

T.C.
GEBZE YÜKSEK TEKNOLOJİ ENSTİTÜSÜ
MÜHENDİSLİK VE FEN BİLİMLERİ
ENSTİTÜSÜ

DAĞITIK BENZETİM ALTYAPILARI İÇİN
PERFORMANS ANALİZ SİSTEMİ

RAMAZAN CENGİZ
YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ
ANABİLİM DALI

GEBZE

2007

T.C.
GEBZE YÜKSEK TEKNOLOJİ ENSTİTÜSÜ
MÜHENDİSLİK VE FEN BİLİMLERİ
ENSTİTÜSÜ

DAĞITIK BENZETİM ALTYAPILARI İÇİN
PERFORMANS ANALİZ SİSTEMİ

RAMAZAN CENGİZ
YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ
ANABİLİM DALI

TEZ DANIŞMANI
YRD. DOÇ. DR. F. ERDOĞAN SEVİLGİN

GEBZE

2007



GEBZE YÜKSEK TEKNOLOJİ
ENSTİTÜSÜ

YÜKSEK LİSANS/DOKTORA JÜRİ ONAY FORMU

G.Y.T.E. Mühendislik ve Fen Bilimleri/Sosyal Bilimler Enstitüsü Yönetim Kurulu'nun tarih ve/..... sayılı kararıyla oluşturulan jüri tarafından/...../..... tarihinde tez savunma sınavı yapılan'ın tez çalışması Anabilim Dalında YÜKSEK LİSANS tezi olarak kabul edilmiştir.

JÜRİ

ÜYE

(TEZ DANIŞMANI) : Yrd. Doç. Dr. F. Erdoğan SEVİLGİN

ÜYE :

ÜYE :

ÜYE :

ÜYE :

ONAY

G.Y.T.E. Mühendislik ve Fen Bilimleri/Sosyal Bilimler Enstitüsü Yönetim Kurulu'nun tarih ve/..... sayılı kararı.

İMZA/MÜHÜR

ÖZET

TEZİN BAŞLIĞI : DAĞITIK BENZETİM ALTYAPILARI İÇİN

PERFORMANS ANALİZ SİSTEMİ

YAZAR ADI : Ramazan CENGİZ

Sistem performansının izlenmesi ve analizlerinin yapılabilmesi, dağıtık benzetim sistemleri için çok önemli ve gerçekleştirilmesi zor bir işlemdir. Bu tür sistemlerde performans kayıplarına yol açabilecek ağ sorunları, kullanılan altyapı yada sistem kaynaklarından kaynaklanan yavaşlıklar gibi unsurların farklılığı sistem analizlerinin karmaşıklaşmasına ve performans takibinin çok zor yapılmasına sebep olmaktadır. Performans ölçümü için ortaya koyulacak kıstaslar ve oluşturulacak test ortamları dağıtık benzetim sistemlerinin gelişimini iyi yönde etkileyecektir. Bu ortamlar sayesinde üzerinde çalışılan altyapının performans sınırları ve kısıtları belirlenebilecek, yeni geliştirilen altyapıların performans kontrolleri yapılarak daha etkin ürünler elde edilmesi sağlanacaktır. Ayrıca, kullanılacak modelleme benzetim alanına göre uygun performanslı yazılımların seçilmesi için karşılaştırmalar yapılabilecektir.

Bu çalışmada, dağıtık benzetim altyapılarının performans analizini sağlayacak bir sistem tasarlanmış ve gerçekleştirilmiştir. Performans analizi için kullanılabilecek metrikler, dağıtık benzetim altyapıları incelenerek ve performans takibinde temel alınabilecek ortak noktalar temel alınarak, belirlenmiştir. Performans analiz sistemi, değişik dağıtık benzetim altyapı yazılımlarının, aynı test ortam ve koşullarında denenmelerini sağlayarak, karşılaştırmalı performans analizlerini yapmak üzere tasarlanmıştır. Analiz sistemimizi test edebilmek ve her türlü benzetim altyapısına kolayca uyum sağlayabileceğini gösterebilmek için, incelenen altyapıların ortak özelliklerini destekleyen yeni bir benzetim altyapısı geliştirilmiştir. Bu altyapının iki HLA (High Level Architecture) tabanlı altyapılı yazılımı olan DMSO RTI 1.3v6, pRTI1516 v2.1 ile belirlenen metrikler çerçevesinde karşılaştırmalı performans analizleri yapılmıştır.

SUMMARY

**THESIS TITLE : ANALYSIS SYSTEM FOR DISTRIBUTED
SIMULATION INFRASTRUCTURES**

AUTHOR : Ramazan CENGİZ

Analyzing and monitoring performance of a distributed system is a very important and hard to achieve. There are various elements such as network problems, middle-ware and system based latency problems that can hinder performance of this kind of systems. For that reason, performance analyses of these systems are quite complicated and monitoring the performance is very difficult. Therefore, establishing performance criteria and realizing performance test-beds may have positive effect on distributed simulation projects. Performance analysis systems incorporating such components can both be used to determine the limits and lacks of the distributed systems on hand and lead to efficiency improvement in middle-wares during their production phase. Moreover, they can facilitate comparisons between similar middle-ware products which help in selecting a middleware depending on the modeling and simulation area.

In this thesis, a performance analysis system distributed simulation middle-wares is designed and implemented. Metrics for performance analysis are determined by researching various distributed simulation systems and based on the common aspects they use for performance monitoring. The performance analysis system is designed for testing different distributed simulation middle-wares in the same environment and conditions and making performance analysis by comparing each system with others. To test our analysis system and to show its capability of being easily adapted to different middle-wares, a new simulation middleware having some common features of existing middle-wares is implemented. Performance analysis of this middleware is performed in comparison with two HLA (High Level Architecture) compliant software, DMSO RTI 1.3v6 and pRTI1516v2.1, based on predefined metrics.

