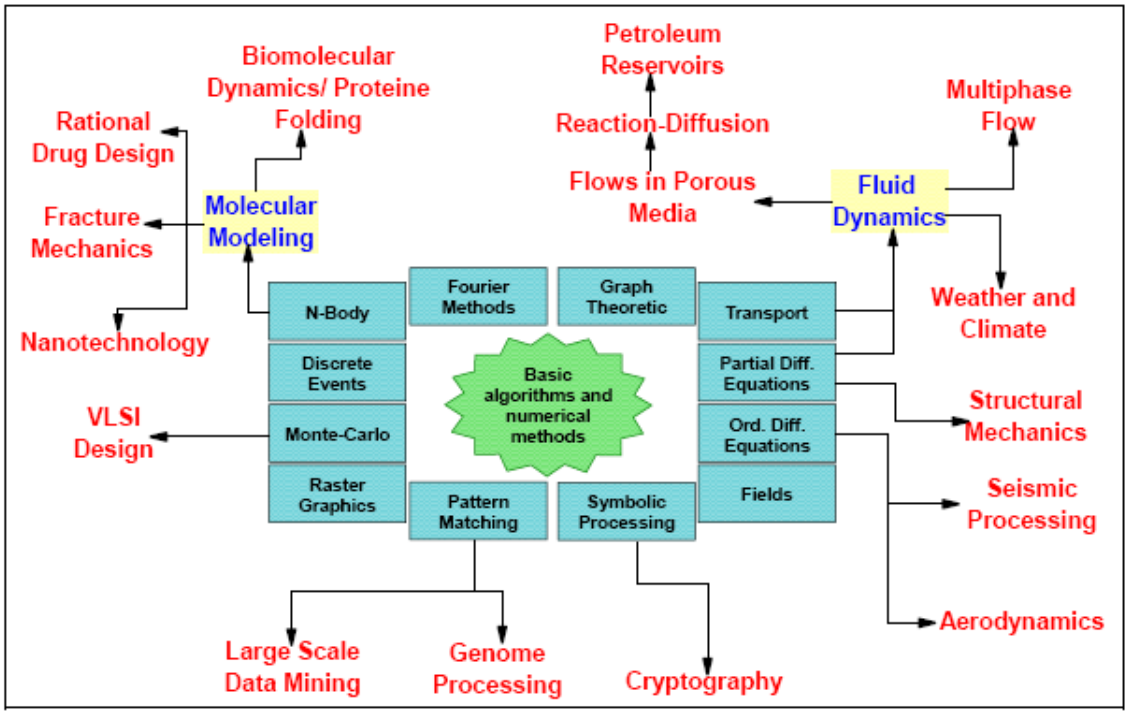
- **Moleküler Dinamik (*Molecular Dynamics*)**
    - Biyomoleküler Dinamik (*Biomolecular Dynamics*)
    - Nanoteknoloji (*Nanotechnology*)
- Kırılma Mekaniği (*Fracture Mechanics*)

Akışkan Dinamiği (*Fluid Dynamics*)

- **Hava ve İklim Modelleme (*Weather and Climate Modeling)***
- Petrol Rezervasyonu (*Petroleum Reservoirs*)

- Hava Trafiği Kontrolü (*Air Traffic Control*)
- Şifreleme (*Cryptography*)
- Yapay Sinir Ağları (*Neural Networks)*
    - **Akıllı Ajan (*Intelligent Agent*)**
- Sinyal İşleme (*Signal Processing)*
- Quantum Chemistry (*Kuantum Kimyası)*
- Populasyon Genetiği (*Population Genetics)*
- Kimyasal Reaktörler (*Chemical Reactors)*
- Yapısal Mekanik (*Structural Mechanics*)

Şekil 5.1'de koşut ortamda çalışabilecek benzetim sistemleri daha detaylı gösterilmiştir.



Şekil 5.1. Koşut Ortamda Çalışabilecek Bazı Benzetim Sistemleri [28]

Yukarıda bahsi geçen bilimsel benzetim uygulamalarından Hava ve İklim Modelleme, Molekül Dinamiği ve Akıllı Ajan konuları tez kapsamında incelenmek üzere şu sebeplerden dolayı seçilmiştir:

Hava tahmin modellemesi ve molekül dinamiği uygulamalarında benzetimi yapılacak alan hücrelere bölünerek gerçekleştirilmektedir. Hava tahmin benzetiminde hücreler bir coğrafi alanı temsil ederken, hücreler üzerindeki hava durumunu etkileyen faktörler daha çok durağan (*static*) bir etkileşim sergilemektedir. Gerek sistemin durağan bir yapı sergilemesi gerekse günlük hayatımızı etkileyen neredeyse her gün ihtiyaç duyduğumuz bir uygulama olması ve sistemin koşut ortamda çalışmaya müsait olması hava tahmin modellemesinin örnek uygulama olarak seçilmesinde etkili olmuştur. Molekül dinamiği uygulamasının hava tahmin modellemesinden farkı ise hücreler içerisindeki parçacıkların hareketli oluşu diğer bir deyişle sistemin dinamik bir etkileşim sergilemesidir. Son olarak ise yapı olarak tamamen bu iki uygulamadan bağımsız

olan ve farklı işlevlere sahip uygulamaların bir arada çalışmasını gerektiren akıllı ajan uygulaması tez metninde incelemek üzere seçilmiştir.

6. HAVA TAHMİN BENZETİMİ

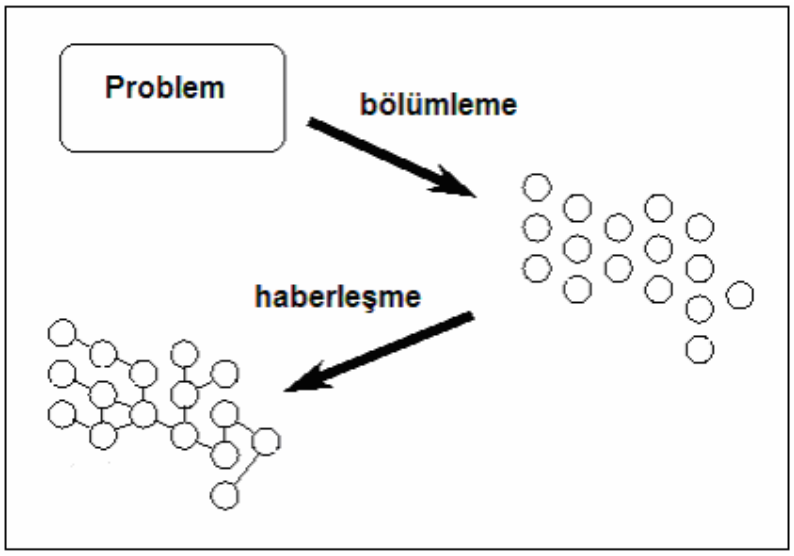
Hava tahmin benzetimleri, günümüzde insan yaşamındaki birçok eylemin, hava koşullarından etkilenmesi nedeniyle, gelecek planlamalarının daha iyi yapılabilmesi için önem kazanmış bilimsel uygulamalardır. En yaygın olarak kullanılan model MM5 hava tahmin modelidir.

MM5 hava tahmin modeli, Amerika'da bulunan NCAR (*National Centre for Atmospheric Research*) tarafından geliştirilen, şu anda birçok modern meteoroloji örgütü ve üniversiteler tarafından tercih edilen bir modeldir. Özellikle tercih edilmesinin nedeni; atmosferde meydana gelen düşey hareketleri 10 km ve altındaki çözünürlüklerde daha sağlıklı hesaplayabilmesidir. Bu nedenle kısa vadeli ve yerel tahminlerde oldukça başarılı bir modeldir. MM5 modeli sayesinde, daha küçük ölçekte alanlar için sağlıklı ve sık zaman aralıklarında ayrıntılı hava tahmini yapmak mümkün olmaktadır [4].

6.1. Hava Tahmin Benzetiminin Koşut Programlama ile Modellenmesi

Bu bölümde hava tahmin modelinin HLA dışında koşut olarak nasıl modellendiği incelenecektir.

Hava tahmin probleminin birçok çalışmada koşut olarak modellendiği görülmüştür. Bu çalışmalarda ortak olarak kullanılan temel koşut programlama adımlarından iki tanesi bölümleme (*partitioning*) ve haberleşmedir (*communication*) [9]. Bu adımların hava tahmin uygulamasının modellenmesinde nasıl kullanıldıkları bir sonraki bölümde ayrıntılı olarak anlatılmaktadır.



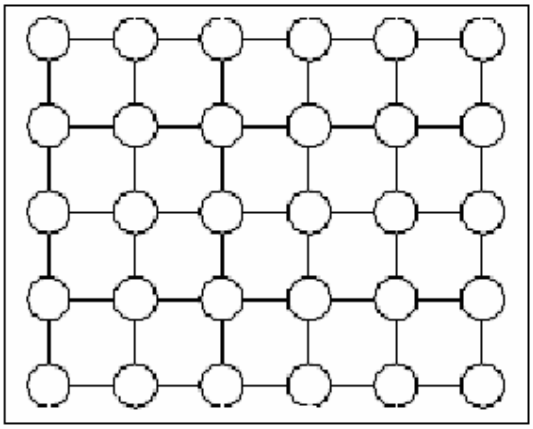
Şekil 6.1. Bölümleme ve Haberleşme

6.1.1. Bölümleme (*Partitioning*)

Bir programı koşutlaştırmak için yapılacak ilk adımlardan biri bölümleme aşamasıdır. Bu nedenle yapılan çalışmalarda, hava tahmini uygulamasının modellenmesi için kullanılacak alan küçük parçalara (hücrelere) bölünmüştür. Hücrelere bölme işlemi hava tahmini uygulamasının modellenmesi için kullanılan önemli bir koşut programlama yaklaşımıdır [9]. Bölme işlemi yapılırken eldeki probleme ve benzetimi yapılacak alana bağlı olarak, çalışılacak alan belirlenmiştir ve belirlenen bu alan küçük parçalara bölünmüştür. Alanın kaç tane küçük parçaya bölüneceği ile ilgili net bir sayı verilmeksizin bu durumun problemin kendisine ve kullanıcıya göre değişiklik göstereceği vurgulanmıştır. Burada önemli olan noktanın, benzetimin işlemcilere nasıl dağıtılacağı kısmı olduğu söylenmiştir. Bölümlemeyi yaparken işlemci sayısı göz önüne alınarak her bir hücreyi bir işlemcide çalıştırabileceği öngörülmüştür. Problemin tasarımına göre bu konunun göz önüne alınması gerektiği ve benzetimin başarımının iyi hesap edilmesi gerektiği vurgulanmıştır.

Hava tahmin modellemesi için geliştirilen koşut algoritmalarda, eğer iki boyutlu uzayda (x,y) çalışılacaksa, bölümlenecek olan alan “x*y” kadar, üç boyutlu çalışılacaksa “x*y*z” kadar hücrenin olması gerektiği söylenmiştir [9].

Şekil 6.2'de iki boyutlu uzay üzerinde 5x6'lık alana sahiptir. Bu durumda alan 30 eş parçaya bölünmüştür.



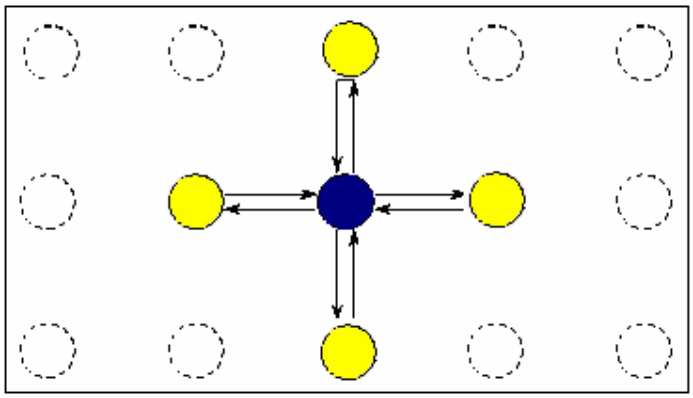
Şekil 6.2. 5x6'lık alan

6.1.2. Haberleşme (Communication)

Bir programı koşutlaştırmak için yapılacak bir diğer adım bölümleme aşamasıdır. Hava modelleme alanında yapılan çalışmalar doğrultusunda, bölümlenmiş hücrelerin her birinin hangi hücrelerle nasıl haberleşeceği bu kısımda anlatılacaktır.

Yapılan çalışmalarda, belirli bir alan üzerinde sadece komşu hücrelerin birbirleri ile veri alış-verişinde bulunacağı belirtilmiştir. Komşu hücrelerinde kendi aralarında ısı, sıcaklık, rüzgar, nem ve basınç gibi bilgileri paylaşacakları söylenmiştir.

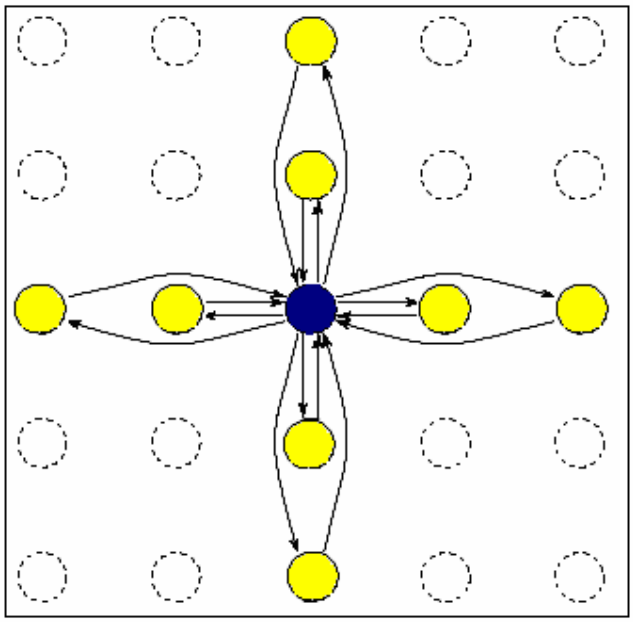
Veri alış-verişi yapmak için de komşu hücrelerin (adjacent cells) tespit edilmesi gerekmektedir. Komşu hücreler iki boyutlu uzayda çalışıldığı varsayılırsa koordinat düzlemi üzerinde Şekil 6.3'deki gibi gösterilmiştir [9].



Şekil 6.3. Beş Noktalı Kalıp

Şekil 6.3'de görüldüğü gibi belirlenen hücrenin koordinatı $X_{i,j}$ ise komşuları koordinat sistemi üzerinde $X_{i-1,j}$, $X_{i+1,j}$, $X_{i,j-1}$, $X_{i,j+1}$ şeklinde bulunmuştur. Şekilde görülen sistem beş noktalı kalıp (five-point stencil) olarak adlandırılmaktadır. Dolayısı ile şekilde görülen hücre ısı, sıcaklık, nem, rüzgar ve basınç değerlerini sadece komşuları ile paylaşacaktır. İki boyutlu uzayda her bir hücrenin kenar uzunluğunun "h" kadar olduğunu varsayarsak beş noktalı kalıp kullanılarak komşulara erişmek için kullanılan formül {(x-h, y), (x, y), (x+h, y), (x, y-h), (x, y+h)} olarak ifade edilmiştir [10].

Beş noktalı kalıbın dışında kullanılan bir diğer model de dokuz noktalı kalıp (nine-point stencil) olarak adlandırılır. Şekil 6.4'de dokuz noktalı kalıp modeli gösterilmiştir.



Şekil 6.4. Dokuz Noktalı Kalıp

Yukarıda tanımlanan her iki modelde de sağ, sol, alt ve üst komşularla haberleşme olanağı sağlanmıştır. Çapraz komşularla haberleşmeyi sağlamak ise bir üst ya da bir alt komşu vasıtasıyla gerçekleştirilebilmektedir [11].

Şekil 6.4'de belirlenen hücre çapraz komşuları göz önüne alınmazsa sadece sekiz komşusuyla ısı, sıcaklık, nem, basınç ve rüzgar bilgilerini paylaşacaktır.

Koşut programlamada alanı hücrelere bölme ve haberleşme işlemleri genel olarak bahsedilen şekillerde gerçekleşmektedir. Bu yaklaşıma uygun olarak modelleme yaparken süreci kolaylaştırmak için bazı kütüphaneler geliştirilmiştir. Örneğin; Colorado Üniversitesinde yapılan bir araştırma sonucu koşut programlamada kullanılan koşut işlem kütüphanesi olan MPI genişletilmiş ve MPI-RGL (*Regular Grid Library)* adı verilen bir kütüphane gerçekleştirilmiştir [12].

MPI-RGL, yukarıda bahsedilen haberleşme kısmıyla ilgili olarak önemli işlevler sunmaktadır. Örneğin; `MPI_RGL_STENCIL_CREATE()` işlevi kullanıcının isteği doğrultusunda bölümlenen hücrelerin komşuluk ilişkilerini, belirtilen kalıp türüne göre, beş noktalı ya da dokuz noktalı kalıp olarak oluşturur. `MPI_RGL_EXCHANGE()` işlevi ise, otomatik olarak gerekli hücreler arasındaki bilgi alış-verişini sağlar. Bu işlev MPI kütüphanesindeki `MPI_SEND()` ve `MPI_RECV()` işlevlerine denk gelmektedir [12].

Yapılan bu çalışmalarda modellemenin koşutluğu için üçüncü bölümde değinilen alan ayrıştırma tekniği kullanılmıştır.

6.2. Hava Tahmin Modeli Örneği: WRF (*Weather Research and Forecast Model*)

WRF, NCAR başta olmak üzere birçok araştırma laboratuarı ve çeşitli üniversitelerden bilim adamlarının katılımıyla tasarlanmış, 1-10 km ölçekli alanlarda hava ve bölgesel iklim tahmin yapmayı hedefleyen bir modeldir [29]. NCAR tarafından tasarlanmış, yaygın olarak kullanılan, büyük ölçekli ve iyi oturtulmuş fakat eskimekte olan MM5 hava tahmin modelinin yerini almaya adaydır [29]. WRF, esnek, geliştirilebilir, verimli, taşınabilir ve sürdürülebilir bir yazılım geliştirmeye olanak sağlamaktadır [30][31].