TEŞEKKÜR

Yrd. Doç Dr. Fatih Erdoğan SEVİLGEN'e teknik ve akademik birikimiyle yaptığı değerli katkılardan dolayı teşekkür ederim.

Yoğun iş temposunun yanında yürüttüğüm yüksek lisans çalışmamın en önemli aşamalarında beni destekleyen ve bana cesaret veren değerli eşim Serpil CENGİZ'e hiç bitmeyen anlayışı için teşekkür ederim.

İÇİNDEKİLER DİZİNİ

	<u>Sayfa</u>
ÖZET	iv
SUMMARY	v
TEŞEKKÜR	vi
İÇİNDEKİLER DİZİNİ	vii
SAYFA	vii
SİMGELER VE KISALTMALAR DİZİNİ	ix
ŞEKİLLER DİZİNİ	x
ÇİZELGELER DİZİNİ	xii
1 GİRİŞ	1
1.1 Dağıtık Benzetim Sistemleri	5
1.2 HLA Standardı	6
1.3 Değişik HLA Altyapı Yazılımları	7
1.3.1 PRTI 1516	8
1.3.2 RTI Next Generation 1.3v6	8
1.4 Dağıtık Benzetim Altyapıları için Ortak Özellikler	8
1.4.1 Ortak Dil Tanımlanması	9
1.4.2 Sistemler Arası Bağlantı İşlemleri	10
1.4.3 Olayların Yönetilmesi	10
1.4.4 Zaman Yönetimi	10
1.5 Dağıtık Benzetim Altyapılarında Performans	11
1.6 Yapılmış Performans Analiz Çalışmaları	12
1.6.1 Gerçek zamanlı RTI, DIS karşılaştırması	12
1.6.2 Gerçek zamanlı DMSO RTI, PRTI karşılaştırması	13

1.6.3	RTI 1.3NGv3.2, RTI 1.3v7, MAK RTI Karşılaştırması	13
2	DAĞITIK SİSTEM TEKNOLOJİLERİ	15
2.1	Java RMI	15
2.2	DCOM	15
2.3	CORBA	16
2.3.1	IDL	16
2.3.2	ORB ve CORBA Servisleri	17
3	PERFORMANS ANALİZİ	21
3.1	Performans Metrikleri	21
3.2	Performans Analizi İçin Geliştirilen Sistemler	24
3.2.1	RAM Dağıtık Benzetim Altyapısı	25
3.2.2	Altyapı Çevirici	33
3.2.3	Altyapı Adaptörleri	35
3.2.4	Örnek Hareketli Molekül Benzetimi	36
3.2.5	Kayıt Sistemi	42
3.2.6	Analiz Aracı	47
4	DENEYSEL ÇALIŞMALAR	52
4.1	Deney1 - 2 Altyapının Mesaj Sayısı X Gecikme Analizi	52
4.2	Deney2 - Altyapının Federe Sayısı X Gecikme Analizi	55
4.3	Deney3 - Altyapının Mesaj Boyutu X Gecikme Analizi	56
5	SONUÇ VE ÖNERİLER	60
	KAYNAKLAR DİZİNİ	62
	ÖZGEÇMİŞ	64
EK 1	ALTYAPI ÇEVİRİCİ API AÇIKLAMALARI	65
EK 2	GELİŞTİRİLEN ÖRNEK ALTYAPI ADAPTÖRÜ	68
EK 3	CORBA KÜTÜPHANE API AÇIKLAMALARI	70

SİMGELER VE KISALTMALAR DİZİNİ

<u>Simgeler</u>	<u>Açıklama</u>
DoD	Amerika Savunma Ofisi (Defense of Directory)
DMSO	Amerika Savunma Modelleme ve Benzetim Ofisi
DIS	Dağıtık İnteraktif Benzetim (Distributed Interactive Simulation)
HLA	Yüksek Seviyeli Mimari (High Level Architecture)
RTI	Koşum Zaman Altyapısı (Run-time Infrastructure)
DBA	Dağıtık Benzetim Altyapısı
IEEE	Institute of Electrical and Electronics Engineers
OMG	Object Management Group
API	Uygulama Program Arayüzü (Application Programming Interface)
CORBA	Common Object Request Broker
GPS	Coğrafi Konumlandırma Sistemi (Geographic Positioning System)
SNTP	Ağ Zaman Protokolü (Simple Network Time Protocol)
PRTI	Pitch RTI
JVM	Java Virtual Machine
DCOM	Dağıtık Bileşen Nesne Modeli (Distributed Component Object Model)
COM	Bileşen Nesne Modeli (Component Object Model)
RMI	Remote Method Invocation
IDL	Arayüz Tanımlama Dili (Interface Definition Language)
ORB	Object Request Broker
DNS	Domain Name Service
UTO	Universal Time Objects
ACE	Adaptive Communication Environment
TAO	the ACE ORB

ŞEKİLLER DİZİNİ

<u>Şekil</u>	<u>Sayfa</u>
1.1 Dağıtık Sistem Ortak Paydalar Genel Görünüş.	9
2.1 IDL Örneği.	16
2.2 IDL Dönüştürme İşlemi.	17
2.3 İtme Sitili Olay Akışı.	19
2.4 Çekme Sitili Olay Akışı.	19
3.1 Benzetim Altyapıları Performans Metrikleri.	22
3.2 Mesaj Kayıt Sistemi.	24
3.3 Performans Analiz Sistemi Genel Şema.	26
3.4 RAM DBA Genel Bağlantı Şeması.	27
3.5 RAM DBA Bağlantı Koordinatörü ve Bağlantı Kütüphanesi Şeması.	27
3.6 Altyapıda Kanal Kullanımı.	28
3.7 Altyapı Zaman Yönetimi Sınıf Şeması.	29
3.8 Veri Yönetimi Sınıf Şeması.	31
3.9 RAM DBA Mesaj Yapısı.	32
3.10 RAM Altyapı Konfigürasyon Dosyası.	33
3.11 Dağıtık Altyapı Arayüzü.	34
3.12 Altyapı Çevirici API.	35
3.13 Moleküllere Ait Aktivite Şeması.	37
3.14 Molekül Benzetimi Hareketli Bileşen Yapısı.	39
3.15 Molekül Benzetimi Sınıf Şeması.	40
3.16 Molekül Benzetimi Arayüzü.	41
3.17 CORBA Kütüphanesi.	44
3.18 Kayıt Merkezi Sistem Yapısı.	45