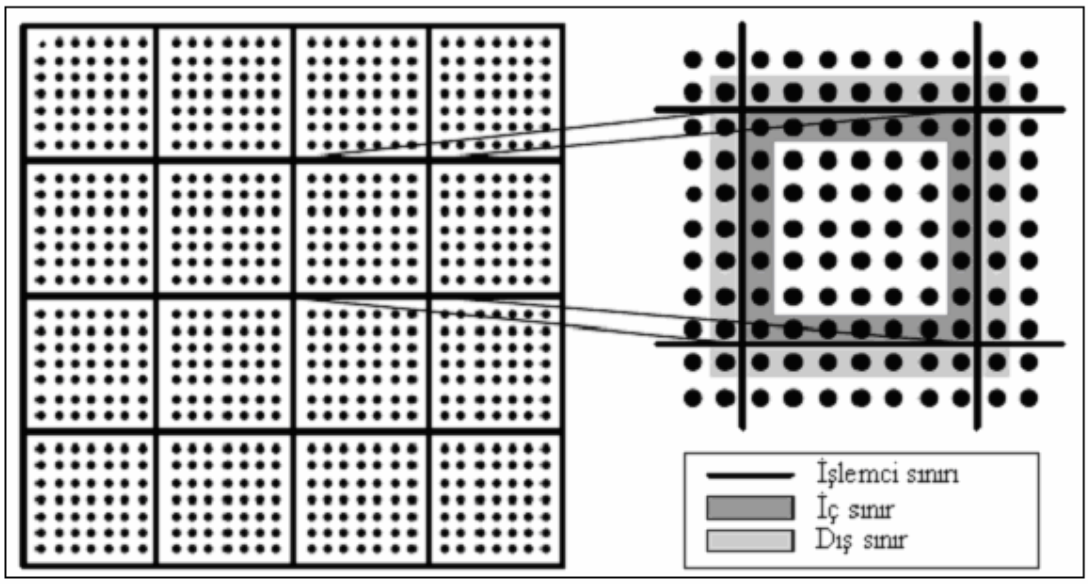
6.2.1. WRF Yazılım Çatısı

WRF Fortran90 programlama dili kullanılarak tasarlanmıştır ve üç katmanlı bir yapıya sahiptir. Bu katmanlar; sürücü katmanı (*driver layer*), model katmanı (*model layer*) ve aracılık katmanıdır (*mediation layer*). Sürücü katmanı, bellek atanması ve kaldırılması, üst düzey ilklendirme ve alan ayrışımı yönetimi gibi işlerden sorumlu olan katmandır. Model katmanı, daha çok bilim adamları tarafından yazılan ve asıl model hesaplamalarının yapıldığı katmandır. Aracılık katmanı ise sürücü katmanı ve model katmanı arasındaki haberleşmeden sorumludur. Bu katmanların yanı sıra destekleyici yapılar olarak kayıt birimi (*registry*), uygulama program arayüzü (*API*) ve girdi/çıktı (I/O) birimleri vardır. Kayıt birimi WRF durum değişkenlerini nitelikleriyle birlikte listeleyen tabloların tutulduğu bir veri tabanıdır. Bu veri tabanından katmanlar arası arayüz, haberleşme ve model girdi/çıktılarına çağrı döngüleri otomatik olarak oluşturulabilmektedir [32].

6.2.2. Koşut Çalışma Yapısı

WRF koşut olarak çalıştırılırken iki seviyeli bir ayrıştırma stratejisi uygulanmaktadır. Birinci seviyede hava tahmini yapılacak bölge yatay düzlemde iki boyutlu (2-D) alanlara (doğu-batı, kuzey-güney) bölünür. Bölgenin üç boyutlu değil

de iki boyutlu alanlara bölünmesinin nedeni, hava içerisinde hava tahmini yapmaya sağlayacak etmenlerin daha çok dikey olarak birbirlerine bağlı olmalarındandır. Bu nedenle yatay bölge üzerinde bulunan havadaki maddelerin dikey bağımlı hesapları, komşu bölgelerden bağımsız olarak yapılabilmektedir. Komşu bölgelerle haberleşme sadece sınır hattı boyunca belirlenecek bölgeler için yapılmaktadır. Şekil 6.5'te birinci seviye ayrıştırma ve her bir bölgenin haberleşme için belirlenen sınır yapısı gösterilmiştir.



Şekil 6.5. İki boyutlu alan ayrıştırma yöntemi

Her bir bölge dağıtılmış bellek yapısındaki bir işlemciye atanır [33]. Bölgelerin merkezinde kalan kısımların güncellemesi haberleşmeye ihtiyaç duyulmadan yapılmaktadır. Bölgelerin sınıra yakın kısımları (iç sınırlar), komşu bölgelerden bilgiye ihtiyaç duymaktadır. Haberleşme yükünü azaltmak için komşu bölgelerin sınıra yakın kısımlarındaki (dış sınırlar) bilgiler yerel olarak bilgiyi alacak olan işlemcilerde kaydedilirler [29].

Tanım olarak, her bir bölgeye yama (*patch*) adı verilmektedir. Her bir yama daha sonra tek tek işlemler bazında ayrıştırılır ve her bir işlem için ayrılan kısma da parça (*tile*) adı verilmektedir. Bir başka deyişle her parça bir işleme karşılık gelmektedir. Bu da WRF'nin ayrıştırma stratejisinin ikinci seviyesidir. Bu seviyede ayrıştırma sayesinde WRF kodu, bellek yapısı hangi türde olursa olsun (örneğin, tamamıyla dağıtılmış, tamamıyla paylaşımlı ya da karma) tek işlemcili yapılara

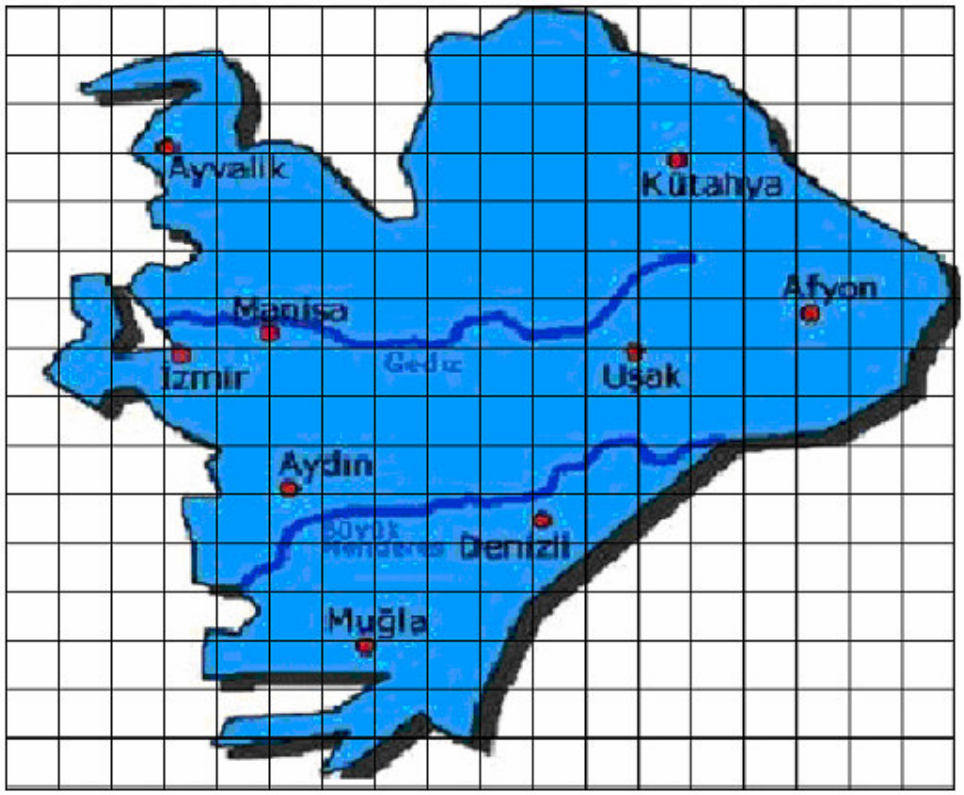
dahi paylaştırılabilmektedir. Bu da WRF'ye taşınabilirlik ve esneklik konusunda önemli avantajlar sağlamaktadır [33].

Yamalar arasında koşut işlem haberleşme kütüphanesi olarak MPI kullanılırken, parçalar arasında OpenMP kullanılmaktadır [29]. Veri transferi bloklar halinde yastık alanlara (*buffer*) yazılarak, bunların `MPI_Bcast()` komutuyla `MPI_BYTE` veri yapısında gönderilmesiyle gerçekleştirilmektedir.

6.3. Hava Tahmin Benzetiminin HLA İle Olan Uygunluğunun İncelenmesi

Tezin bu bölümünde verilen HLA bilgileri hava tahmin modellemesinde kullanılarak, teorik olarak bunun uygulanabilirliği incelenecektir.

Tasarımın ilk adımı olarak, hava tahmin modelinin HLA'ya uygun olarak tasarlanması için gerekli olan bazı tanımlamalar yapılmıştır. Bunlardan ilki, model olarak çalışılacak herhangi bir haritadır. Bu harita üzerinde veri dağıtımı yönetimi konusunda değinilen algoritmalardan ızgara tabanlı veri dağıtımı yönetimi kullanılmıştır. Bu algoritmanın seçilmesindeki en önemli sebep diğer algoritmalara göre gereksiz bilgi alış-verişinin daha az olmasıdır. Tasarımın bir sonraki aşamasında, harita sayıdan bağımsız "**NxM**"lik hücrelere bölünmüştür. Şekil 6.6'da hücrelere bölünmüş harita görülmektedir.



Şekil 6.6. Hücrelere Bölünmüş Harita

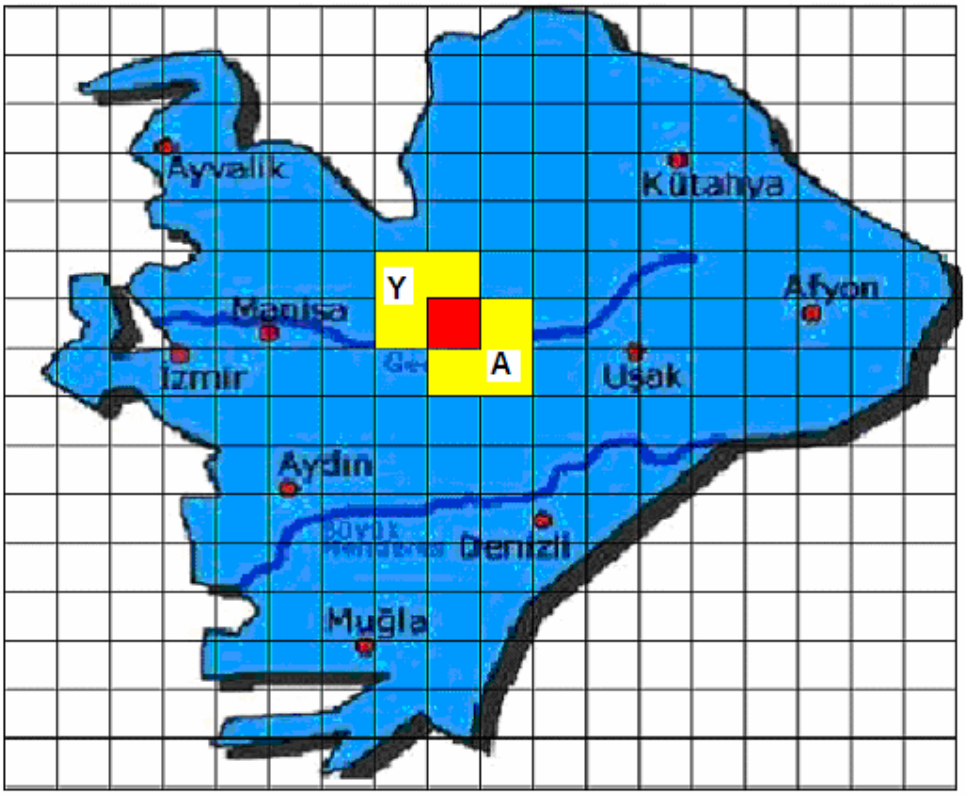
Buradaki N ve M tamamen modellemenin amacına ve tasarımcıya göre değişebilen sayılardır. N ve M sayılarını artırmak, daha fazla sayıda küçük hücre elde etmek demektir. Hücreler ne kadar küçük olursa, gerekli olmadığı halde iletilen bilgilerin alınma olasılığı o kadar az olacaktır. Ancak hücre büyüklüğünün küçülmesi, hücrelerin sayısının artması ve bu hücrelere düşen abonelik ve yayınlama bölgelerinin yönetilmesinin daha pahalı olması anlamına gelmektedir. Bunun tersi durumda, hücre sayısı azaldıkça, daha geniş alanda çalışıldığı için gereksiz bilgiler, ilgisiz alanlara iletilebilmektedir. Başta da değinildiği gibi bu artı ve eksiler tasarımın türüne göre göz önüne alınmalı ve ona göre tasarım yapılmalıdır.

Bir sonraki aşamada, federeler tasarlanmıştır. Başlangıç olarak herhangi bir federe sayısı verilmemiştir. İlk federe benzetime katıldığı andan itibaren diğer federeler dinamik olarak benzetime katılma ve çıkma hakkına sahiptirler.

Her federe ısı, sıcaklık, nem, basınç, rüzgar ve koordinat niteliklerine sahiptir. Her alanın ısı, sıcaklık, nem, basınç, rüzgar ve koordinat değerleri birbirinden farklı olduğu için federelerin birbirleri ile haberleşmelerinde bu değerler esas alınmıştır.

Koordinat bilgisi, hangi federeden bilgi geldiğini veya hangi federeye bilgi gönderileceğinin bilinmesi açısından önemlidir. Federelerin birbirleriyle haberleşmeleri veri dağıtım yönetimi tarafından belirlenen standartlar doğrultusunda yayınlama (*publish*) ve abone olma (*subscribe*) mekanizmaları tarafından sağlanmıştır.

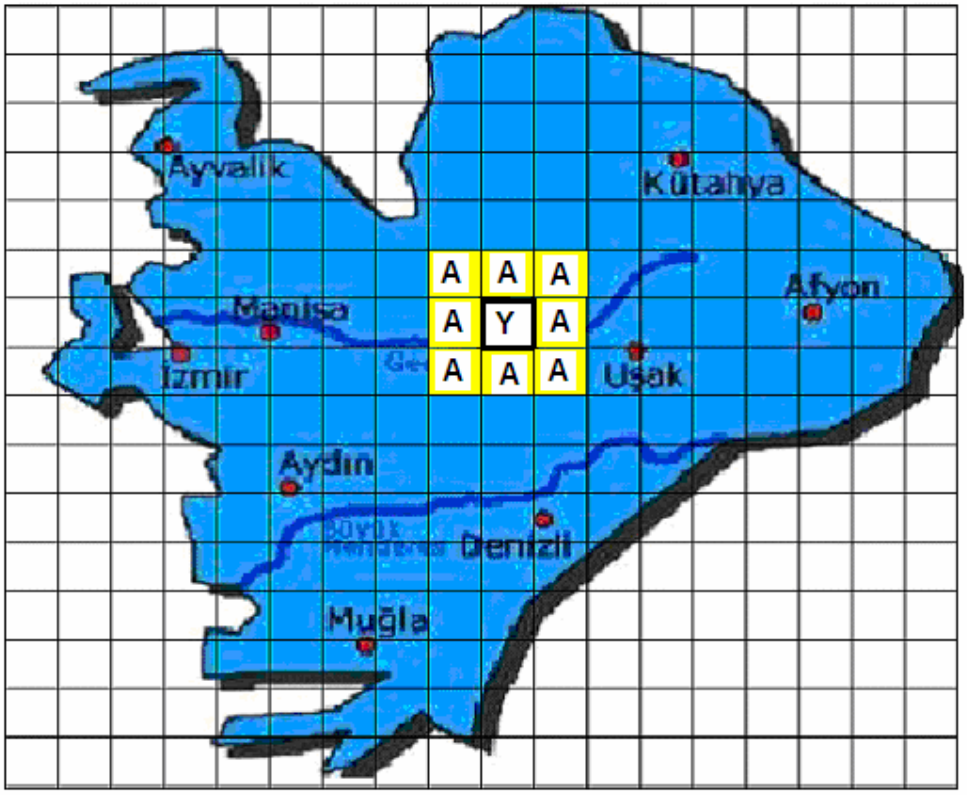
Örneğin; Federe_X'in, hücre NxM alanını yayınlamış olduğu varsayılsın. Başka bir federe (Federe_Z) yine aynı hücredeki alana abone olsun. NxM hücresi içerisinde bir kesişme olup olmadığının kontrolü ilgili algoritmalarca yapılır. Eğer herhangi bir kesişmeye rastlanmış ise Federe_Z, Federe_X'in yayınladığı bilgileri alabilir. Bu durum Şekil 6.7'de detaylı gösterilmiştir.