3.19	Kayıt Sunucusu Başlatma ve Durdurma.	46
3.20	Analiz Sistemi Yapısı.	48
3.21	Analiz Sistemi Dosya Yükleme.	49
3.22	Analiz Sisteminde Listelen Federasyon Bilgileri.	50
3.23	Federasyona Ait Çizdirilebilecek Grafikler.	51
3.24	Federasyon Gecikme-Benzetim Zamanı Grafiği.	51
4.1	Deneylerde Kullanılan Bileşen Özellikleri Tablosu	53
4.2	DMSO RTI 1.3, RAM DBA Karşılaştırma Tablosu	53
4.3	DMSO RTI 1.3, RAM DBA Gecikme Karşılaştırma Grafikleri	54
4.4	Deney2 DMSO RTI Koşum Raporu Tablosu	55
4.5	Deney2 Federe Sayısına Bağlı Gecikme Grafikleri	56
4.6	Deney3 DMSO RTI Koşum Sonuçları 1	58
4.7	Deney3 DMSO RTI Koşum Sonuçları 2	59
5.1	DMSO RTI Adaptör Yapısı	68
5.2	DMSO RTI 1.3v6 Adaptör Kullanım Şeması	69

ÇİZELGELER DİZİNİ

<u>Çizelge</u>	<u>Sayfa</u>
3.1 Benzetim Altyapısı Arayüzleri.	21
3.2 Analiz için Tutulacak Kayıt Tipleri.	23

1 GİRİŞ

Son yıllarda, benzetim sistemleri kullanıcı müdahalesiz çalışan sistemler olmaktan çıkmış, dağıtık, kullanıcı ile etkileşimli çalışabilen ve genişleyebilir yapılar haline dönüşmüşlerdir. Bu dönüşümde en önemli rolü büyük çaplı ve karmaşık deneylere ihtiyaç duyulan savunma sanayi, nükleer ve genetik alanlarında bu ihtiyacı gidermek için yapılan araştırmalar üstlenmiştir. Karmaşık sistemlerin bir bütün halinde denenmelerinin çok zor olmasından dolayı bu tür sistemlerin araştırma ve geliştirme safhalarında deney gereksinimleri benzetim sistemleriyle karşılanabilmektedir. Benzetim sistemlerinin kullanımı hem maliyet olarak hem deneme süresi olarak büyük kazımlar sağlamaktadır. Özellikle, savunma alanında, benzetim sistemleri, savaş sistemlerinin ve taktiklerinin eğitimlerini daha geniş kapsamlı ve birlikte çalışabilecek şekilde yapabilmek için kullanılmışlardır. Öyle ki, bu tarz eğitim ve taktik amaçlı benzetim sistemleri genişleyebilir yapıda olmaları, birbiriyle entegre olarak çalışabilmeleri açısından savunma sektöründe aynı amaç için yapılan tatbikatlardan daha ucuza mal olmaları yönünden çok büyük bir katma değer katmıştır.

Dağıtık benzetim sistemlerinin en önemli özelliği sistemlerin modüler yapıda tasarlanmaları ve modüler olarak tasarlanan her bir parçanın başka bir benzetim sistemine de entegre olarak çalışabilmeleridir. Modülerlik sayesinde her bir parçacık benzer amaç doğrultusunda bir başka benzetim sisteminde tekrar kullanılabilir ve diğer sistemlerle birlikte çalışabilir. Bu nedenle geniş çapta yapılacak deney benzetimleri için büyük önem arz etmektedir. Çünkü benzer konularda yapılan araştırmalarda aynı amaca hizmet eden ortak parçacıklar muhakkak bulunacaktır. Bu ortak parçacıkların modüler olarak geliştirilmeleri, bu parçacıklar için her iki araştırma bünyesinde de geliştirme yapılma sıkıntısını ortadan kaldıracaktır. Dağıtık benzetim sistemleri konusunda yapılan araştırmalarda hedeflenen nokta geliştirilen benzetim sistemlerinin tekrar kullanılabilirliğini ve birlikte çalışabilirliğini sağlamaktır.

DoD (Defense of Directory, Amerikan Savunma Bakanlığı) savunma amaçlı dağıtık benzetim çalışmalarında DMSO (Defense Modeling and Simulation Office,

Savunma Modelleme ve Benzetim Ofisi) ile öncü atılımlar yapmıştır. Bu kapsamda ilk olarak 1990-1995 arasında DIS (Distributed Interactive Simulation, Dağıtık İnteraktif Benzetim) protokolü ortaya çıkartılmış ve dağıtık benzetim dünyasında yeni bir sayfa açılmıştır. DIS tabanlı sistemlerin kullanılmasıyla özellikle eğitim amaçlı benzetimlerde daha modüler ve birlikte çalışabilir sistemlerin geliştirilmesi sağlanmıştır [1]. DIS ile birbiriyle mesajlar aracılığıyla haberleşen, birbirinden bağımsız çalışan bileşenlerin oluşturduğu sanal bir dünya oluşturulmuştur. Sistem döngüsünde kendisine gelen mesajlar sayesinde bileşenler yeni durumlara geçmekte, bu durumlarında oluşan mesajlar gönderildikleri diğer bileşenlerde durum değişikliklerine yol açmaktadır. Bu şekilde birbiriyle etkileşim pek çok dağıtık benzetim bir arada çalışabilmektedir.