Şekil 6.7. Yayınlama ve Abone Olma Mekanizması

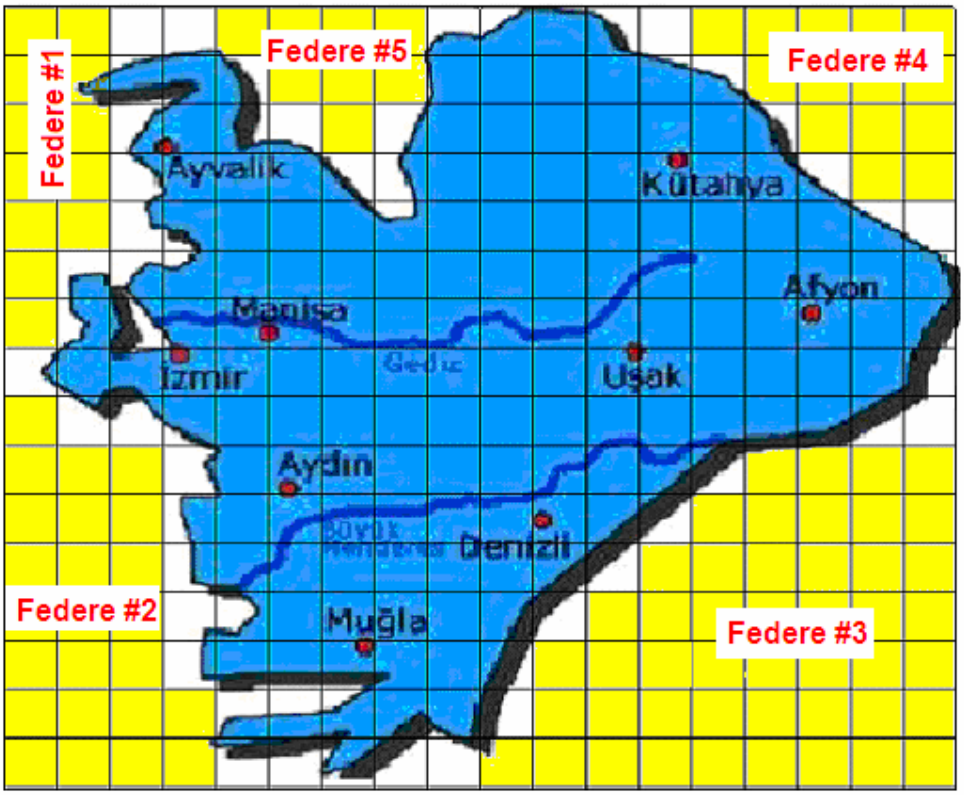
Şekil 6.7'de görüldüğü gibi, Federe_Y dört hücrelik bir alanı yayınlamış ve Federe_A'da bu 4 hücrelik alandan birini kendi içerisine alacak şekilde bu alana abone olmuştur. Her iki alanın kesiştiği nokta gösterilmiştir.

Federeler arası veri alış-verişi yapılırken, bir hücrenin içerisindeki alanı yayınlayan federe, kenar ve köşelerinin temas ettiği komşusu olan sekiz hücreye abone olması ve sadece bu bölümlerdeki federelerin ısı, sıcaklık, nem, basınç, rüzgar ve koordinat değerlerini almaları tasarım olarak uygun görülmüştür. Diğer bir ifadeyle, bilgilerini yayınlayan federenin kendisinden çok uzaktaki bir alanın bilgilerine ihtiyaç duymayacağı düşünülmüştür. Şekil 6.8'de bu tasarım gösterilmiştir.



Şekil 6.8. Federeler Arası İletişim

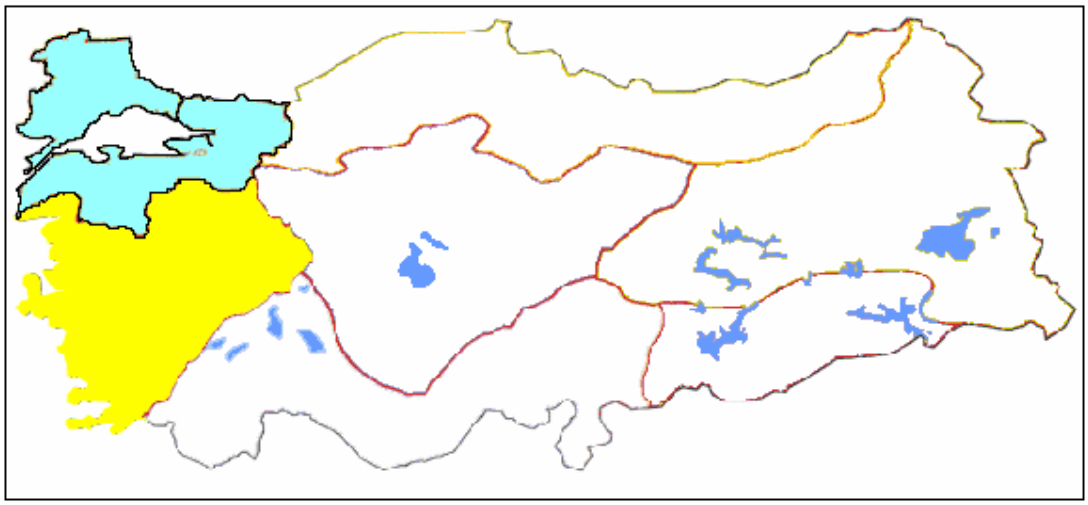
Şekil üzerinde görülen boş hücreler için de farklı bir tasarım düşünülmüştür. Harita alanına girmeyen köşelerdeki her bir hücreyi bir federeye atamak yerine topluca tek bir federenin sorumluluğuna atamak daha yararlı görülmektedir. Böylece ağ trafiği de fazla meşgul edilmemiş olacaktır. Bu durum Şekil 6.9'da görülmektedir.



Şekil 6.9. Harita Alanına Girmeyen Federeler

Şekil 6.9'da görüldüğü gibi, harita alanına girmeyen hücreler topluca tek bir federenin sorumluluğuna verilmiştir. Örneğin; haritanın sol köşesinde kalan alanla ilgili her türlü bilgi Federe #2 tarafından yürütülecektir.

Bir diğer önemli konu, bu benzetimin yeniden kullanılabilir olması için neler yapılması gerektiği konusudur. Bunun için hücre konum bilgisi yerine enlem-boylam bilgisinin kullanılması daha yerinde olacaktır. Örneğin; Ege Bölgesi üzerinde çalışıldığında ve bu çalışma üzerine Marmara Bölgesi eklenmek istendiğinde, verilen hücre sayıları problemlere yol açacaktır. Marmara Bölgesi için tasarlanan hücre 5x6 ile Ege Bölgesi için tasarlanan 5x6 hücresi aynı alanı gösteriyor gibi algılanmaktadır. Oysaki her iki hücre farklı alanlara işaret etmektedir. Bu yüzden NxM hücresi demek yerine 29:06 doğu enlemi, 37:46 kuzey boylamı, Federe_X tarafından yayınlanmış ya da abone olunmuş şeklinde ifade edilmesi daha evrensel ve gerçekçi bir tanım olacaktır.



Şekil 6.10. Benzetimin Yeniden Kullanılabilirliği

6.4. Hava Tahmin Modellemesi için Benzetim Nesne Modeli (*SOM*)

Hava tahmin modellemesi için yapılan tasarımda federe sayısı dinamik olarak değişmekteydi. Tasarımdaki bütün federelerin nitelikleri aynı olmakla birlikte, bu nitelikler ısı, sıcaklık, nem, basınç, rüzgar ve koordinat olarak belirlenmiş idi. Bu bölümde ise HLA'nın federeler için standartlaştırdığı kurallardan biri tasarlanacaktır. Bu kural "her federenin, HLA nesne model şablonuna göre düzenlenmiş bir HLA benzetim nesne modeli (SOM) olmalıdır " şeklindedir. Bu kural doğrudan HLA'nın üç standardından biri olan HLA nesne modeli şablonu (OMT) ile tanımlanır. OMT ile SOM oluşturmak için OMT'nin ondört bileşenini oluşturmak gerekmektedir. Bu bileşenler tezin nesne modeli şablonu başlığı altında verilmiştir.

Hava tahmin modeli için ilk tanımlanan bileşen Nesne Modeli Kimlik Çizelgesi (*Object Model Identification Table*) dir.

HLA nesne modellerinin tasarlanmasındaki temel amaçlardan biri tekrar kullanılabilirliği sağlamaktır. Nesne modeli kimlik çizelgesinin temel amacı, nesne model tanımının yanında bir takım kimlik bilgilerini belgelemektir.

Hava tahmin modeli için tasarlanan nesne modeli kimlik çizelgesi aşağıda görülmektedir.

Çizelge 6.1. Nesne Modeli Kimlik Çizelgesi

Nesne Modeli Kimlik Çizelgesi	
Kategori	**Bilgi**
İsim	HavaTahminModeli
Tür	SOM
Versiyon	1.0 Alpha
Değişme Tarihi	07 Eylül 2007
Amaç	Hava tahmin federesi için nesne modeli
Uygulama Alanı	Hava durumu işlemleri
Sponsor	Hacettepe Üniversitesi
Kişi	Tülin Erçelebi
Organizasyon	ABC
Telefon	2977500-119
E-mail	tulin@cs.hacettepe.edu.tr
Referans	www.kayhan_imre.com
Diğer	MM5 uygulaması incelenebilir

SOM oluşturmak için gerekli olan bileşenlerden bir diğeri de Nesne Sınıfı Yapı Çizelgesidir. HLA nesne sınıfları, belirli özellikleri ve nitelikleri olan nesnelerdir. HLA nesne sınıfı yapı olarak nesneye yönelik sistemlerin sınıf hiyerarşisine benzerler. “HLAobjectRoot” SOM’daki diğer tüm sınıfların atası, yani kök sınıftır. Nesne sınıfları arasında çoklu kalıtıma izin verilmez, yani her sınıfın tek bir atası olabilir.

Hava tahmin modeli için tasarlanan nesne sınıfı yapı çizelgesi Çizelge 6.2’de görülmektedir.

Çizelge 6.2. Nesne Sınıfı Yapı Çizelgesi

		Nitelikler
HLAobject Root (N)	Hava (Air)	Isı (AY)
		Sıcaklık (AY)
		Nem (AY)
		Basınç (AY)
		Rüzgar (AY)
		Koordinat (AY)

Her nitelik, çizelgede de görüldüğü gibi yayınlanabilir (publishable), abone olunabilir (subscribable) ve/veya hem abone olunabilir hem de yayınlanabilir olabilir (AY). Bir niteliğin hem abone olunabilir hem de yayınlanabilir olması demek kendi özelliklerini yayınlayabileceği gibi başka sınıfların özelliklerindeki değişimleri gözleyip buna göre tepki gösterebileceği anlamına gelir.

Bir diğer bileşen Etkileşim Sınıfı Yapı Çizelgesidir. Etkileşim, bir federenin gerçekleştirdiği ve başka bir federeyi etkileyebilecek dış eylemdir. Etkileşimler HLA nesne modellerinde nesne sınıfları gibi sınıf hiyerarşisinde tanımlanmalıdırlar. "HLAinteractionRoot" etkileşim sınıfı FOM ve SOM'daki tüm diğer etkileşim sınıflarının atası, yani kök etkileşim sınıfı olmalıdır.

Bir diğer bileşen zaman gösterim çizelgesidir. Benzetim içinde kullanılacak zaman ilerleme durumu, benzetimin amacına göre belirlenir. Gerçek zamanlı sistemlerden daha hızlı olay güdümlü (*event driven*) benzetimler analiz uygulamalarında sıkça kullanılırken, gerçek zamanlı, zaman adımlı (*time stepped*) benzetimler eğitim uygulamalarında kullanılır. Benzetimin zaman ilerleme durumunun doğru seçilmesi benzetimin başarımını ve benzetimin federasyondaki diğer benzetimlerle birlikte çalışabilirliğini etkilediği için oldukça önemlidir. Bu nedenle benzetimin zaman ilerleme durumunun SOM'da belgelenmesi gerekmektedir. Hava tahmini yapabilmek için, ısı, sıcaklık, nem, basınç, rüzgar ve koordinat niteliklerinin

zamana bağlı grafiklerinin bilinmesi gerekir. Bunun için de bu niteliklerin belirli zaman aralıklarında örneklenmesi gerekmektedir. Bu sebeple, hava tahmin benzetimi tasarımı için zaman adımlı zaman ilerleme durumu daha uygundur. Hava tahmin modeli için tasarlanan zaman gösterim çizelgesi Çizelge 6.3'te verilmiştir.

Çizelge 6.3. Zaman Gösterim Çizelgesi

Zaman Gösterim Çizelgesi		
Kategori	**VeriTürü**	**Anlamsal**
Zaman Damgası	ZamanTürü	Negatif olmayan bir Δt değerine sahiptir
Lookahead	LookaheadTürü	Negatif olmayan bir Δt değerine sahiptir

OMT ile SOM oluşturmak için OMT'nin ondört bileşenini oluşturmak gerektiği tezin önceki bölümlerinde bahsedilmiş idi. Ancak hava tahmin modellemesinde gerekli olan nesne ve etkileşimler doğrultusunda bu ondört bileşenin hepsini kullanmak bu tasarım için gerekli görülmediğinden sadece kullanılacak bileşenlere yer verilmiştir.

Sonuç olarak, hava tahmin modellemesinin günümüzde pek çok mimariyle benzetiminin yapılmasının yanında HLA ile benzetiminin yapılmasının mümkün olduğu görülmüştür. Bunun yanında HLA ile modellendiğinde yeniden kullanılabilirlik açısından avantaj sağlamakla birlikte başarım ve maliyet konularında da önemli yararlar sağlayacağı düşünülmektedir.

6.5. WRF ve HLA ile Tasarlanan Hava Tahmin Modelinin Karşılaştırılması

Çizelge 6.4'te WRF ve HLA ile tasarlanan hava tahmin modeli sözde kod yapısı temel alınarak kavramsal olarak karşılaştırılmıştır. Çizelge 6.5'te ise WRF ile HLA ile tasarlanan hava tahmini benzetimi, başarım, yeniden kullanılabilirlik, ölçeklenebilirlik, kolay gerçekleştirilebilirlik ve birlikte kullanılabilirlik kıstasları açısından karşılaştırılmıştır. Bu kıstasların karşılaştırılmasında şu hususlar dikkate alınmıştır:

Başarım analizinde işlemciler arası veri gönderim sıklığı, gönderilen iletilerin boyutları ve kullanılan haberleşme alt yapısından kaynaklı gecikmeler incelenmiştir. Sık sık haberleşme ihtiyacı duyan ve bu haberleşmeler esnasında küçük boyutlu iletiler gönderen uygulamalar genel olarak başarımı düşük olarak değerlendirilmiştir.

Ölçeklenebilirlikte uygulamanın işletim zamanının, veri yükü ve işlemci sayısıyla olan ilişkisine bakılmıştır. Veri yükü ve işlemci sayısı aynı oranda artırıldığında işletim zamanı değişmeyen uygulamalar ölçeklenebilir olarak değerlendirilmiştir.

Kolay gerçekleştirilebilirlikte uygulamanın boyutuna, dinamik veri yapısı kullanılıp kullanılmadığına ve uygulamayı gerçekleştirirken yardımcı araç desteğinin sağlanıp sağlanmadığına bakılmıştır. Büyük boyutlu ve dinamik uygulamalar kolay gerçekleştirilebilir olarak değerlendirilmemiştir.

Yeniden kullanılabilirlik ve birlikte işlerlikte uygulamanın başka platformlara adapte edilmeyi destekleyip desteklemediğine ve geliştirilen uygulama parçalarının başka uygulamalarda tekrar kullanılıp kullanılamadıklarına bakılmıştır.

Çizelge 6.4. WRF ve HLA ile tasarlanan hava tahmin benzetimi sözde kodları

WRF'nin sözde kodu:	**HLA ile tasarlanan hava tahmin benzetimi sözde kodu:**

ilk_degerleri_ata();
hucre_dizinle();
komsuluk_belirle();
while (! benzetim_sonu)
{
zaman_bilgisini_guncelle();
if (bilgi_gereksinimi)
aracilik_katmani_cagrisi();
 	
while(aracilik_katmani_is_liste!=bos)
bekle();
 	
tahmin_hesaplarini_yap();
//komşulara verileri gönder
broadcast();//MPI ile
//merge
aracilik_katmanini_haberdar_et();
 	
}
sonlandir();
	RTI :
	federasyon_yarat();
	while (! benzetim_sonu)
	{
	federe_listesi_guncelle();
	yayin_abonelik_listesi_guncelle();
	while (! basarili_alim)
	{
	reflect : federeleri_haberdar_et();
	bekle();
	}
	}
	
	Federe :
	benzetime_katil();
	bilgileri_guncelle();
	yayin_listesi_gonder();
	abone_listesi_gonder();
	while(! benzetim_sonu)
	{
	tahmin_hesaplarini_yap();
	bilgileri_yayinla();
	while (! reflect)
	bilgileri_al();
	attribute_guncelle();
	}

HLA ile tasarlanan hava tahmin benzetiminde zaman yönetimi federeleri_haberdar_et (reflect) işlevi aracılığıyla otomatik olarak sağlanmaktadır. Etkileşim hesaplarını yapıp, sonuçlarını yayınlayan federe, diğer federelerin de sonuçlarını yayınlayıp, daha sonra RTI tarafından ilgilendiği bilgilerin başarılı bir şekilde kendisine gönderilmesini bekler. RTI ise tüm federeler güncellemelerden başarılı bir şekilde haberdar olmadan, yeni yayın alımı yapmaz.

Çizelge 6.5. WRF ve HLA ile tasarlanan hava tahmin modellemesi karşılaştırması

	WRF (MPI ile)	HLA ile Hava Tahmin Benzetimi
Başarım Analizi	X	√
Ölçeklenebilirlik	X	√
Kolay gerçekleştirilebilirlik	X	√
Yeniden kullanılabilirlik	√	√
Birlikte kullanılabilirlik	√	√

Hava tahmin modellemesinde, MPI iletişimini kötü yönde etkileyen küçük zaman aralıklarında veri değişimi ihtiyacı olmaktadır ve çoğu zaman ileti boyutları küçüktür [29]. Bu da veri iletimi etkinliğini önemli ölçüde azaltmaktadır. Hücrelerin sınır bölgelerinde bulunan ve komşu hücrelerdeki değişkenlere bağlı olan değerler, hesaplar tamamlanmadan iletilemeyecekleri için, sık ileti gönderimi zaman maliyetini önemli ölçüde olumsuz etkileyen bir faktör olmaktadır. HLA'da WRF gibi alan ayrıştırma yöntemi kullanılmasına rağmen, MPI gibi bir veri iletimi kütüphanesinin kullanılmaması ve haberleşmeyi düzenleyen veri dağıtımı yönetimi servisinin bulunması, HLA'nın başarımının daha iyi olacağı beklentisini yaratmaktadır.

WRF'de kullanılan alan ayrıştırma yöntemi molekül dinamiği benzetiminde kullanılan alan ayrıştırma yöntemiyle benzerlik göstermektedir. Molekül dinamiği alan ayrıştırma yönteminden farklı olarak, parçacıkların yer değiştirmesi gibi bir şey söz konusu olmadığından, parçacık takibi gibi bir durum söz konusu değildir. Ancak yine de bu, hava tahmin modellemesinin kolay gerçekleştirilebilir olarak değerlendirilmesine yeterli olmamaktadır.