DIS birlikte çalışabilirlik sağlayan bir protokol olmakla birlikte tam anlamıyla benzetim dünyasında duyulan ihtiyaçlara cevap veremediğinden; genişleyen benzetim alanında tekrar kullanılabilir, farklı zaman yönetim mekanizmalarına sahip benzetimleri birleştirebilecek genel bir mimariye ihtiyaç duyulmuştur. Ayrıca, DIS protokolünün kullandığı yayımlama tekniğinden dolayı her yayımlanan bilginin bütün benzetimlere gönderilmesi sistem yoğunluğunu artırmakta ve özellikle gerçek zamanlı benzetimler için tasarımı zorlaştırmaktadır. Benzetim gurubuna dahil olacak küçük benzetim parçalarının bile yayımlanacak bütün veriler için protokole uygun tanımlama yapılması zorunluluğu, birlikte çalışabilirliği ve genişleyebilirliği artırmak amacıyla yeni bir sistemin geliştirilmesini gerekli kılmıştır. Bu kapsamda yapılan araştırmalar 1998 yılında HLA (High Level Architecture, Yüksek Seviyeli Mimari) standardının ortaya çıkmasını sağlamıştır [2]. HLA standardının geliştirilmesiyle fonksiyonel açıdan daha büyük ölçekli benzetim guruplarının birlikte çalıştırılabilmesini sağlamak amacıyla standardı destekleyen değişik altyapı yazılımları geliştirilmiştir.

Zaman içerisinde yeni dağıtık benzetim altyapılarının geliştirilmesinde en önemli etken altyapıların gelişen benzetimler için fonksiyonel açıdan yetersiz oluşu olmuştur. Dağıtık benzetim konusunda yapılan çalışmalar SimNET ile laboratuvar ortamında kurulu ve kendisi için kurulan özel ağda çalışan sistemle başlamış. Ağ kullanımının yaygınlaşmasına ve ağ hızına paralel olarak gerekli güncellemeler yapılarak geniş alan ağlarında da çalışabilecek DIS sistemi geliştirilmiştir. DIS

protokolü ile ortaya çıkan birlikte çalışabilirlik ve genişleyebilirlik özelliklerinin, genişleyen benzetim sistemi kullanımında ihtiyaca cevap verememesi HLA mimarisinin geliştirilmesi sonucunu vermiştir.

Dağıtık benzetim sistemlerinin kullanıma geçirilebilmesi için fonksiyonel açıdan eksiklerin giderilmesi yetmemektedir. Fonksiyonel açıdan istekleri karşılayan her altyapı beklenen performansı verememektedir. Benzetim sistemlerinin geliştirme ve kullanım aşamalarında sürekli olarak performans problemi gündeme gelmekte ve bu problem konusunda çalışmalara ihtiyaç duyulmaktadır. Sistem tasarımcıları ve program geliştiriciler, fonksiyonel açıdan birbirinin aynı olmalarına karşın üzerinde çalıştıkları sisteme en uygun performanslı altyapı yazılımını kullanmak için sistem performansını ölçebilecek bir sisteme ihtiyaç duyarlar. Yine sistem yazılımı devam ettiği sırada geliştiriciler altyapının performans eksiklerini görebilmek ve takip edebilmek için bir ortama ihtiyaç duyarlar ya da geliştirilmesi biten bir ürünün hatalarını takip edebilmek için sistem bakımıcısı benzetim sistemini ve üzerinde çalıştıkları alt yapıyı sürekli kontrol eden bir sisteme ihtiyaç duyar. Burada verilen örnekler daha da çoğaltılabilir. Dağıtık sistemlerin farklı makineler üzerinde, farklı işletim sistemleri üzerinde, farklı yoğunlukta sistemlerde çalışabilmeleri bu sistemlerin performans analizlerini karmaşıktırılmaktadır [3].

Dağıtık benzetim sistemlerinde performans analizi ihtiyacını gidermek amacıyla pek çok çalışma yapılmıştır. Bu araştırmalar altyapıların geliştirme sürecinde sürekli devam etmiş ve gelişim durumları kontrol edilmiştir. Özellikle HLA konusunda yapılmış birçok analiz çalışması vardır. Çalışmalara örnek olarak gerçek zamanlı HLA ve DIS karşılaştırması [11], gerçek zamanlı DMSO RTI, PRTI karşılaştırması [13], değişik RTI versiyonları arasında yapılan bir karşılaştırma [14] verilebilir. Bu çalışmalardan ileriki bölümlerde daha detaylı bahsedilecektir.

Bu tez çalışmasında, bir çok benzetim altyapısında kullanılabilecek bir performans inceleme yöntemi ve bu yöntemeye uygun yazılım sisteminin geliştirilmesi amaçlanmıştır. Çalışmada geliştirilecek analiz sisteminin daha önce yapılmış sistemlerden en belirgin farklılığı benzetim altyapısından bağımsız olarak çalışabilmesi ve bu sayede farklı benzetim altyapıları için ortak analiz aracı olarak kullanılabilesidir. Çalışma kapsamında HLA RTI (Run-time Infrastructure, Koşum

Zaman Altyapısı) altyapıları üzerinde yapılan incelemeler sonucunda analiz işlemlerinde kullanılacak ortak kriterler belirlenmiş ve aynı kriterlerin ölçümlendirilebileceği farklı bir benzetim altyapısı geliştirilmiştir. Ortaya çıkan performans analiz sistemi sahip olduğu altyapı dönüştürme mekanizması ile ileride geliştirilecek altyapılar için de kullanılabilir bir yapıda tasarlanmıştır. Çalışmanın bir parçası olarak geliştirilen her yazılım aracı ve kütüphanesi modüler olarak tasarlanmış ve her türlü benzer görevde kullanılabilir şekilde ortaya koyulmuştur. Çalışma içeriğinde;

- 1) Her türlü altyapının analize tabi tutulabilmesini sağlayacak bir Altyapı Dönüştürücü Yazılımı,
- 2) Denemelerde kullanılmak üzere RTI 1.3NG için Altyapı Servis Adaptörü Yazılımı,
- 3) Temel benzetim yeteneklerine sahip bir RAM DBA (Dağıtık Benzetim Altyapı) Yazılımı,
- 4) RAM DBA'sı için Altyapı Servis Adaptörü Yazılımı,
- 5) Performans kayıtlarının tutulması amacıyla Senkron Dağıtık Mesaj Kayıt Sistemi Yazılımı,
- 6) Performans Kayıtları üzerinden analiz işlemlerini yapabilmek amacıyla Benzetim Performansı Analiz Aracı Yazılımı geliştirilmiştir.

Çalışmalarda üretilen sistemlerin fonksiyonlarını test edebilmek amacıyla HLA tabanlı RTI 1.3NG altyapısı yazılımı ile RAM DBA yazılımının değişik mesaj yoğunluklarında performans incelemesi yapılmış ve iki sistemin karşılaştırmalı performans grafikleri elde edilmiştir. Bu grafikler ışığında RAM altyapısının performansının RTI dan düşük çıktığı ve özellikle mesaj yoğunluğu artışlarında performansın katlı olarak düştüğü gözlenmiştir.

Tez içeriğinde birinci bölümde genel olarak dağıtık benzetim sistemlerinden bahsedilecek, dağıtık sistemlerinin ortak yönleri ifade edilecektir, daha önce yapılmış analiz çalışmalarından örnekler verilecek, performans çalışmasında ele alınacak kriterlerden bahsedilecektir. İkinci bölümde dağıtık sistem geliştirilmesinde

kullanılan deęişik teknolojilere deęinilerek, bu teknolojilerden birisi olan CORBA hakkında temel bilgiler verilecektir. Üçüncü bölümde analiz sistemi hakkında ayrıntılı bilgiler verilecektir. Dördüncü bölümde analiz sisteminin örnek bir kullanımı hakkında yapılan deneyden bahsedilecektir. Son bölümde elde edilen sistemden ve gelecekte yapılabilecek çalışmalardan bahsedilecektir.

1.1 Dağıtık Benzetim Sistemleri

Teknolojinin hızlı gelişimi, fiziksel devrelerin hızlandırılması ve haberleşme ağlarının geliştirilmesi, veri işleme hızının geçmişe nazaran çok daha artmasına sebep olmuştur. Günümüzde veriler üzerinde yürütülen işlemler de en az verilerin boyutları kadar büyük ve karmaşıktır. Geliştirilen sistemlerin karmaşıklığı ve deęişik performans gereksinimleri yazılım sistemlerinde yeni açılımlar oluşmasını sağlamıştır. Dağıtık programlama ya da dağıtık sistemler genel olarak 1990lı yıllarda gelişmeye başlamış ve teknolojinin performans gereksinimine binaen ortaya atılmıştır. Karmaşık yazılımların, dağıtık sistemler aracılığıyla dağıtık makineler üzerinde çalıştırılabilmesi sistem performansını artırmıştır.

Teknolojinin bir alt kolu olarak benzetim sistemleri de zaman içinde gelişmiş ve çok büyük boyutlarda, analiz amaçlı çalışmalarda kritik değerler ifade eden sistemler tasarlanmıştır. Bir çok olayın gerçek zamanda testinin, analizinin maliyet olarak pahalıya mal olması benzetim sistemlerinden beklenileni artırmıştır. Bu maliyet artışı hem zaman planında hem maddi giderler planında yansımaları sebep olmuştur. Benzetim sistemlerinden beklenilenin artış göstermesiyle bu isteklere cevap verebilecek yeni yapıların geliştirilmesine çalışılmıştır. Dağıtık benzetim sistemlerinin geliştirilmesi bu isteklerin cevaplanmasında bir çözüm ifade etmektedir.

Dağıtık sistem olarak tasarım bir benzetime genişleyebilirlik, modülerlik, geri kullanılabilirlik, performans konularında değerler katar. Benzetimden beklenen bir nokta yeni eklentilerin yapılabilmesi, geliştirilen sistemin kolayca farklı amaçlar için deęiştirilebilmesidir yani modüler bir yapıda çalışmasıdır; dağıtık tasarım sayesinde sistem birbirinden bağımsız modüller olarak tasarlanıp istenilen modüllerin özellikleri deęiştirilerek sistem deęiştirilebilir ve yeni modüller eklemek suretiyle

sistemin genişleyebilmesi sağlanır. Modüler olarak tasarlanmış bu yapı farklı bir dağıtık benzetimle protokolleri uyduğu sürece birlikte çalışabilir bu da büyük yapıları benzetimlerin kurulmasına, tasarlanan dağıtık sistem parçalarının değişik amaçlarda tasarlanmış farklı benzetimler içinde kullanılabilmesine olanak sağlar.