WRF'nin ölçeklenebilirliğini ölçmek için çeşitli platformlarda, işlemci sayısını ve hücre sayısını aynı oranda artırarak işlem zamanı ölçülmüştür [29]. İşlemci sayısı birden başlayarak artırılmıştır. İşlemci sayısı ve hücre sayısı aynı oranda artırıldığında işlem zamanı değişmeyen sistemler ölçeklenebilir olarak kabul edilmektedir. Buna göre başlangıç değerleri için (1-50) ölçeklenebilirliğin oldukça zayıf olduğu gözlemlenmiştir. Sisteme her yeni işlemci katıldığında, paylaşımlı

bellek veri yolunda yaşanan bellek erişimi çekişmesi (*contention*) durumu daha da kötüleştirmiştir. Ayrıca artan MPI haberleşme yükü de fazladan başarım düşüşüne sebep olmuştur. Testlerde kullanılan, daha düşük işlem kapasitesine rağmen daha hızlı bir iletişim ağı altyapısına sahip olan PSC TSC (*Pittsburgh Supercomputing Center Terascale Computing System*) sistemi dışındaki sistemlerde WRF ölçeklenebilir olmaktan uzak bir görünüm sergilemiştir [29]. Sonuç olarak ölçeklenebilirliğin haberleşme etkinliğiyle doğru orantılı olduğu da söylenebilir.

WRF'nin katmanlı yapısı tasarlanırken, her bir katmanın bağımsız olarak da çalışabilmesi hedeflenmiştir. Örneğin WRF için yazılan sürücü katmanı arayüz gereksinimleri yerine getirildiği müddetçe başka modeller tarafından da kullanılabilmekte, ya da başka modeller için yazılmış sürücü katmanları WRF'ye bütünleştirilebilmektedir [33]. Bu özellikleriyle WRF birlikte kullanılabilir ve yeniden kullanılabilir olarak değerlendirilmektedir.

WRF'yi geliştirenler Fortran90 programlama diliyle beraber MPI kullandıkları için WRF'nin kaynak kodu oldukça uzun ve bir o kadar da anlaşılması zor bir kod haline gelmiştir. Aynı uygulamanın HLA ile gerçekleştiriminde ise, veri dağıtımı yönetimi servisi aracılığıyla komşuluk ilişkileri belirlenerek, abone olma ve yayınlama mekanizmalarınca federeler arasındaki iletişim sağlanabilmektedir. Federeler arası iletişim gerçekleştirilirken ızgara tabanlı veri dağıtım yönetimi kullanılarak haberleşme alanı istenilen boyutta genişletilebilir veya daraltılabilir.

7. MOLEKÜL DİNAMİĞİ (MOLECULAR DYNAMICS) BENZETİMİ

Molekül dinamiği (*molecular dynamics, MD*), fizik ve kimya gibi temel bilimlerde kullanılan çok küçük boyutlardaki parçacıkların etkileşimini fiziksel ve matematiksel olarak açıklayan bir bilgisayar benzetimidir.

Atomlar arası etkileşmeyi veren potansiyellerin metalleri de içerecek şekilde oluşturulması ve bunların yaygınlığı nedeniyle moleküler dinamik benzetimi metallerin ve alaşımların termodinamik, yapısal ve dinamik özelliklerinin hesaplanmasında yaygın olarak kullanılmaktadır [7].

Gerçek bir sistemde parçacıkların sayısı benzetimi yapılamayacak kadar fazla olduğu için MD, laboratuarda yapılan deneylerin nispeten çok daha küçük bir modelinin bilgisayar üzerinde benzetimini yapmaktadır. MD'nin sağladığı temel yarar laboratuarda uzun uğraşlar ve sağlanması masraflı koşullar sonucunda yapılacak olan deneylerin, benzetim sayesinde kısa ve masrafsız bir şekilde görülmesidir.

7.1. Molekül Dinamiği Benzetiminin Koşut Programlama İle Modellenmesi

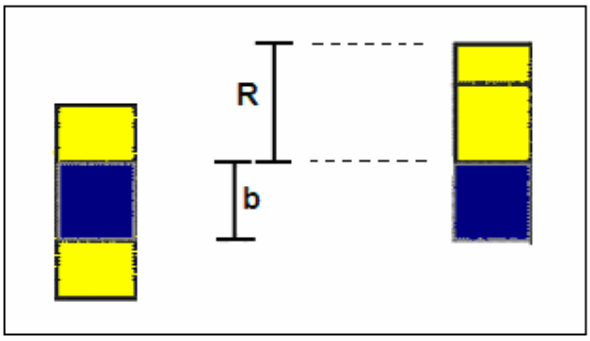
Molekül dinamiği benzetimlerinde parçacıkların konum ve hız bilgileri klasik olarak kullandığımız fizik kurallarına bağlıdır. Bu fizik kuralları çerçevesinde her zaman adımında bütün parçacıkların kuvvetleri hesaplanır (*force computation*). Newton'un hareket kanununa göre de bu kuvvet kullanılarak yine tüm parçacıkların konum ve hız bilgileri hesaplanır [14].

Molekül dinamiğinde 1 mikrosaniyelik bir hesaplama için 1 gün süren benzetimler yapılması gerekebildiği için benzetimi koşut olarak çalıştırarak görevleri işlemcilere dağıtmak daha avantajlı bir yaklaşımdır [13].

Molekül dinamiğinin koşut olarak modellenebilmesi için birçok algoritma ve tasarım vardır. Bu tasarımların hemen hemen hepsi birbirine benzemektedir. Bu adımda, bu modellemelerden farklı olarak ortaya sürülen bir çalışma olan "Desmand" tekniğinden bahsedilerek, Desmand tekniği ile molekül dinamiği modellemesinde işlemcilerin ne tür görevler üstlendikleri ve haberleşme düzeneğinin nasıl olduğu incelenecektir.

Desmand modelinde, parçacıkların bulunduğu alan eğer üç boyutlu uzayda ele alınacaksa küplere, iki boyutlu düzlemde ele alınacaksa karelere bölünmüştür. Burada da hava tahmin benzetiminde olduğu gibi hücreler arası iletişim sadece komşu hücrelerle sağlanmıştır. Komşu hücreler arasında paylaşılacak olan veriler parçacıkların konum, hız ve kuvvet bilgileri olarak belirlenmiştir. Komşu hücrelerin nasıl belirlendiği aşağıda detaylı olarak anlatılacaktır. Açıklamalar, iki boyutlu düzlemde çalıştığımız varsayılarak yapılacaktır.

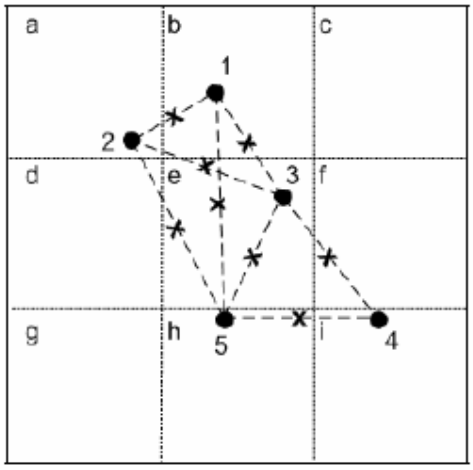
Molekül dinamiği için geliştirilen Desmand tekniğinde her kare için bir işlemci görevlendirilmiştir. Karelerin kenar uzunlukları b olarak verilmiştir. Üzerinde diğer atomlarla etkileşimi hesaplanacak olan parçacığın bulunduğu alan ana kutu "home-box" olarak adlandırılmıştır ve ana kutu içerisindeki bir parçacığın hangi parçacıklarla etkileşim halinde olacağını hesaplayabilmek içinde bir "R" sabiti tanımlanmıştır. Tanımlanan bu R sabitinin alabileceği tüm değerler denenmiş ve yapılan modellemede bu yöntemin kullanılması için öne sürülen iyileştirmelerden yararlanabilmek için bu sabitin R=1.5xb olması öngörülmüştür. Şekil 7.1'de b ve R kavramları gösterilmiştir.



Şekil 7.1. Desmand yaklaşımında b ve R değerleri

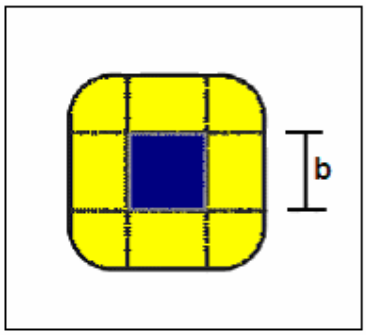
Şekilde gösterilen b ve R değerlerinin belirlenmesinin ardından, parçacıklar arasındaki mesafe R'den büyükse iki parçacık arasında bir etkileşimin olmadığı kabul edilmiştir. Bu durumda, ana kutu içindeki bir parçacığa R mesafeden daha yakın olan tüm parçacıklar etki edebilecektir. Daha değişik bir ifadeyle, bir parçacığa R mesafeden daha uzak olan parçacıklar komşu hücreler kapsamına girmediği için bu parçacıkla herhangi bir hız, konum ve kuvvet bilgisi paylaşılmayacaktır.

Bir sonraki adımda, arasında etkileşim olabilecek parçacıklar ve aralarındaki mesafenin tam orta noktası bulunur. Bu yönteme orta nokta (midpoint) adı verilir. Şekil 7.2'de bu nokta daha ayrıntılı verilmiştir.



Şekil 7.2. Orta Nokta (Midpoint) [13]

Bu noktanın, üç değişik alanda olma ihtimali vardır. Bunlardan ilki, üzerinde işlem yapılan parçacığın içinde bulunduğu alanda yani ana kutuda olabilme ihtimali, ikincisi bu parçacığa etki eden diğer parçacığın alanı içerisinde olabilme ihtimali, son olarak da her iki alanda da olmayıp ikisinden de farklı bir alanda olabilme ihtimalidir. Bu noktanın denk düştüğü alana da etkileşim kutusu (*interaction box*) denilmiştir. İki parçacığın etkileşim hesabı da bu etkileşim kutusundaki işlemci tarafından yapılmaktadır. Bütün bunların ışığında seçilen herhangi bir parçacığın etkileşim hesabının yapılacağı yer (*interaction box*) bu parçacığa en fazla R/2 mesafede olduğu sonucuna varılmıştır. Bir alanın sınırlarına (kareye) R/2 mesafe uzaklıktaki tüm noktaların birleştirilmesi ile çizilecek alana etki alanı (*import region*) denilmiştir. Şekil 7.3'te etki alanı temsili gösterilmiştir.



Şekil 7.3. Etki alanı

Şekil 7.3'te görülen koyu alan ana kutuyu, açık alanlar ise etki alanlarını temsil etmektedir.

Öne sürülen bu yöntemle parçacıklar arasındaki etkileşim mesafesi hesaplanmış, bu mesafe içerisinde kalan parçacıkların birbirleri ile haberleşmeleri öngörülmüştür. İki parçacığın etkileşim hesabı da orta noktaya düşen hücreden sorumlu işlemciye yaptırılmaktadır.

7.1.1. Molekül Dinamiği Benzetiminde Kullanılan Koşut Algoritmalar

Molekül dinamiği benzetimini koşut olarak gerçekleştirmek için temel olarak üç farklı algoritma kullanılmaktadır. Bunlar; atom ayrıştırma (*atom decomposition*), kuvvet ayrıştırma (*force decomposition*) ve alan ayrıştırma (*domain decomposition*) yöntemleridir. Bu yöntemler aşağıda kısaca açıklanarak başarım karşılaştırmaları, işlemcilere düşen yük miktarları ve gerçekleştirim detayları verilmiştir. Benzetimi yapılacak sistemdeki parçacık sayısına N, koşut işlemci sayısına da P denilmiştir.

i. **Atom Ayrıştırma Yöntemi:** Bu yöntemde benzetimi yapılacak sistemdeki parçacıklar, koşut ortamdaki işlemcilere eşit olarak paylaştırılır (N/P). Her bir işlemcinin sorumluluğundaki parçacıklar benzetim süresince değişmez ve her işlemci sorumluluğundaki parçacıklarla ilgili tüm hesaplamaları parçacıkların konumlarından bağımsız olarak yapar. Bu yöntemde tüm parçacıkların konum bilgilerinin kopyası tüm işlemcilerde tutulur. Kolay kodlanabilir olmasına rağmen, tüm parçacıkların konum bilgilerinin tüm işlemcilerde tutuluyor olması, hem bellek gereksinimini hem de haberleşme yükünü önemli

ölçüde etkilediğinden, parçacık sayısının çok fazla olduğu büyük ölçekli uygulamalarda tercih edilmemektedir [25][26][27].

ii. **Kuvvet Ayrıştırma Yöntemi:** Bu yöntem atom ayrıştırma yöntemine benzemekle birlikte, atom ayrıştırma yöntemindeki haberleşme yükünü azaltmayı hedefleyerek geliştirilmiştir. Bunu parçacıklar arası kuvvet matrisinin (NxN) bloklar şeklinde işlemcilere atanmasıyla sağlamaktadır. NxN boyutundaki kuvvet matrisinde N^2 tane etkileşim bulunmaktadır. Bu etkileşimler P adet işlemciye eşit olarak paylaştırıldığında, her bir işlemciye düşen bloğun boyutu N^2/P olacaktır. Bu ayrıştırma şu şekilde yapılmaktadır:

P, 2'nin kuvveti (2^n) ve N de P'nin katı olacak şekilde seçilir. Kuvvet matrisi satır ve kolonları $N/\sqrt{P}$ uzunluğunda $\sqrt{P}$ tane parçaya bölünür. Her bir alt bloğun boyutu $(N/\sqrt{P}) \times (N/\sqrt{P})$ olacaktır ve işlemcilere N^2/P tane etkileşim düşecektir.

Her bir işlemci tüm etkileşimler yerine, kendi üzerine düşen etkileşimleri hesaplar. İşlemciler sorumlu oldukları etkileşimleri hesaplarken, diğer işlemcilerden bilgi isteğinde bulunmazlar. Orta ölçekli uygulamalarda tercih edilen bir yöntemdir [25][26][27].

iii. **Alan Ayrıştırma Yöntemi:** Bu yöntemde benzetimi yapılan sistem geometrik olarak eşit hücrelere bölünür ve her bir hücrenin sorumluluğu bir işlemciye verilir. İşlemcilerin yük miktarı hücre içerisine düşen parçacık sayısıyla doğru orantılıdır. Homojen bir sistemde işlemcilere düşen yük miktarının eşit olacağı varsayılmaktadır (N/P). Her bir işlemci etkileşim hesapları için sadece komşu olduğu hücreye ait işlemcilerle haberleşir. Bu da bu yöntemin diğer yöntemlere oranla daha az haberleşme yüküne sahip olmasını sağlamaktadır. Büyük ölçekli uygulamalarda bu özelliği açısından tercih edilmektedir. Ancak parçacıklar hareketli olduklarından dolayı, parçacıkların sahiplik takibi zor olmaktadır ve bu da yazılan kodun karmaşıklığını artıran bir etkendir [25][26][27].

7.1.2. Molekül Dinamiği Benzetiminde Kullanılan Koşut Algoritmaların Karşılaştırılması

Yukarıda bahsedilen üç yöntem kod karmaşıklığı, haberleşme ölçeği, bellek ölçeği ve işlemci zamanı ölçeğine göre aşağıdaki Çizelge 7.1'de karşılaştırılmıştır. Çizelge 7.1'deki veriler [27]'den alınmıştır. Aşağıda atom ayrıştırma yöntemi ve alan ayrıştırma yönteminin haberleşme ölçeklerinin nasıl hesaplandığı anlatılmıştır:

Haberleşme ölçeği, alınan ve gönderilen verinin büyüklüğüyle doğru orantılıdır. Veri büyüklüğü de, işlemciler arasında ne kadar parçacığın bilgisinin iletildiğiyle doğru orantılı olarak artmaktadır. Atom ayrıştırma yönteminde, her bir işlemci N/P kadar parçacıktan sorumludur ve geriye kalan tüm işlemcilerdeki tüm parçacıkların (N-N/P) konum bilgisini kendinde tutar. Dolayısıyla atom ayrıştırma yönteminin haberleşme ölçeği O(N) olmaktadır.

Alan ayrıştırma yönteminde benzetimi yapılan sistem 3 boyutlu veya 2 boyutlu olarak alanlara ayrıştırılabilir. Benzetim sistemi 3 boyutlu olarak modellendiğinde, sistem uzayının, kenar uzunluğu *d* olan küpler halinde hücrelere bölündüğü ve etkileşim mesafesinin de *r* olduğu varsayılmıştır. Aşağıdaki Şekil 7.4'de örnek bir hücre ve bu hücrenin etkileşim sınırları geometrik olarak gösterilmiştir. Her bir hücrenin, etkileşim bölgesine giren alanın hacmi şu şekilde hesaplanmaktadır:

$$V_{etki} = (d+2r)^3 - d^3 \tag{7.1}$$

İşlemciler etkileşim hesaplarını yaparken, sorumlu oldukları alanlara komşu alanlardaki, etkileşim sınırları içerisinde kalan parçacıkların bilgisine ihtiyaç duymaktadırlar. Parçacıkların homojen olarak dağıldığı varsayılırsa her bir hücrede, yani d^3'lük hacimde N/P tane parçacık bulunur. Buna göre etkileşim bölgesinde bulunan parçacık sayısı ve buna bağlı haberleşme ölçeği aşağıdaki gibi bulunmaktadır:

i) $d << r \rightarrow V_{etki} \approx 8r^3$ (7.2)

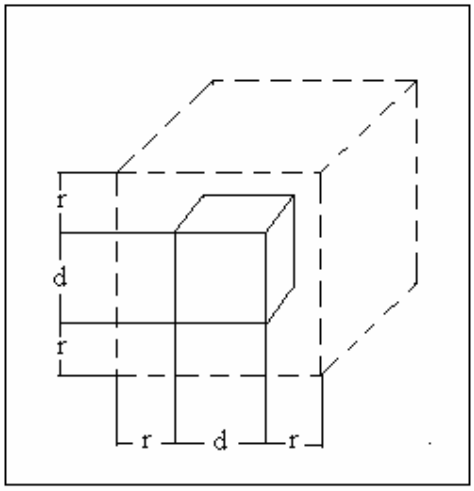
Bu durumda haberleşme ölçeği $8r^3$ ile ölçeklenmektedir, yani $O(r^3)$'tür.

ii) $d \approx r \rightarrow V_{etki} \approx (3d)^3 - d^3 = 26d^3$ (7.3)

Bu durumda V_{etki} hacmi içerisinde bulunan parçacık sayısı 26x(N/P) olacağı için haberleşme ölçeği de O(N/P) olarak belirlenmektedir.

iii) $d >> r$ (7.4)

Bu durumda V_{etki} yukarıda kullanılan formülden farklı bir hesap yöntemiyle hesaplanmaktadır. Hücrenin 6 yüzeyinin *r* kadar yakınındaki bölgelerin kapladıkları hacim yaklaşık $6d^2r$ kadardır. Bu hacim içerisinde bulunan parçacık sayısı da $6r(N/P)^{2/3}$ kadar olacaktır ve sistemin haberleşme ölçeği $O((N/P)^{2/3})$ olarak hesaplanmaktadır.