Büyük çapta dağıtık benzetim sistemlerinin geliştirilmesi için bütünü teşkil edecek parça benzetimlerinin belirli bir formata uygun tasarlanması amacına yönelik çalışmalar 1996 yılında DMSO HLA'in ortaya çıkmasıyla boyut değiştirmiştir. HLA modüler olarak geliştirilecek benzetimler için standart bir mimari yaklaşım getirmektedir. Böylece aynı mimari yapıda geliştirilen modüllerin birlikte çalışabilirlikleri sağlanmış olacaktır. HLA'in ortaya çıkmasından sonra dağıtık benzetim çalışmaları HLA üzerine yapılacak güncellemeler ve iyileştirmeler üzerinden devam etmiştir. HLA 1.3 1998 yılında IEEE (Institute of Electrical and Electronics Engineers, Elektrik Elektronik Mühendisler Odası) kontrolünde OMG (Object Management Group, Nesne Yönetim Gurubu) tarafından bir standart olarak kabul edilmiştir. Daha sonraki yıllarda performans ve fonksiyonel açılardan yapılan güncellemelerle standart 2000 yılında HLA 1516 standardı olarak IEEE tarafından kabul edilmiştir[4].

1.2 HLA Standardı

HLA standardı temel olarak 3 ana parçadan oluşmaktadır. Bunlar HLA kuralları, arayüz tanımlamaları ve nesne model şablonlarıdır. HLA standardına göre benzetimin bütünü federasyon ve benzetim bileşenleri federe olarak isimlendirilir.

HLA Kuralları: HLA kuralları geliştirilecek federasyon ve federelerin diğer geliştirilmiş federelerle birlikte çalışabilmeleri için taşıması gereken özellikleri ifade eder. Bu kural ifadeleri standart tanımlarından incelenebilir[4].

Arayüz Tanımlamaları: Arayüz tanımlamaları geliştirilecek federelerin kullanacakları ortak API (Application Programming Interface, Uygulama Programlama Arayüzü) bilgilerini içerir. Bu arayüzü sağlayan HLA yazılımı RTI (Run-Time Infrastructure) ismiyle geliştirilmektedir. IEEE standardında tanımlı 6

yönetim başlığı vardır. Federeler bu 6 yönetim arayüzünü kullanarak benzetim döngüsüne dahil olabilirler. Bu yönetim bölümleri şu şekilde sıralanabilir:

- **Federasyon Yönetimi:** Federasyonların başlatılması, durulması, tekrar başlatılması, sonlandırılması, federelerin federasyonlara katılıp, federasyonlardan ayrılması işlemlerinin yönetilmesini sağlayan arayüz bölümüdür.
- **Bildirim Yönetimi:** Federasyon çalıştırılması sırasında yayınlanacak yada güncellemelerin alınması talebinde bulunulacak verilerin yayınlanması ve federasyon ortamına bildirimde bulunulmasını sağlayan arayüz bölümüdür.
- **Nesne Yönetimi:** Federeler tarafından güncellenen nesnelerin ve iletilerin federasyon üzerinde ilgili federelere bildirilmesini sağlayan arayüz bölümüdür.
- **Sahiplik Yönetimi:** Federasyonda tanımlı nesne özniteliklerinin sahipliklerinin federeler arasında değiştirilmesini sağlayan bölümdür.
- **Zaman Yönetimi:** Mantıksal federasyon zamanının federeler arasında koordineli bir şekilde ilerletilmesini sağlayan bölümdür.
- **Veri Dağıtım Yönetimi:** Federasyon içinde verinin daha etkili dağıtılmasını ve yönlendirilmesini sağlayan bölümdür.

Nesne Model Şablonları: Federasyonda kullanılacak nesne ve iletilerin tanımlamalarına ait bilgilerin yapıldığı Nesne Modellerinin ortak tanımlama ifadelerini oluşturan şablonlardır.

1.3 Değişik HLA Altyapı Yazılımları

Günümüzde HLA standardının bir bakıma geliştirilme nedeni olan DMSO'nun gerçek standartlarda ürün olarak kendini ispatlayan RTI yazılımları listesinde [5] sadece 43 farklı sürümlü ürün bulunmaktadır. Bu bölümde yapılan araştırmalarda kullanılan iki RTI yazılımından bahsedilecektir. Bunlar Pitch AB tarafından üretilen

pRTI1516 ve SAIC tarafından üretilen RTI 1.3v6 dır. Her iki yazılımda DMSO'nun listesinde bulunan dünyada pek çok benzetim sisteminde araştırma veya son kullanıcı boyutunda kullanılan altyapı yazılımlarıdır.

1.3.1 PRTI 1516

PRTI 1516 yazılımı, (Nisan 2003) IEEE 1516 2. sürümlü standardında belirlenen bütün servisleri sağlayan Pitch AB firması tarafından üretilmiş bir RTI yazılımıdır [17]. DMSO sayfasında yayınlanan RTI yazılımları listesinde listelenen ürünlerden 11 tanesi PRTI yazılımının değişik sürümleridir. Java tabanlı geliştirilen sistem C++ arayüzleri aracılığıyla, C++ uygulamalarında da kullanılabilir. DMSO'nun araştırma çalışmalarını yürüten SAIC firması tarafından üretilen ilk RTI yazılımlarındandır. RTI 1.3v6 SAIC firması halen RTI NG (RTI Next Generation) serisinde değişik versiyonlar çıkartmaktadır [18]. RTI NG yazılımları C++ programlama dili kullanılarak yazılmıştır. Yazılım çalışmalarında CORBA (Common Object Request Broker) ara katmanı kullanılmıştır.

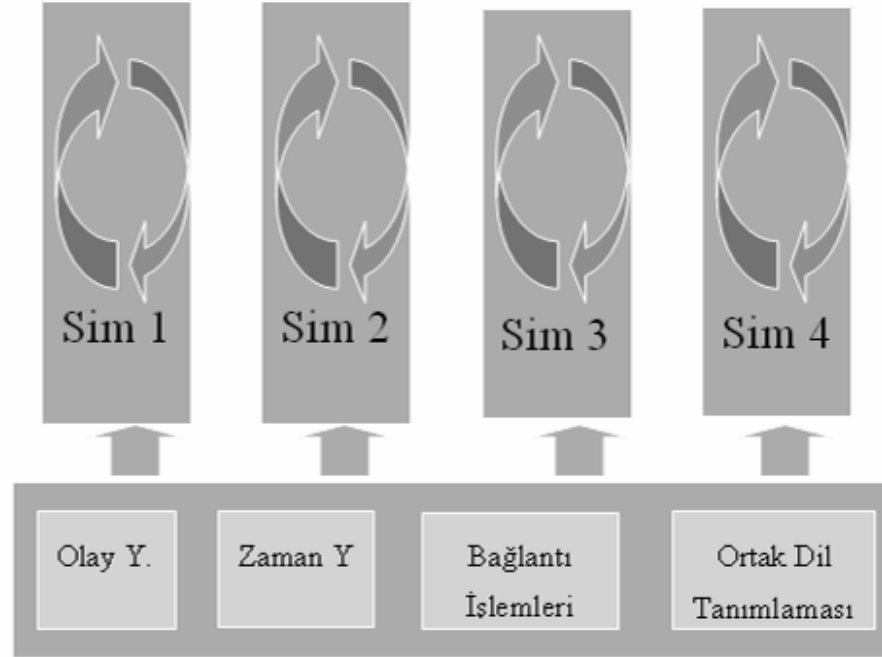
1.3.2 RTI Next Generation 1.3v6

DMSO'nun araştırma çalışmalarını yürüten SAIC firması tarafından üretilen ilk RTI yazılımlarındandır. RTI 1.3v6 SAIC firması halen RTI NG (RTI Next Generation) serisinde değişik versiyonlar çıkartmaktadır [18]. RTI NG yazılımları C++ programlama dili kullanılarak yazılmıştır. Yazılım çalışmalarında CORBA (Common Object Request Broker) ara katmanı kullanılmıştır.

1.4 Dağıtık Benzetim Altyapıları için Ortak Özellikler

Farklı benzetim sistemlerinin birlikte çalışabilirliğini sağlayabilmek için öncelikle ortak bir dil tanımlanmasına ihtiyaç var. Bu ortak dil aracılığıyla sistemler arası iletim gerçekleştirilebilir. Sistemlerin hangi bilgileri paylaşacaklarını bildirmelerini sağlayacak bir servis yönetimi gerçekleştirilmelidir. Ayrıca, iletilecek verilerin sistemlere karşılıklı uygun zamanda ve uygun değerlerle gönderilmesini sağlamak için olay ve zaman yönetim birimleri altyapının bir parçası olmalıdır. Birbirine bağlanacak olan benzetimler

Şekil 1.1'de görüldüğü gibi ilgili servislerden oluşan bir altyapı ile birbirine bağlanabilirler.



Şekil 1.1 Dağıtık Sistem Ortak Paydalar Genel Görünüş.

1.4.1 Ortak Dil Tanımlanması

Her benzetim programının kendine özgü bir tasarımı ve uygulama geliştirme dili vardır. Bu kendine özgü tasarım sonucunda oluşan sistemin başka sistemlerle çalışabilmesi için ortak bir dil tanımlanması gerekir. Öyle ki bir benzetimde tanımlanan ve kullanılan bir nesne diğer sistemlerde de tanımlanabilmesi için onların da aynı ifadelerle konuşuyor olması gerekir. Sistemin geliştirilme dili C++ yada Java gibi farklı diller olsa da sistemlerin konuşmasını sağlayacak daha üst düzeyli bir dil tanımlaması yapılmalı ve paylaşımlı nesneler ve servisler tekrar kullanılabilirlik için o dille ifade edilmelidir. Ve nesnelere erişim o dille sağlanan arayüz aracılığıyla yapılmalıdır. HLA mimarisinde bu ortak dil tanımlamasını nesne modelleri oluşturur. Federe diye tabir edilen ve bir benzetim gurubuna dahil olacak benzetimler yayınlama yapılacak nesnelerle ilişkin tanımlarını federe nesne modeli içinde belirtirler Böylece nesnelerin diğer benzetimlere yayınlanması RTI yazılımı aracılığıyla bu federe nesne modeli üzerinden yürütülür.

1.4.2 Sistemler Arası Bağlantı İşlemleri

Benzetim sistemlerinin birbiriyle haberleşebilmeleri için bazı alt seviye ağ haberleşme işlemleri yapılması gerekir. Bu alt düzey işlemlerin benzetim sistemi sırasında karmaşıklığı artırmamalı ve benzetim sistemlerinin tasarımında altyapıdan daha çok model tasarımına ağırlık verilebilen bir bağlantı sistemi olmalıdır. Bu sistem ağa veri gönderme, veri alma gibi dağıtık benzetim için temel işlev sayılabilecek işlemleri gerçekleştirecek ve benzetimi alt işlemlerden soyutlayacaktır. Bu işlev RTI tarafından otomatik olarak yapılmaktadır. RTI konfigürasyon dosyasından ağ yoğunluğu ve mesaj gönderimi için ağ adresi kısıtlamaları tanımlanarak iletim sistemi özelleştirilebilir.

1.4.3 Olayların Yönetilmesi

Benzetim döngüleri başladıktan sonra meydana gelen olayların bütün sisteme duyurulması ve o olayla ilgilenen birimlerin bundan haberdar edilmesi işlemlerini yerine getiren bir olay yönetim sistemi olmalıdır. Olay yönetim sistemi yazılım boyutunda gerçekleştirilecek olayların takibini ve yönetilmesini sağlar. Bağlantı işlemlerini sağlayan birim ise olay yöneticisinin gerçekleştirmek üzere olduğu bir olay iletiminde ağ katmanına veri gönderilmesi ve alması gibi alt işleri yürütür. Böylelikle bileşenler birbirlerine herhangi bir anda herhangi bir olay bilgisini gönderebilecek ve o olaya ilişkin değişik durumlar diğer benzetim bileşenleri tarafından algılanıp işlenebilecektir. Sistemin düzgün çalışması için olay bilgilerinin gecikmesiz ve uygun sırayla bileşenlere iletilmesi gerekir.