Şekil 7.4. 3 boyutlu alan ayrıştırma yöntemi hücre ve etkileşim alanı gösterimi

Benzetimi yapılan sistem 2 boyutlu olarak modellendiğinde ise 3 boyutlu ayrıştırmaya benzer hesaplamalar yapılmakta, tek fark etkileşim hacmi yerine etkileşim alanı hesabının yapılmasıdır. Kenar uzunluğu *d* olan kare şeklindeki hücrelerde (alan=d^2) N/P tane parçacık bulunmaktadır. Hücrelerin etkileşim alanları ise aşağıdaki şekilde hesaplanmaktadır:

$$A_{etki} = (d+2r)^2 - d^2 \quad (7.5)$$

i) $d << r \rightarrow A_{etki} \approx 4r^2$ (7.6)

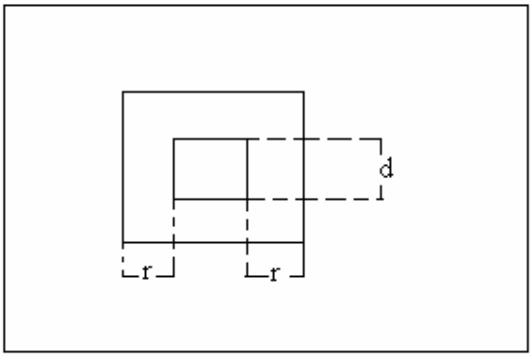
Bu durumda haberleşme ölçeği $4r^2$ ile ölçeklenmektedir, yani $O(r^2)$'dir.

ii) $d \approx r \rightarrow A_{etki} \approx (3d)^2 - d^2 = 8d^2$ (7.7)

Bu durumda A_{etki} alanı içerisinde bulunan parçacık sayısı 8x(N/P) olacağı için haberleşme ölçeği de O(N/P) olarak belirlenmektedir.

iii) $d >> r$ (7.8)

Bu durumda A_{etki} yine yukarıda kullanılan formülden farklı bir hesap yöntemiyle hesaplanmaktadır. Hücrenin 4 kenarının *r* kadar yakınındaki bölgelerin kapladıkları alan yaklaşık 4*dr* kadardır. Bu alan içerisinde bulunan parçacık sayısı da $4r(N/P)^{1/2}$ kadar olacaktır ve sistemin haberleşme ölçeği $O((N/P)^{1/2})$ olarak hesaplanmaktadır.



Şekil 7.5. 2 boyutlu alan ayrıştırma yöntemi hücre ve etkileşim alanı gösterimi

Çizelge 7.1. Koşut MD Algoritmalarının Karşılaştırılması

	Kod Karmaşıklığı	Haberleşme Ölçeği	Bellek Ölçeği	İşlemci Zaman Ölçeği
Atom Ayrıştırma	Kolay	O(N)	O(N/P)	$O(N^2/P)$
Kuvvet Ayrıştırma	Orta	$O(N/\sqrt{P})$	O(N/P)	$O(N^2/P)$
Alan Ayrıştırma	Zor	$O((N/P)^{2/3})$	O(N/P)	$O(N^2/P)$

Çizelge 7.1'de görüldüğü gibi tüm algoritmaların bellek ve işlemci zaman ölçekleri aynıdır. Hangi algoritmanın seçileceği kod karmaşıklığı ve haberleşme ölçeğine göre belirlenmektedir. Kod yazımı kolay olan atom ayrıştırma yöntemi, sistemdeki parçacık sayısı arttığında işlemci sayısından bağımsız olarak artan haberleşme yükünden (O(N)) dolayı, ancak küçük boyutlu uygulamalarda tercih edilmektedir. Büyük uygulamalarda ise haberleşme ölçeği en küçük olan ($O((N/P)^{2/3})$) alan ayrıştırma yöntemi tercih edilmektedir. Orta boyutlu uygulamalarda ise, hem kod karmaşıklığı hem de haberleşme ölçeği bakımından ($O(N/\sqrt{P})$) bu iki yöntemin arasında kalan kuvvet ayrıştırma yöntemi tercih edilmektedir.

Aşağıda örnek olarak koşut programlanmış bir MD benzetim uygulaması incelenmiştir.

7.1.3. Koşut Molekül Dinamiği Benzetim Örneği: Moldy

Moldy, C programlama dilinde yazılmış, koşut olarak çalışan bir molekül dinamiği benzetimi uygulamasıdır [21]. Atom ayrıştırma yöntemi kullanılarak modellenmiştir. Koşut işlemciler arasındaki haberleşme arayüzü MPI, PVM, BSP ve TCGMSG kütüphanelerinden istenilen seçilerek oluşturulabilmektedir.

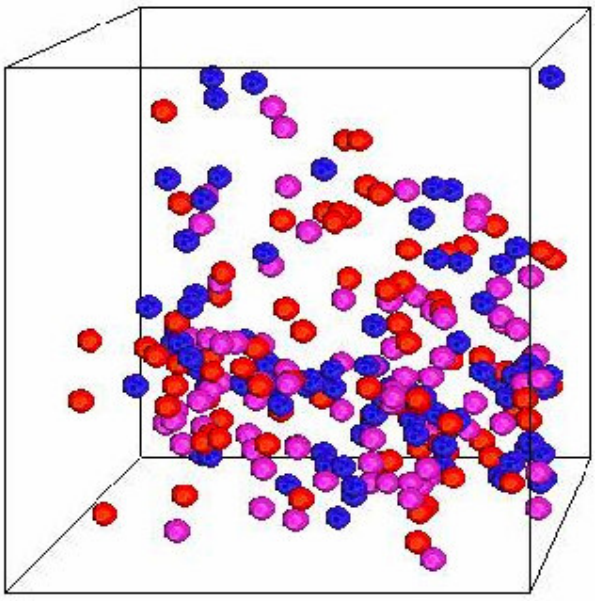
Moldy'de parçacıkların hız, ivme, sıcaklık ve konum bilgileri bağlı liste (*linked list*) veri yapısıyla tutulmaktadır. Tüm işlemcilerin tüm parçacıkların konum bilgilerinden haberdar olma zorunlulukları yüzünden, iletiler tüm işlemcilere broadcast

yöntemiyle gönderilmektedir. MPI arayüzü kullanıldığında, tüm işlemcilerden sonuçları derleyip yine tüm işlemcilere bu sonuçları gönderme işlemi `MPI_Allgather`, `MPI_Allreduce` ve `MPI_Bcast` komutlarıyla yapılmaktadır. Veri tipi olarak `MPI_BYTE` kullanılmaktadır ve veriler işlemciler arasında blok halinde gönderilmektedir.

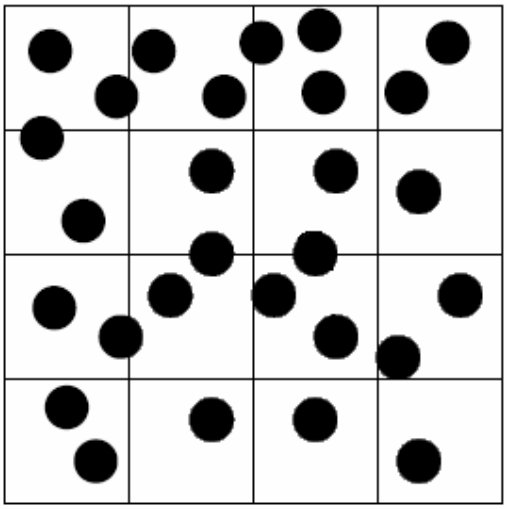
7.2. Molekül Dinamiği Benzetiminin HLA İle Olan Uygunluğunun İncelenmesi

Molekül Dinamiğinin HLA' ya uygunluğunu incelerken ilk önce dağıtılmış sistemde bilgisayarlar arasındaki iletişimin nasıl olması gerektiği belirlenmelidir. İletişim, benzetim başladığı zaman gerekli değişken değerlerinin iletilmesi, benzetim devam ederken kendi alanından ayrılıp diğer alana geçen parçacığın el değiştirmesi ve benzetim sona ererken benzetimin ne zaman bitirileceğinin iletilmesi olarak belirlenmiştir.

İkinci adım olarak benzetimde kullanılacak Federeler belirlenmiştir. Bir önceki hava tahmin benzetiminde tasarlandığı gibi bu benzetimde de, federeler alan ve hacim olarak bulundukları bölgeden sorumlu olacaklardır. Veriler işlemciler arasında blok halinde gönderilecektir. Şekil 7.6'da moleküllerin üç boyutlu olarak gösterimi ve Şekil 7.7'de de benzetimi tasarlanan alan gösterilmiştir.



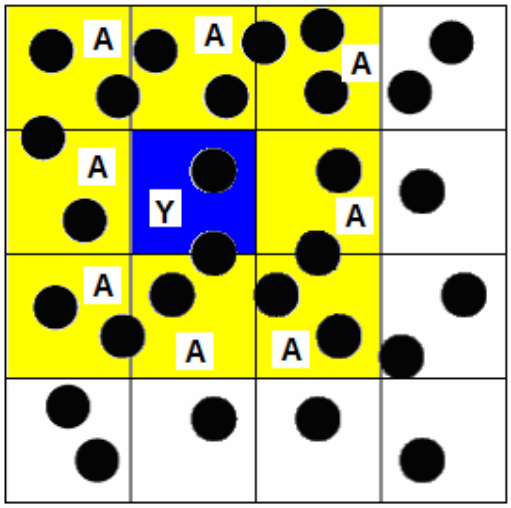
Şekil 7.6. Üç Boyutlu Molekül Gösterimi



Şekil 7.7. Benzetimi Tasarlanan Alan

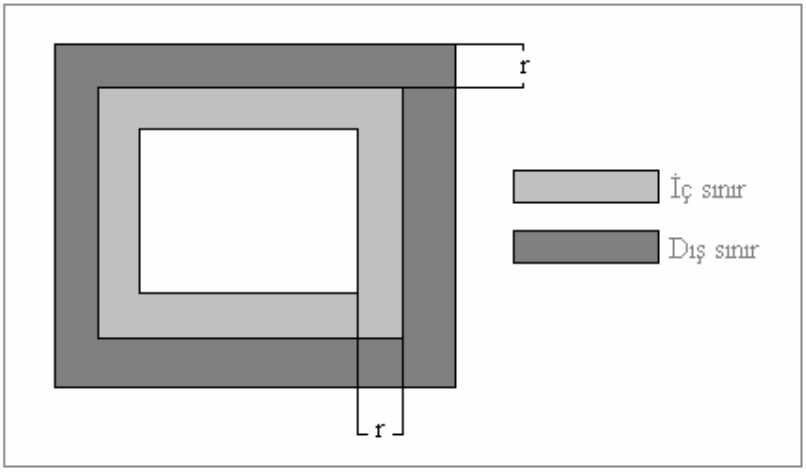
Şekil 7.7'de gösterilen alanda moleküller zamana bağlı olarak serbestçe hareket etmektedirler. Moleküllerin hareketleri esnasında birbirleri ile çarpışma durumları ve bitişik iki molekülün taşıdığı elektrostatik yükten kaynaklanan "Van der Vaals" adı verilen çekim kuvvetleri vardır. Moleküllerin hareketlerinde etkili olan hız, sıcaklık, ivme ve konum bilgileri "Alan_Federe" adı verilen birden çok federe tarafından kontrol edilmektedir.

Federeler arasındaki iletişim hava tahmin benzetiminde olduğu gibi abone olma ve yayınlama mekanizmaları tarafından sağlanmaktadır. Bu mekanizma federelerin yayınlamak istedikleri alanı belirlemeleri ve diğer federelerin bu alana abone olarak bilgileri almaları şeklinde işler. Şekil 7.8'de federelerin yayınlama ve abone olma mekanizmaları gösterilmiştir.



Şekil 7.8. Yayınlama ve Abone Olma Mekanizması

Hücrelerin içinde bulunan parçacıkların etkileşim hesapları hücreler arasındaki mesafeye bağlıdır. Eğer hücreler arasındaki mesafe kesme mesafesinden (*cut-off distance*) fazla ise, hücrelerin birbirleriyle etkileşmedikleri varsayılmaktadır. Dolayısıyla, bir hücre sınırlarına kesme mesafesi kadar yakın bölgedeki parçacıkların bilgileriyle ilgilenmekte, daha uzaktakilerle ilgilenmemektedir. Aynı zamanda komşu hücrelerin de bir hücre içerisinde ilgilendikleri alan o hücrenin sınırlarının kesme mesafesi kadar iç kısmında kalan bölgesidir. Bu durum Şekil 7.9'da bir hücrenin komşu hücrelerde ilgilendiği alanlar *dış sınır,* komşu hücrenin bu hücre içerisinde ilgilendikleri alan *iç sınır,* kesme mesafesi de *r* olarak gösterilmiştir. Her bir hücre yayın yapacağı zaman kendi iç sınırında kalan parçacıklara ait bilgileri göndermesi planlanmıştır. Bu sayede abone olan hücrelerin de abone oldukları hücrelerden sadece ilgilendikleri alana ait parçacık bilgilerinin gelmesi sağlanmış olacaktır.

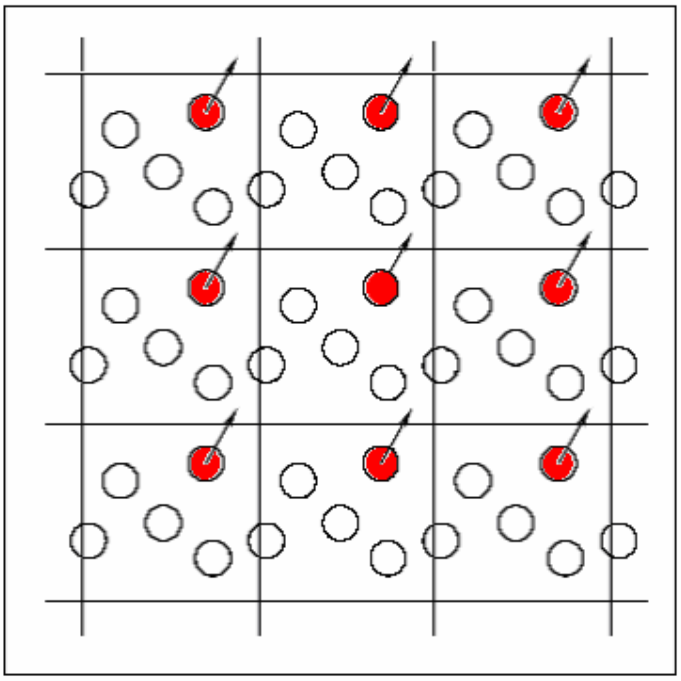


Şekil 7.9. Hücrelerin bilgilerini yayınladıkları parçacıkların bulunduğu bölge (iç sınır) ve bilgilerini aldıkları parçacıkların bulunduğu bölge (dış sınır)

7.3. Parçacıkların Devredilmesi ve Günlenmesi

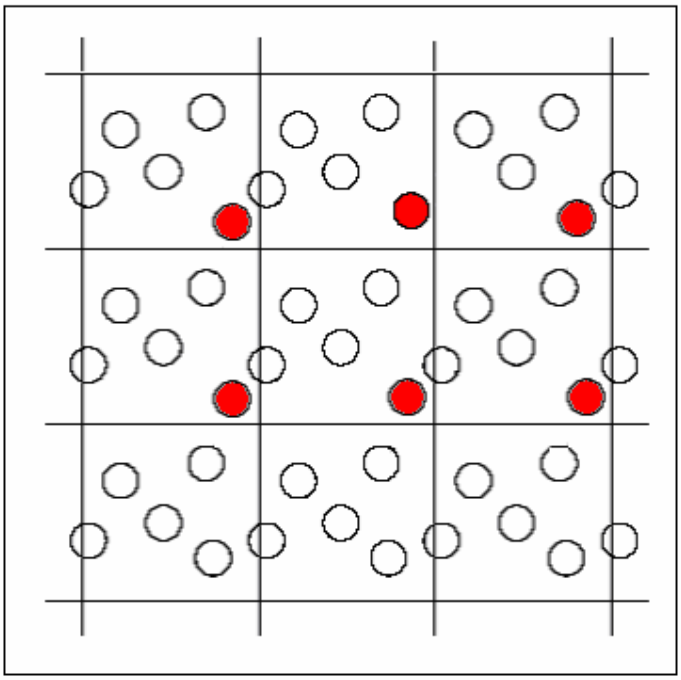
Parçacığın kendi alanından çıktığını anlayan federe, bu parçacığın sahipliğini bıraktığını `release_ownership()` işleviyle RTI'ya bildirir. Bu bildirim yapılırken parçacığın konumu da bu iletinin bir parametresi olarak diğer federelere iletilir. Her federe bu konum değerini kendi alanıyla karşılaştırır. Eğer kendi alanı içerisinde olduğunu tespit ederse, bu parçacığı kendi sahipliğine almak için gerekli iletileri gönderir (*`acquire_ownership()`*). Bütün bu işlemler sonunda RTI, parçacığın sahibinin değiştiği bilgisini her iki federeye de gerekli iletilerle haber verir.

Şekil 7.10'da parçacıkların yer değiştirme durumları gösterilmiştir.



Şekil 7.10. Parçacıkların Yer Değiştirme Durumları

Parçacıkların yer değiştirmelerine sebep olan etmenler sıcaklık değişiminden kaynaklanan enerji, pozitif ve negatif yüklü moleküllerin birbirlerini çekmeleri sonucu oluşan çarpışmalardır. Yer değiştiren moleküllerin durumları Şekil 7.11'de gösterilmiştir.



Şekil 7.11. Yer Değiştiren Parçacıklar

7.4. Molekül Dinamiği Modellemesi için Benzetim Nesne Modeli (SOM)

Hava tahmin modeli ve molekül dinamiği modeli uygulamalarının etkileşimleri ve nesneleri benzerlik gösterdiği gibi federelerinin alan ve hacim bakımından incelenmesi söz konusu olduğu için molekül dinamiği modellemesi için tasarlanan SOM, hava tahmin modellemesi için tasarlanan SOM'un aynısıdır. Sadece içerisindeki bilgiler molekül dinamiği modellemesine uyarlanmıştır. Aşağıdaki çizelgelerde sırasıyla nesne modeli kimlik çizelgesi, nesne sınıfı yapı çizelgesi ve zaman gösterim çizelgesi gösterilmiştir.

Çizelge 7.2. Nesne Modeli Kimlik Çizelgesi

Nesne Modeli Kimlik Çizelgesi	
Kategori	**Bilgi**
İsim	MolekülDinamiğiModeli
Tür	SOM
Versiyon	1.0 Alpha
Değişme Tarihi	13 Eylül 2007
Amaç	Molekül Dinamiği federesi için nesne modeli
Uygulama Alanı	Molekül Dinamiği işlemleri
Sponsor	Hacettepe Üniversitesi
Kişi	Tülin Erçelebi
Organizasyon	ABC
Telefon	2977500-119
E-mail	tulin@cs.hacettepe.edu.tr
Referans	www.kayhan_imre.com
Diğer	Diğer MD uygulamaları incelenebilir

Çizelge 7.3. Nesne Sınıfı Yapı Çizelgesi

		Nitelikler
HLAobject Root (N)	Parçacık (AY)	İvme (AY)
		Sıcaklık (AY)
		Konum (AY)
		Hız (AY)

Çizelge 7.4. Zaman Gösterim Çizelgesi

Zaman Gösterim Çizelgesi		
Kategori	**VeriTürü**	**Anlamsal**
Zaman Damgası	ZamanTürü	Negatif olmayan bir Δt değerine sahiptir
Lookahead	LookaheadTürü	Negatif olmayan bir Δt değerine sahiptir

Sonuç olarak, yapılan incelemeler ve tasarımlar itibariyle bu uygulamanın da HLA ile modellenmesinin uygun olduğu tespit edilmiştir.

7.5. Moldy ve HLA ile tasarlanan MD benzetimi karşılaştırması

Koşut olarak modellenen MD benzetimi programı Moldy ile HLA ile tasarlanan MD benzetimi Çizelge 7.5'de sözde kodları verilerek kavramsal olarak karşılaştırılmıştır.

Çizelge 7.6'da ise MD benzetiminde kullanılan algoritmalar ve HLA ile tasarlanan MD benzetimi, başarım, yeniden kullanılabilirlik, ölçeklenebilirlik, kolay gerçekleştirilebilirlik ve birlikte kullanılabilirlik kıstasları açısından karşılaştırılmıştır.

Çizelge 7.5. Moldy ve HLA ile tasarlanan MD Benzetimi sözde kodları

Moldy'nin sözde kodu (*pseudo code*) :	**HLA ile tasarlanan MD benzetimi sözde kodu:**

ilk_degerleri_ata();
while (! benzetim_sonu)
{
parcacik_bilgileri_guncelle();
etkilesim_hesaplarini_yap();
sonuc_derle();
broadcast();
}
sonlandir();
	RTI :
	federasyon_yarat();
	while (! benzetim_sonu)
	{
	federe_listesi_guncelle();
	yayin_listesi_guncelle();
	while (! başarili_alim)
	{
	reflect : federeleri_haberdar_et();
	bekle();
	}
	
	Federe :
	benzetime_katil();
	bilgileri_guncelle();
	yayin_listesi_gonder();
	abone_listesi_gonder();
	while(! benzetim_sonu)
	{
	etkilesim_hesaplarini_yap();
	bilgileri_yayinla();
	while (! reflect)
	bilgileri_al();
	attribute_guncelle();
	if(yeni parcacik)
	acquire_ownership();
	if (giden parcacik)
	release_ownership();
	}

HLA ile tasarlanan MD benzetiminde zaman yönetimi, daha önce hava tahmin benzetimi tasarımında olduğu gibi, `federeleri_haberdar_et()` (reflect) işlevi aracılığıyla otomatik olarak sağlanmaktadır. Etkileşim hesaplarını yapıp, sonuçlarını yayınlayan federe, diğer federelerin de sonuçlarını yayınlayıp, daha sonra RTI'nın sağladığı veri dağıtım yönetimi servisi tarafından ilgilendiği bilgilerin başarılı bir şekilde kendisine gönderilmesini bekler. Tüm federeler güncellemelerden başarılı bir şekilde haberdar olmadan, yeni yayın alımı yapılmaz.

Çizelge 7.6. Moldy ve HLA ile tasarlanan MD benzetimi karşılaştırması

	MPI ile MD			HLA ile MD
	Atom Ayrıştırma	Kuvvet Ayrıştırma	Alan Ayrıştırma	
Başarım Analizi	X	Sınırlı	√	√
Ölçeklenebilirlik	X	Sınırlı	√	√
Kolay gerçekleştirilebilirlik	√	Sınırlı	**X**	√
Yeniden kullanılabilirlik	**X**	**X**	**X**	√
Birlikte kullanılabilirlik	**X**	**X**	**X**	√

Atom ayrıştırma yönteminde tüm işlemcilerin, tüm parçacıkların konum bilgilerinden haberdar olma zorunlulukları bu yöntemde veri iletiminin oldukça fazla zaman tutmasına neden olmaktadır. Bu bakımdan başarım analizinde diğer yöntemlere göre en başarısız olarak değerlendirilmiştir. Kuvvet ayrıştırma yönteminde atom ayrıştırma yöntemine oranla daha az haberleşme gereksinimi olmakla birlikte, sadece komşularıyla haberleşme ihtiyacı olan alan ayrıştırma yöntemi kadar kısa sürede veri iletimi sağlanamamaktadır. HLA, 2 boyutlu alan ayrıştırma yöntemini kullanmaktadır. Bu durumda haberleşme ölçeği yukarıda hesaplandığı gibi $O((N/P)^{1/2})$ olacaktır. Ayrıca bellek ölçeği ve işlemci zaman ölçeği koşut molekül dinamiği benzetiminde kullanılan tüm yöntemlerdeki gibi sırasıyla $O(N/P)$ ve $O(N^2/P)$ olacaktır. Parçacıklar topluca blok halinde güncellenmektedir. HLA bununla birlikte veri dağıtım yönetimi servisi sayesinde daha etkin bir haberleşme seçeneği sunacaktır.

Atom ayrıştırma yöntemi, haberleşme ölçeğinin işlemci sayısından bağımsız bir şekilde sadece parçacık sayısına bağlı olması sebebiyle ($O(N)$), parçacık sayısı çok artırıldığında haberleşme yükü benzetimin çalışmasının önemli ölçüde sınırlandıracaktır. Bu bakımdan bu yöntem ölçeklenebilir olarak değerlendirilmemiştir. Kuvvet ayrıştırma yöntemindeyse $O(N/\sqrt{P})$ olan haberleşme ölçeği, parçacık sayısı artışı, işlemci sayısı artışıyla aynı oranda dengelenemediği için sınırlı bir ölçeklenebilirlikten bahsedilebilir. MPI ile gerçeklenen MD benzetimlerinde alan ayrıştırma yöntemi $O((N/P)^{2/3})$ haberleşme ölçeğiyle, parçacık ve işlemci sayısı aynı oranda artırıldığında aynı başarımı gösterebilmekte

ve bu yüzden ölçeklenebilir özelliği taşımasıyla diğer benzetim yöntemlerinden ayrılmaktadır.

MPI'nın tamamıyla platformdan bağımsız olmayışı ve işlemcilere yük dağılımını otomatik olarak yapamayışı sebebiyle, farklı uygulamalar arasında birlikte çalışabilirliği ve birlikte kullanılabilirliği kolaylıkla sağlayamamaktadır [22].

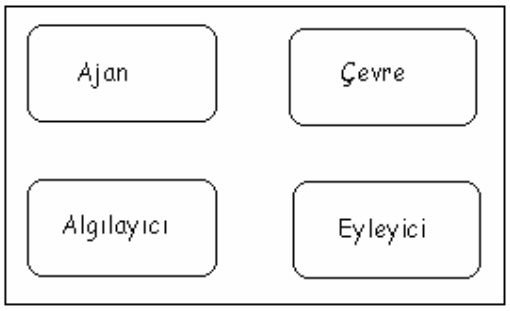
Ölçeklenebilirlik, birlikte çalışabilirlik ve yeniden kullanılabilirlik değerlendirmelerinde HLA'nın önemli bir üstünlüğü söz konusudur.

8. AKILLI AJAN (INTELLIGENT AGENT) BENZETİMİ

Akıllı ajan kullandığı algılayıcılar (*sensors*) sayesinde çevreyle bağlantı kurup, çevrede neler olup bittiğini gözlemleyen ve bu algılara göre eylemlerini belirli bir hedef doğrultusunda gerçekleştiren yapay zeka disiplininin uygulamalı bir koludur.

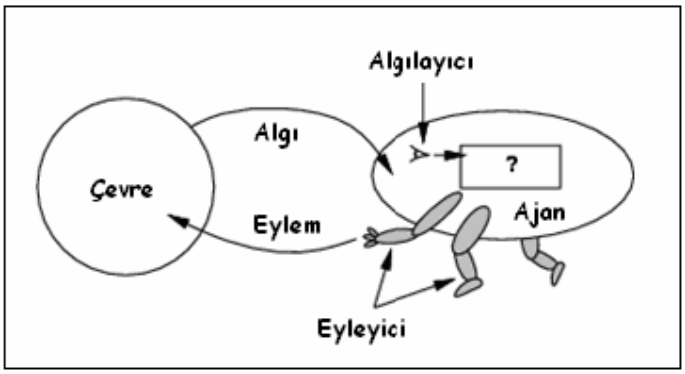
Akıllı ajan teknolojisinin, otonom bilgi ajanları ve çok ajanlı sistemler olmak üzere iki temel alanda ilerleme sağladığı gözlemlenmektedir. Otonom bilgi ajanları, önceden tanımlanmış işlevlerin kullanıcı müdahalesi olmadan insan üzerindeki iş ve bilgi yükünü azaltan ajanlardır. Çok ajanlı sistemler ise daha çok kullanıcı tarafı ile sınırlı kalmayan ve her biri bir taraf adına hareket eden bir grup ajandan oluşmaktadır.

Bu tez kapsamında incelenecek olan akıllı ajan uygulamasının dört temel unsuru bulunmaktadır. Bu unsurlar ajan, algılayıcı, eyleyici (*actuator*) ve çevredir. Örneğin; insan ajanının algılayıcıları gözleri, kulakları olabilirken elleri, ayakları, dudağı eyleyici olarak düşünülebilir. Uygulamanın işleyişine göre ajan algılayıcıları vasıtasıyla çevrede olup bitenlerden haberdar olduktan sonra algılarını eyleyiciye göndermektedir. Eyleyici alınan algılara göre değerlendirme yapıp bir sonraki adımın ne olması gerektiğini belirlemektedir. Şekil 8.1'de akıllı ajanın bu dört unsuru görülmektedir.



Şekil 8.1. Akıllı Ajan Unsurları

Bu dört unsurun birbirleri ile olan iletişimleri Şekil 8.2'de gösterilmiştir.



Şekil 8.2. Unsurlar Arasındaki İletişim

8.1. Akıllı Ajan Uygulamasının Koşut Programlama ile Modellenmesi

Ajan tabanlı uygulamalar robotik, yapay zeka ve çevre izleme sistemleri gibi değişik alanlarda sıklıkla kullanılmaktadır.

Akıllı ajan uygulamaları tek bir ajandan oluşabileceği gibi birden fazla ajandan da oluşabilir. Birden fazla ajanın oluşturduğu sisteme çoklu ajan (multi-agent) adı verilir. Çoklu ajan uygulamasına örnek olarak iki ajanın karşılıklı olarak satranç oynaması verilebilir. IBM'nin tasarladığı derin mavi (*Deep Blue*) ajanı ise tek ajanlı uygulamalara örnek olabilir. Bu uygulama dünyaca ünlü satranç oyuncusu Kasparov'u altı oyundan sonra yenmeyi başarmıştı [36].

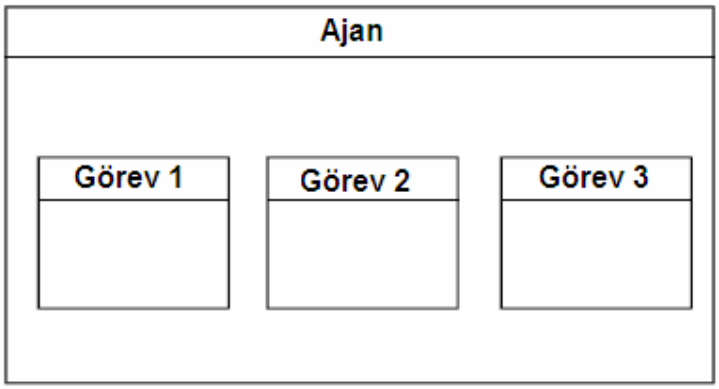
Ajan ve ajan tabanlı uygulamaların modellenebilmesi için FIPA (*Foundation for Intelligent Physical Agents*) tarafından bir standart geliştirilmiştir [16]. FIPA standardı mimarisi "FIPA Soyut Mimari Belirtimi" ("*FIPA Abstract Architecture Specification*") adlı dokümanda tanımlanmıştır [17]. FIPA'ya uygun bir sistem, iki ajanın ortak bir platforma nasıl kayıt olacağını ve birbirleriyle nasıl bir ileti alışverişi içerisinde bulunacağını tanımlayan soyut bir mimari gerçekleştirmelidir.

Standartta tanımlanan iki temel servis aşağıdaki gibidir:

1) Dizin servisi (*directory-service*): bir ajanın servis tanımından yola çıkarak diğer ajanları bulmasını sağlar.

2) İleti taşıma servisi (*message-transport-service*): iletilerin bir platformdan diğerine ve en nihayetinde hedef ajana taşınmasını sağlar.

FIPA standardına göre bir ajan birden fazla göreve (task) sahip olabilir, diğer ajanlarla bilgi alış-verişinde bulunabilir ya da algılayıcılardan veri elde edebilir. Ajanın sahip olduğu görevler ajanı oluşturan temel parçalardan biridir. Bu görevlere örnek olarak sağa/sola dönme, durağan görüntüyü elde etme ve diğer ajanlara ileti gönderme verilebilir. Şekil 8.3'de üç görevli ajanın yapısal olarak gösterimi verilmiştir.

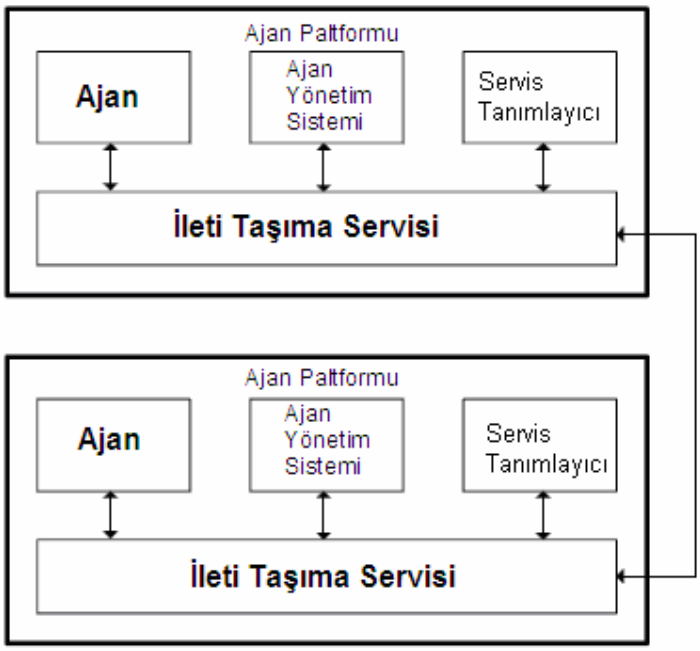


Şekil 8.3. Üç görevli ajanın yapısal olarak gösterilmesi

Ajanlar arası iletişim iletiler yoluyla sağlanmaktadır. Her iletinin kendine özgü (*unique*) göndericisi, alıcısı ve içeriği olmak durumundadır. Bir ajan ileti gönderdiğinde, iletinin yerine ulaşması için gerekli olan bilgiler (gönderici/alıcı adresleri ve iletinin taşınma tipi detayları) ileti taşıma servisi tarafından düzenlenir. İleti taşıma servisi bu işlemleri yapabilmek için http (*Hypertext Transfer Protocol*), SMTP (*Simple Mail Transfer Protocol*), IIOP (*Internet Inter-Orb Protocol*) gibi haberleşme protokolleri kullanır [16].

Bir ajanın bulunduğu platformda varlığını sürdürebilmesinin kuralları "FIPA Ajan Yönetimi Belirtimi"nde (*"FIPA Agent Management Specification"*) tanımlanmıştır [34]. Şekil 8.4'de iki ajanlı bir sistemin FIPA ajan yönetimi modeli gösterilmiştir. Şekilde görülen Ajan platformu (*Agent Platform*) ajanların yer aldığı yazılım alt yapı sistemidir. Bu sistem işletim sistemi tarafından oluşturulur. İleti taşıma servisi ise farklı ajan platformları üzerinde olan ajanların birbirleri ile haberleşmeleri için kullanılan bir yöntemdir. Servis tanımlayıcı (*directory facilitator*) olarak adlandırılan yapı ajanların servis bilgilerini tutar. Bir ajan kendi görevi doğrultusunda bir başka ajanla haberleşmek istediğinde kendi işine yarayacak servislerin o ajanda olup

olmadığını servis tanımlayıcı aracılığıyla öğrenir. Hangi ajanın ne tür görevi olduğunu anlayabilmek için de ajan tanımlayıcı (*agent identifier*) etiketi vardır. Ajan sisteme giriş yaptığı andan itibaren geçerli olan bir etiket bilgisi ajana gönderilir.



Şekil 8.4. FIPA Ajan Yönetimi Modeli

Akıllı ajan uygulamalarında tek bir ajan varsa bu uygulamaların koşut olarak modellenmesine gerek yoktur [15]. Ancak çoklu ajan uygulamalarında, her bir ajanın oldukça fazla bellek ve işlemci kapasitesine ihtiyaç duyması ve toplam çalışma zamanının çok fazla olması sebebiyle, çoklu ajan uygulamalarının başarımını artırmak için bu uygulamaların koşut olarak modellenmesini gerekli kılmaktadır. Akıllı ajan uygulamalarında koşutluğu sağlayabilmek için görevler küçük iş parçacıklarına bölünerek her bir iş parçacığından sorumlu işlemciler atanmaktadır. Bu işlemciler arasındaki yükün dağılımı (*load balancing*) mümkün olduğunca eşit olacak şekilde dağıtılmalıdır [16].

Koşut ortamda modellenen çoklu ajan uygulamalarında haberleşme kütüphanesi olarak genellikle MPI kullanılmıştır. MPI ile gerçekleştirilen ajan tabanlı uygulamalarda MPI'nın iki önemli problemiyle karşılaşılmıştır. Bunlardan ilki MPI'nın işlemciler arasındaki görev paylaşımını otomatik olarak

gerçekleştirememesi, ikincisi ise MPI'nın platformdan tamamen bağımsız olmayışı nedeniyle farklı uygulamalar arasındaki birlikte çalışabilirliği kolay sağlayamaması ve kodların taşınabilir (*portable*) olmayışıdır [22]. Bu dezavantajların sonucu çoklu ajan tabanlı uygulamalarda başarım yetersizliği de gözlenmektedir [24].

8.2. Akıllı Ajan Benzetiminin HLA İle Olan Uygunluğunun İncelenmesi

HLA'nın model tabanlı yaklaşımı ve farklı türdeki modellerin ve benzetimlerin birlikte çalışabilirliğini desteklemesi, HLA'yı çoklu ajan uygulamaları için dikkat çekici bir seçenek olarak sunmaktadır. Ayrıca RTI benzetimlerin genel olarak ihtiyaç duyduğu zaman yönetimi, veri dağıtım yönetimi gibi servisleri otomatik olarak sağladığı için, sistem modellemelerini kolaylaştırmaktadır [23]. Zaman yönetimi servisi federelerin sahip olabileceği farklı zaman modellerini uyumlu (*synchronized*) hale getirebilmekte ve federasyon için sanal evrensel bir zaman sağlayabilmektedir. Bu da ajanların tek tek farklı yerel zamanlara sahip olsalar da federasyonun ortak zamanı sayesinde nedenselliği sağlayabilmelerini sağlamaktadır [23].

Bunların yanı sıra, HLA'nın geliştirilmesindeki birincil amaç ajan tabanlı uygulamalar olmadığı için bazı eksikliklerin olması doğaldır. Bu eksikliklerden bir tanesi HLA'ya KQML (*The Knowledge and Query Manipulation Language*) eklenerek giderilmeye çalışılmıştır. KQML, ajanlar arasındaki bilgi alış verişini destekleyen ve haberleşme dilleri arasında (FIPA-ACL, KQML, KIF) son yıllarda standartlaşmaya başlayan bir dildir [23].

HLA ile gerçekleştirilmiş birçok akıllı ajan uygulaması mevcuttur [23][24][35]. Bunlardan havaalanı benzetimi [35] HLA'nın birlikte çalışabilirliğini iyi bir şekilde yansıttığı için örnek olarak incelenmiştir. Havaalanı benzetimi altı federeden oluşmaktadır. Bunlar aşağıdaki gibidir:

i) Kamera federesi: havaalanındaki tüm kameraların benzetiminden sorumludur.

ii) Yer radarı federesi: uçağın fark edilmesini ve konumunun belirlenmesi işlevini yerine getiren yer radarlarının davranışsal modelidir.

iii) Hava federesi: mevcut hava durumu bilgilerini modellemektedir.

iv) Uçak federesi: uçak veya uçakların havaalanı yerleşkesindeki hareketini modellemektedir.

v) Planlama federesi: uçağa hareket komutlarını gönderen denetleyici davranışını modellemektedir.

vi) Animasyon federesi: benzetimin gerçek zamanlı davranışını gösterir.

Benzetim çalışmadan önce kullanıcı arayüzü aracılığıyla, benzetimin yapılacağı havaalanı, havaalanının yerleşimi, uçuşlar hakkındaki bilgiler (kalkış/varış zamanı, kapı numarası, iniş/kalkış apronu), hava durumu, algılayıcıların yerleşimi ve benzetimin çalıştırılacağı süre gibi bilgiler girilmektedir. Hava durumu basit olarak zaman (gece/gündüz), mevsim (kış/yaz), görüş mesafesi, meteorolojik durum (açık/yağmurlu/sisli) gibi bilgileri içermektedir.

Benzetimde zaman yönetimi olay tabanlı olarak gerçekleştirilmektedir. Bir olay zaman damgalı bir durumu (uçağın kalkış veya varış zamanı, ya da uçağın havaalanın belli bir noktasından geçiş zamanı gibi) göstermektedir. Federasyonun çalışması esnasında zaman damgaları uçak federesi tarafından, benzetimi yapılan uçak sayısı, her bir uçağın konumu ve hızı, havaalanın planına göre hesaplanmaktadır. Her bir zaman adımında kamera federesi havaalanı üzerindeki uçakların günlenmiş görüntülerini alır. Yine aynı şekilde yer radarı federesi algılanan uçak bilgilerini günler. Uçak federesi, alınan komutlara göre sorumluluğundaki uçakların hareketini gerçekleştirir. Planlama federesi, kamera ve yer radarından alınan uçakların yaklaşık veya tam konumlarına göre karmaşa önleyici bir algoritma çalıştırır ve uçak federesine, uçaklar için “dur” ya da “git” gibi komutlar gönderir. Animasyon federesi ise benzetimi yapılan nesnelerin havaalanı üzerindeki davranışlarının çevrimiçi olarak görüntülenmesini sağlar. RTI’nın günlenmiş nitelikleri yansıtmasıyla (reflect) birlikte uçakların bir sonraki konum bilgilerine göre uçakları havaalanı üzerinde hareket ettirir.

Havaalanı benzetiminin anlatıldığı çalışmada sonuç olarak, HLA’nın açık ve ölçeklenebilir bir benzetim altyapısı sunmasıyla evrensel havaalanı modellemesi ve benzetimine olanak sağladığına vurgu yapılmıştır [35]. Ayrıca havaalanı federasyonunun, HLA’nın birlikte çalışabilirlik ve yeniden kullanılabilirlik hususlarında ihtiyaçları karşıladığının önemli bir göstergesi olduğuna dikkat çekilmiştir [35].

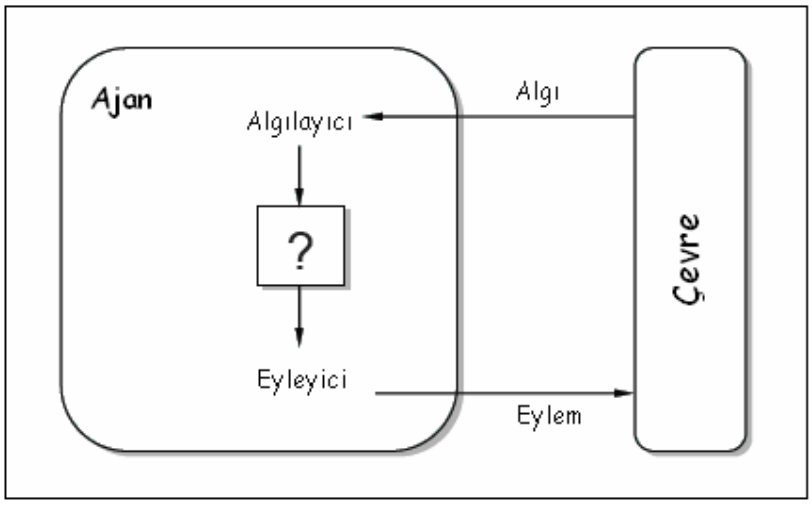
Çizelge 8.1'de yaygın kullanılan bazı akıllı ajan modelleri örnek teşkil etmesi bakımından verilmiştir. Bu çizelgedeki akıllı ajan modellerinden taksi sürücüsü ajan modeli tez kapsamında tasarlanmak üzere seçilmiştir. Tasarımı anlatılan taksi sürücüsü ajanının birden fazla sayıda aynı ortamda benzetiminin yapılmasıyla çoklu ajan uygulaması gerçekleştirilmiş olacaktır.

Çizelge 8.1. Örnek Akıllı Ajan Modelleri

Ajan Adı	Algılar	Eylemler	Hedefler	Çevre
Tıbbi teşhis sistemi	Bulgular, hastanın cevapları	Sorular,testler, tedaviler	Sağlıklı hasta, düşük maliyet	Hasta, hastane
Taksi sürücüsü	Kameralar, takometre, GPS	Durmak, yavaşlamak, harekete geçmek, müşteri almak, müşteri indirmek	Güvenli, yasal yolculuk	Yollar, müşteriler,diğer ajanlar, trafik ışıkları
Uçuş kontrol sistemi	Hava durumu algılayıcıları, yer radarı, kamera	Durmak, kalkmak, inmek	Güvenli kalkış, güvenli iniş	Diğer uçaklar, pistler

Taksi sürücüsü ajanının kendisi ile beraber dört tane federesi vardır. Şekil 8.5'te ajanın algılayıcıları, eylemleri, hedefleri ve çevresi görülmektedir.

Bu unsurlara göre, taksi sürücüsü kameralar ve diğer algılayıcıları sayesinde yollardan, trafikten ve müşterilerden haberdar olacaktır. Algılayıcılardan algı olarak geri dönen veriler değerlendirme aşamasından sonra eyleyiciye gönderilerek bir sonraki aşamaya geçilmiş olunmaktadır. Eyleyiciye gelen veride hızlanmak, durmak, fren yapmak gibi eylemler olarak çevreye iletilmektedir. Bu iletişim Şekil 8.5'te görülmektedir.



Şekil 8.5. Federeler Arası İletişim

Taksi sürücüsü akıllı ajanı dört federeden oluşan bir federasyon nesne modeline (FOM) sahiptir. Şekil 8.6'da taksi sürücüsü ajanının federeleri ve FOM'u gösterilmektedir.

Şekil 8.6. Federeler ve FOM

8.2.1. Yayınlama ve Abone Olma Mekanizması

Taksi sürücüsü ajanı, tüm federelerden haberdar olması gerektiği için algılayıcı, eyleyici ve çevre federelerinin niteliklerine hem abone olmalı hem de yayınlamalıdır. Algılayıcı federesi, çevreyi keşfedip bu algıları eyleyiciye ileteceği için çevre federesinin niteliklerine abone olmalı ve eyleyici federesinin niteliklerini de yayınlamalıdır. Çevre federesi, eyleyici federesinin niteliklerine abone olmalı ve algılayıcı federesinin niteliklerini de yayınlamalıdır. Son olarak eyleyici federesi ise,

algılayıcı federesinin niteliklerine abone olmalı ve çevre federesinin niteliklerini yayınlamalıdır. Bu durum Çizelge 8.2'de gösterilmiştir.

Çizelge 8.2. Yayınlama ve Abone Olma Mekanizması

Ajan	Algılayıcı	Eyleyici	Çevre
Algılayıcı (A,Y)	Çevre (A)	Algılayıcı (A)	Algılayıcı (Y)
Eyleyici (A,Y)	Eyleyici (Y)	Çevre (Y)	Eyleyici (A)
Çevre (A,Y)	Ajan (Y)	Ajan (Y)	Ajan (Y)

8.2.2. Akıllı Ajan Modellemesi için Benzetim Nesne Modeli (SOM) ve Federasyon Nesne Modeli (FOM)

Diğer uygulamalarda yapıldığı gibi akıllı ajan uygulamasında da benzetim nesne modeli için gerekli olan çizelgeler gösterilecektir. Bu çizelgeler arasında önceki uygulamalarla benzerlik gösteren ya da tamamen aynı olan çizelgeler tekrar gösterilmeyecektir. Aşağıdaki çizelgelerde her federe için tanımlanan nesneler ve etkileşimler gösterilmiştir.

Çizelge 8.3. Algılayıcı Federesi için Nesne Sınıfı Yapı Çizelgesi

		Nitelikler
HLAobject Root (N)	Algılayıcı (AY)	Trafik Kamerası (Y)
		Takometre (Y)
		GPS (Y)

Çizelge 8.4. Çevre Federesi için Nesne Sınıfı Yapı Çizelgesi

		Nitelilkler
HLAobject Root (N)	Çevre (AY)	Trafik Işıkları (Y)
		Yayalar (Y)
		Müşteriler (Y)
		Diğer Taksiler (Y)

Çizelge 8.5. Eyleyici Federesi için Nesne Sınıfı Yapı Çizelgesi

			Parametreler
HLAobject Root (N)	Eyleyici (AY)	Eylem (Y)	Yavasla (Y)
			Dur (Y)
			YeniHedef Belirle (Y)
			Harekete Gec (Y)
			Musteri Al (Y)
			Musteri Indir (Y)

Çizelge 8.6. Algılayıcı Federesi için Etkileşim Sınıfı Yapı Çizelgesi

			Parametreler
HLAinteraction Root (N)	Algılayıcı Durumu (Y)	CevreyiKesfet (A)	EngelVarMi (AY)
			MusteriVarMi (AY)
		Eyleyiciye Gonder (Y)	EngelDurumu (AY)
			MusteriDurumu (AY)

Çizelge 8.7. Çevre Federesi için Etkileşim Sınıfı Yapı Çizelgesi

		Parametreler
HLAinteraction Root (N)	Cevre Durumu (Y)	CevreDurumu Gonder (Y)
		EylemAl (A)
		EngelDurumu Gonder (Y)
		MusteriDurumu Gonder (Y)

Çizelge 8.8. Eyleyici Federesi için Etkileşim Sınıfı Yapı Çizelgesi

		Parametreler
HLAinteraction Root (N)	Eyleyici Durumu (Y)	AlgıAl (A)
		EylemBildir (Y)

8.3. HLA ile Tasarlanan Taksi Sürücüsü Benzetiminin Davranış Modeli

Aşağıdaki sözde kodda HLA ile tasarlanan taksi sürücüsü benzetiminin davranış modeli verilmiştir.

```
RTI :
federasyon_yarat();
while (! benzetim_sonu)
{
   federe_listesi_guncelle();
   yayin_abonelik_listesi_guncelle();
   while (! basarili_alim)
{
   reflect : federeleri_haberdar_et();
bekle();
}
}

Ajan Federesi :

benzetime_katil();
harekete_gec();
while(! benzetim_sonu)
```

```
{
   algilayici_ciktilarini_derle();
   if (takside_musteri_yok)
   {
        if(yeni_musteri)
        {
           dur();
           musteri_al();
           yeni_hedef_belirle();
           harekete_gec();
         }
     }
    if (engel_var)
    {
       if(engel_mesafesi>=5m)
          yavasla();
       else
          dur();
       }
  else
   {
      if(hız==0)
        harekete_gec();
    }
if(hedef_mesafesi<=30 m)
 {
    if(hedef_mesafesi==0 m)
     {
        dur();
        musteri_indir();
        harekete_gec();
     }
    else
        yavasla();
   }
}
```

Algılayıcı Federesi :

```
while(!benzetim_sonu)
{
    cevreyi_tara();
    attribute_yayinla();
}
```

Eyleyici Federesi :

```
while(!benzetim_sonu)
{
    komutlari_derle();
```

```
    attribute_guncelle();
    attribute_yayinla();
  }
```

Çevre Federesi :

```
while(!benzetim_sonu)
{
    attribute_guncelle();
    if(rand() <= 0.3)
      yeni_musteri_yarat();
    attribute_yayinla();
}
```

Yukarıda verilen sözde kodda, taksinin hızı 50 km/s olduğu ve 3 çeşit engel olduğu varsayılmıştır. Bu engeller diğer ajanlar (diğer taksiler), trafik ışıkları ve yayalar olarak düşünülmüştür. Sözde kodda geçen "attribute_guncelle()" işlevi içinde müşteri bilgileri, diğer ajanların konumları ve engellerin durumu gibi verileri barındırmaktadır. Ayrıca, "rand()" 0-1 arasında rastgele ondalıklı sayı üreten bir işlevdir.

Zaman yönetimi, zaman adımlı (*time stepped*) olarak tasarlanmış olup davranış modellemesi aşağıdaki gibidir.

```
zaman_bilgisi_ilk_atamasini_yap(); // initialize
while(!benzetim_sonu)
{
  while(!basarili_alim)
  {
    reflect: federeleri_haberdar_et();
    bekle();
  }
 zaman_ilerlet();
}
```

Yapılan tasarımda, her bir zaman adımı 100 ms. olarak seçilmiştir. Bu durum 50 km/s hızla giden bir araç için yaklaşık 1.5 m'lik bir ilerlemeye karşılık gelmektedir. Her şey bu zaman dilimi göz önüne alınarak günlenmektedir.

Taksi sürücüsü benzetiminde uygulamanın koşutluğunu sağlamak için sistemdeki ajanlar uygulamanın türüne göre işlemcilere atanır.

9. HLA BENZETİMLERİNİN BAŞARIMI

Tez metni içerisinde koşut işlem gereksinimi olan bilimsel benzetim uygulamalarının HLA ile olan uygunlukları incelenmiş idi. Bu uygulamaların sadece HLA'ya uygun olmaları değil, HLA ile geçekleştirildiklerinde başarımlarının nasıl olduğu da önemlidir.

HLA'ya uygun olarak geliştirilen benzetimlerin başarımını etkileyen faktörler aşağıdaki gibi sıralanabilir:

HLA Standardı: HLA standardı HLA kuralları, nesne model şablonu (OMT) ve arayüz belirtimi olmak üzere 3 ana kısımdan oluşur. HLA kuralları federasyonun işletimi esnasında federeler arasında düzgün bir etkileşimin olması için uyulması gereken ilke ve uzlaşımlardır. OMT her bir FOM'un hangi yapıda olması gerektiğini belirleyen şablondur. Arayüz belirtimi ise RTI ile federeler arasındaki 6 adet hizmetin (federasyon yönetimi, tanımlama yönetimi, nesne yönetimi, sahiplik yönetimi, zaman yönetimi ve veri dağıtımı yönetimi) yerine getirilmesini düzenleyen belirtimdir. Kısacası HLA standardı HLA'ya uygun benzetim yapılmak istendiğinde karşılanması zorunlu gereksinimleri tanımlar. RTI'nın yukarıda sayılan 6 adet hizmeti sağlaması zorunlu, ancak sağlanan hizmetlerin kullanılması zorunlu değildir.

RTI Gerçekleştirimi: HLA, RTI geliştiren yazılımcıların, geliştirme ortamı ve kullanılan algoritmaları kendilerinin belirlemesine olanak sağlayan bir esnekliktedir. Bu durumda farklı ortamlarda geliştirilen ya da aynı ortamda farklı algoritmalar koşan RTI'lar birbirlerinden farklı başarım sergileyebilir. Buna bağlı olarak, federe geliştiricilerin hangi RTI'yı seçenekleri benzetimin genel başarımını etkileyen bir faktör olacaktır.

Bölümümüzde yürütülmekte olan doktora tezi kapsamında yüksek başarım ve ölçeklenebilirlik gerektiren benzetimlerin işletilmesini sağlayabilecek, dağıtılmış bellekli koşut bilgisayarlar üzerinde çalışacak, HLA tabanlı koşut bir RTI yazılımının tasarlanması ve kısıtlı gerçekleştirimi yapılmaktadır. Bu tezde, her türlü koşut mimaride çalışacak bir RTI yerine, belirli koşut mimarilerde çalışabilecek bir yazılım hedeflenmiştir. Hedef koşut mimari 3 boyutlu Torus (3-D Torus) topolojisi olarak seçilmiştir. Bu topolojinin seçilme nedeni, var olan tera/peta flop düzeyinde

işlem yapabilen (IBM Blue Gene/L, Cray XT3, Cray XT4 gibi) süper bilgisayarların bu topolojiyi desteklemeleridir. Yazılım ise MPI ile gerçekleştirilmektedir. Böylece hem uygulamanın taşınabilirliğini sağlamayı, hem de seçilen hedef platformlara özgü MPI iyileştirmelerinden faydalanarak başarımı arttırma hedeflenmiştir.

Federasyonun RTI'dan talep ettiği hizmetler: Federasyonlar RTI'ların sağladığı tüm hizmetlerden yararlanmak zorunda değildir. Faydalanılacak hizmet sayısının azlığı, RTI tasarımcısının bu hizmetler üzerinde yoğunlaşmasını sağlayarak daha yüksek başarım seviyesinde bir RTI geliştirmesinin önünü açacaktır. Bu başlık altında değerlendirilecek bir diğer nokta ise bu hizmetleri talep eden federe sayısı, bir başka deyişle federasyonun büyüklüğü ve haberleşmede abone olunacak nesne sayısıdır.

Federelerin üzerinde çalıştığı fiziksel kaynakların gücü: Federelerin üzerlerinde çalışacakları bilgisayarların işlemci hızı, bellek sığaları ve eğer benzetim dağıtılmış ortamda gerçekleştirilecekse bilgisayarlar arasındaki haberleşme kanallarının bant genişliği ve veri taşınma hızı da doğal olarak benzetimin başarımını etkileyen faktörler arasında sayılmalıdır. Özetle, FOM'un nasıl tasarlandığı, RTI'nın hangi platformda hangi algoritmaları koşacağı, federelere sorumlulukların nasıl dağıtılacağı ve federeler arası ağın nasıl biçimlendirileceği gibi hususlar benzetimin başarımını önemli ölçüde etkileyen faktörlerdir. Tasarlanacak olan federasyonda bu etkenlerden hangisinin başarımda daha belirleyici olacağı değerlendirilmeli ve tasarım sürecinde o faktöre ağırlık vermek benzetimde yüksek başarım elde etmede faydalı olacaktır. Ayrıca birbirinden farklı özellikteki FOM, RTI, görev dağılımı uygulamalarının denenerek en yüksek başarımın sağlandığı sistem de seçilebilir.

Benzetim geliştirilirken kullanılan yardımcı araçlar: HLA uyumlu benzetim sistemleri geliştirilirken kullanılan yardımcı araçlar şüphesiz kullanımı etkinleştirmektedir. Ancak bu araçların eksiklikleri bulunmaktadır. Mevcut modelleme araçlarının tespit edilen üç önemli eksiği, yeterli görsel modelleme desteklerinin olmaması, davranış modellemesine olanak sağlamamaları ve bütünleşik bir platformun bir parçası olmamalarıdır. Bütünleşik bir platformun parçası olarak gerçekleştirilmiş modelleme araçlarının bir kısmı ise çalıştıkları platformun lisansıyla kullanılabildiklerinden pahalıya mal olmaktadırlar. Mevcut

araçların eksikliklerini taşımayan, açık kaynak kodlu ve tamamıyla ücretsiz olarak dağıtılan bir platform olan Eclipse platformu üzerinde, bu eksiklikleri giderici özgün bir HLA araç setinin tasarımı yapılmıştır [20].

10. SONUÇ

Bu tez çalışması, günümüzde yaygın olarak kullanılan ve koşut işlem gereksinimi olan bilimsel benzetim uygulamalarının HLA ile olan uygunluklarını konu alır.

HLA savunma amaçlı sistemlerin benzetiminde kullanılmak üzere geliştirilmiş ve standartlaşmış (IEEE 1516) bir mimaridir. HLA'nın savunma amaçlı uygulamalarındaki başarısı bu mimarinin sivil uygulamalarda da kullanımının yaygınlaşmasına neden olmuştur. Ancak koşut işlem gereksinimi olan bilimsel benzetim uygulamalarında henüz tam anlamıyla yerini almamıştır.

Bu tez kapsamında, koşut işlem gereksinimi olan bilimsel benzetim uygulamalarından durağan bir etkileşim sergileyen Hava-İklim Modelleme, farklı bir etkileşim içermesi bakımından (dinamik) Moleküler Dinamik ve bu iki uygulamadan tamamen bağımsız olan Akıllı Ajan uygulamaları örnek olarak ele alınmış ve bu uygulamaların HLA ile gerçeklenebilirliği incelenmiştir. Bu kapsamda her bir uygulama için HLA kurallarına uygun SOM ve FOM tasarlanıp, nesneler arasındaki komşuluk ve abone olma / yayınlama mekanizmalarının nasıl belirleneceği belirtilmiştir. RTI'nın sağladığı zaman yönetimi servisinin nasıl sağlanacağı verilen sözde kodlar içerisinde anlatılmıştır.

Yukarıda bahsedilen üç örnek uygulamanın, koşut ortamda gerçeklenmiş benzetim modelleri bulunmuş ve incelenmiştir. Bu örneklerin koşut olarak modellenmiş uygulamalarıyla HLA ile tasarlanan yapıları başarım analizi, ölçeklenebilirlik, kolay gerçekleştirilebilirlik, yeniden kullanılabilirlik ve birlikte çalışabilirlik kıstasları açısından karşılaştırılmıştır.

Molekül dinamiği ve hava tahmin modeli uygulamalarında kullanılan alan ayrıştırma yöntemi HLA'da kullanılan ızgara tabanlı veri dağıtım yönetimiyle büyük ölçüde benzerlik göstermektedir. Veri dağıtım yönetimi yapısının büyük ölçüde benzerlik göstermesi ve bu iki uygulamanın daha önce koşut ortamda gerçeklenmiş olması, bu uygulamaların dağıtılmış mimariyi destekleyen HLA ile de gerçeklenebileceğinin önemli bir göstergesidir.

Akıllı ajan uygulamalarında ise, bir sistemi oluşturan farklı işlevlere sahip parçaların her biri bir ajan olarak kabul edildiğinde bu ajanların aynı ortamda bir arada çalışabilmeleri gerektiği gibi aynı işleve sahip ajanlar farklı ortamlara

yeniden tasarlanmaksızın adapte olabilmelidirler. Bahsedilen bu iki gereksinim HLA'nın sağladığı en önemli iki özellik olan birlikte çalışabilirlik ve yeniden kullanılabilirlik ilkelerine son derece uygundur. Dolayısıyla akıllı ajan uygulamaları benzetimi için HLA'yı seçmek doğru bir tercih olacaktır. Nitekim yapılan inceleme ve araştırmalarda HLA ile gerçeklenmiş akıllı ajan uygulamalarına rastlanmıştır. Bu uygulamalardan havaalanı uygulamasının HLA ile gerçeklenme detayları tez içerisinde incelenmiştir. Gerek akıllı ajan uygulamalarının HLA'nın en önemli iki özelliğine gereksinim duyuyor olması gerekse daha önceden HLA ile gerçeklenmiş olması bu uygulamanın HLA ile gerçeklenebilir olduğunun somut bir göstergesidir.

Sonuç olarak, bütün bunlar göz önüne alındığında örnek olarak incelenen üç uygulamanın da HLA ile modellenebileceği tespit edildiği gibi gerçekleştirilen modellemelerin de özellikle yeniden kullanılabilirlik ve birlikte çalışabilirlik kıstasları açısından daha avantajlı olacağı sonucuna varılmıştır. Bunun yanında HLA, koddan bağımsız nesne modelleme imkanı sağladığı için yazılan kodun her defasında yeniden düzenlenmesi ve başkası tarafından anlaşılma zorluğu da ortadan kalkmaktadır.

Elde edilen bilgiler ışığında, gelecek çalışma olarak bu tez kapsamında incelenen bilimsel benzetim uygulamaları HLA ile gerçekleştirilebilir.

KAYNAKLAR DİZİNİ

[1] Defense Modeling and Simulation Office (DMSO),1998, High Level Architecture Interface Specification, Version 1.3

[2] Kuhl, F. Weatherly R., Dahmann J.,1999, Creating Computer Simulation Sytems-An Introduction to the High Level Architecture, Prentice Hall, Chapter 4, pp. 39-82

[3] Defense Modeling and Simulation Office (DMSO), 1998, HLA ObjectModelTemplate, Version 1.3

[4] İnternet: Meteoroloji Genel Müdürlüğü, 2008, http://www.meteo.gov.tr

[5] Aslantürk O., 2007, Hacettepe Üniversitesi Bilgisayar Mühendisliği Bölümü, Doktora Tez İzleme Komitesi Raporu, pp. 38-48

[6] Ünsal E., 2007, USMOS, Dağıtılmış Benzetim Uygulamalarında Yeniden Kullanılabilirlik, Veri Dağıtımı ve Zaman Yönetimi, ODTU, pp. 370- 379

[7] Venables D.S., Schmutternmaer C.A., 2000, Structure and dynamics of nonaqueous mixture of dipolar liquids. I. Molecular dynamics simulations. J. Chem Phys., Vol 113, No 8, pp. 3249-3260

[8] Çelik T., 2005, Yüksek Düzeyli Mimari İçin Bir Modelleme Aracı, Yüksek Lisans Tez Çalışması, Hacettepe Üniversitesi, Fen Bilimleri Enstitüsü

[9] Foster I., 1995, Designing and Building Parallel Programs, Chapter 2.6 Case Study: Atmosphere Model, Addison-Wesley

[10] Reed D.A., Adams L.M., Patrick M.L., 1987, Stencils and Problem Partitionings: Their Influence on the Performance of Multiple Processor Systems, IEEE, Vol 36, No 7, pp. 845-858

[11] Rabenseifner R., 2006, Parallelization of Explicit and Implicit Solvers, Parallel Programming Workshop-Online

[12] Baillie C.F., Broker O., McBryan O.A., Wilson J.P., 1995, MPI-RGL: a Regular Grid Library for MPI, Version 0.1, Department of Computer Science, University of Colorado, pp. 1-30

[13] Bowers K.J., Chow E., Xu H., Dror R.O., Eastwood M.P., Gregersen F.D., Klepeis B.A., Kolossvary J.L., Moraes I., Sacerdoti M.A., Salmon J.K., Shan Y., Shaw D.E., 2006, Scalable Algorithms for Molecular Dynamics Simulations on Commodity Clusters, IEEE, pp. 1-13

[14] Rapaport, D.C., 1997, The Art of Molecular Dynamics Simulation, Second Edition, Cambridge University Pres, pp. 1-5

[15] Lees, M., Logan, B., 2002, Simulating Agent-Based Systems with HLA: The Case of SIM_AGENT-Part II, pp. 1-9

[16] Gentile, A., Cossentino,M., Vitabile, S., Chella, A., Sorbello, F.,2002, Intelligent Agent Mapping on a Massively Parallel MIMD Computing Platform, pp. 1-3

[17] İnternet: Foundation for Intelligent Physical Agents, 2001,FIPA Specification documents, http://www.fipa.org

[18] İnternet: IBM, 27.11.2006, http://www.ibm.com/news/tr/tr/2006/11/sb.html

[19] Michalakes, J., Canfield, T., Nanjundiah, R., Hammond, S., 1994, Parallel Implementation, Validation, and Performance of MM5, Mathematics and Computer Science Division, Argonne National Laboratory, pp. 1-7

[20] Sözen, A., 2007, Hız Kesicilerin Trafik Yüklemesi Altındaki Dinamik Simulasyonu, Yüksek Lisans Tezi, Süleyman Demirel Üniversitesi, Fen Bilimleri Enstitüsü, İnşaat Mühendisliği Bölümü, Isparta, pp. 20-25

[21] Refson K., 2001, Moldy User's Manual, Department of Earth Sciences, Oxford, pp. 44–62

[22] Rahimi, S., Narayaman, A., Sabharwal, M., 2003, A Novel Agent Architecture for Parallel Processing, IEEE, Department of Computer Science, Southern Illinois University, pp. 1-4

[23] Andersson, J., Löf, S., 1999, HLA as Conceptual Basis for a Multi-Agent Environment, Proceedings of 8th Conference on Computer Generated Forces and Behavioral Representation, pp. 1-6

[24] Lees, M., Logan, B., Theodoropulos, G., 2007, Distributed Simulation of Agent-Based Systems with HLA, ACM Transactions on Modeling and Computer Simulation, Vol 17, No 3, Article 11

[25] Murty R., Okunbor D., 1999, Efficient Paralel Algorithms for molecular Dynamics Simulations, Paralel Computing, Vol. 25, pp. 217–230

[26] Ilnytskyi J., Wlson M. R., 2001, A domain decomposition molecular dynamics program fort he simulation of flexible molecules with an arbitrary topology of Lennard-Jones and/or Gay-Berne sites, Computer Physics Communications, Vol. 134, pp. 23–32

[27] Plimpton S., 1995, Fast Paralel Algorithms for Short-Range Molecular Dynamics, Journal of Computational Physics, Vol. 117, pp. 1–19

[28] IBM International Technical Support Organization, 2005, Unfolding the IBM @ server Blue Gene Solution, Redbooks

[29] Xue M. , Droegemeier K. K. , Weber D. , 2007, Numerical Prediction of High Impact Local Weather : A Driver for Petascale Computing, Ch. 18, pp. 1-17

[30] Michalakes J. , Chen S. , Dudhia J. , Hart L. , Klemp J. , Middlecoff J. , 2001, Development of a Next Generation Regional Weather Research and

Forecast Model, in Developments of Teracomputing, World Scientific, River Edge, New Jersey, pp. 269-276

[31] Michalakes J. , Dudhia J., Gill D., Klemp J., Skamarock W., 1999, Design of a Next Generation Regional Weather Research and Forecast Model, Towards Teracomputing, World Sicentific, River Edge, New Jersey, pp. 209-225

[32] Michalakes J., Dudhia J., Gill D., Henderson T., Klemp J., Skamarock W., Wang W., 2004, The Weather Research and Forecast Model : Software Architecture and Performance, Proceedings 11th ECMWF Workshop on the Use of High Performance Computing in Metorology, pp. 1-13

[33] Michalakes J., McAtee M., Wegiele J., 2005, Software Infrastructure fort he Weather Research and Forecast Model, pp. 1-13

[34] FIPA Agent Management Specification (Refinements). FIPAspecification documents (03-10-01). http://www.fipa.org/specs/fipa00023/XC00023H.html

[35] Adelantado M., 2001, Airport Simulation using the High Level Architecture, European Simulation Interoperability Workshop

[36] Kitano H., 1997, RoboCup as a Research Program, Sony Computer Science Laboratory, IEEE, pp. 1-2

ÖZGEÇMİŞ

Adı Soyadı : Tülin ERÇELEBİ

Doğum Yeri : Denizli

Doğum Yılı : 1981

Medeni Hali : Bekar

Eğitim ve Akademik Durumu:

Lise : 1996-2000 Denizli Süper Lisesi

Lisans : 2000-2005 Çankaya Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği Bölümü

Yabancı Dil : İngilizce

İş Tecrübesi:

2006 Eylül-Aralık Araştırma Görevlisi, Başkent Üniversitesi Bilgisayar Mühendisliği Bölümü

2006 Aralık-..... Araştırma Görevlisi, Hacettepe Üniversitesi Bilgisayar Mühendisliği Bölümü