1.4.4 Zaman Yönetimi

Bir benzetimin düzenli bir şekilde zaman planında ilerlemesini sağlamak için bir zaman yönetimi gerekmektedir. Bu zaman yönetimiyle olayların oluşma sıraları bileşenler tarafından algılanabilecek ve meydana gelecek olaylar senkron bir şekilde yapılabilecektir. Bu yönetim sistemi farklı benzetimlerin birlikte çalışmaları durumunda benzetimler arası zaman yönetimini de yapabilmelidir. Ancak bu sayede dağıtık çalışan sistemlerden birbirine gönderilen olaylar uygun sırada ilgili bileşenlere iletebilir.

1.5 Dağıtık Benzetim Altyapılarında Performans

Benzetim sistemlerinde gelişen teknolojiler sayesinde büyük ölçekli benzetimlerin dağıtık olarak çalıştırılması sağlanabilmektedir. Benzetim sistemleri için kullanılan altyapılar da sürekli fonksiyonel olarak gelişmekte, benzetim sistemi tasarımını kolaylaştırıcı hale gelmektedir. Bu gelişim sürecinde ele alınan başlıca gelişim kriteri fonksiyonelliktir. Mevcut altyapıların istenilen sistem tasarımları için fonksiyonel olmamaları yeni altyapılar geliştirilmesini sağlamıştır. Örneğin, ilk dağıtık benzetim çalışmalarından SimNET'te benzetimlerin prototip çalışmalarda birlikte çalışması sağlanmış fakat sistemin genişleyebilirliği göz önünde bulundurulmamıştır. İlerleyen süreçte bu yeteneğe sahip altyapıların geliştirilmesi çalışmaları yürütülmüştür. Geliştirilen her altyapının amaçlandığı gibi mükemmel bir yapı olmadığı sürekli gelişmeye devam etmesinden anlaşılabılır.

Geliştirilen yeni altyapılarda fonksiyonellik yanında görülen en kritik eksiklikler performans eksiklikleridir [11]. Bu performans eksikliklerinin belirlenmesi ve gelişim sürecinde kullanılması dağıtık sistemlerde oluşan karmaşık yapıdan dolayı zordur. Bundan dolayı altyapılarda geliştirme sürecinde faydalı olabilecek ve benzetim altyapıları üzerinden analiz yapabilecek sistemlere ihtiyaç vardır [9].

Performans analizi ihtiyacı sadece yeni altyapıların geliştirme sürecinde önemli değildir. Bir altyapının geliştirme sürecin bitmesi ve bir kaç prototip için kullanılabilmesi genel anlamda yüksek performanslı çalışabileceğini göstermez. Altyapının kullanılması sırasında özellikle benzetim sisteminin tasarımı sırasında altyapının göstereceği performansa göre değişik tasarımlar içinden en uygun performanslı federasyon tasarımının seçilebilmesi için de yine altyapının performans analizinin yapılabilmesine ihtiyaç vardır [10].

Ayrıca geliştirilecek bir benzetim sistemi için ihtiyaçlara cevap verebilecek en uygun altyapının seçilebilmesi sırasında göz önünde bulundurulması gereken bir önemli faktör de altyapının göstereceği performanstır. CORBA gibi geniş kapsamlı bir dağıtık sistem altyapısının basit amaçlı yazılımlarda göstereceği performansın, Java RMI gibi karmaşık yazılımlar için basit olan bir yazılımın aynı sistemde

göstereceği performanstan yavaş olması, sistem geliştirmede kullanılacak altyapının sisteme olan katkısının doğrudan altyapı ile yapılabilecek fonksiyon sayısı ile orantılı olmadığını göstermektedir [7].

1.6 Yapılmış Performans Analiz Çalışmaları

Dağıtık altyapılar için yapılan performans analiz çalışmaları yeni altyapıların geliştirme sürecine paralel devam etmiştir. Altyapıların diğer altyapılarla karşılaştırmaları yapılarak sistemlerin gelişme durumları kontrol edilmiştir. Özellikle HLA uyumlu yazılımlar üzerinde yapılan bir çok analiz çalışması mevcuttur [15], [16], [12], [13], [14]. Bu bölümde ele alınan analiz çalışmalarından farklı sistemler üzerinde yapılan ve özellikle kayıt senkronizasyonu birbirinden farklı gerçekleştiren üçü örnek olarak ele alınacaktır.

1.6.1 Gerçek zamanlı RTI, DIS karşılaştırması

Purdy ve Wuerfel gerçek zamanlı bir uçak benzetiminde RTI ve DIS yazılımlarının kullanıldığı 2 versiyon arasında karşılaştırmalar yapmıştır [12]. Dağıtık koşum işlemleri hem RTI ile hem de DIS ile ayrı ayrı yapılmıştır. Aradaki gecikmelerin saptanabilmesi için gecikme hesaplamaları ikiye ayrılarak yapılmıştır. Gecikmeler önce hafızadan ağ kartına gönderilme sırasında ve alıcı tarafında ağ kartından alınıp hafızaya gönderilme sırasında oluşan gecikme olarak iki şekilde hesaplanmıştır. Ölçümler için her makinede GPS (Geographic Coordination System) zamanı kullanılarak, sistemin belirli noktalarında bu saat üzerinden kayıtlar tutulmuştur. Alınan kayıtlar üzerinden değişik hesaplamalarla performans karşılaştırmaları yapılmıştır.

Sistemde koşan uçak sayısı artırılarak denemeler yapılmıştır. Her makinede belirli sayılarda uçak simülatörü koşturmak üzere, sırayla 2, 3, 4, 5, 6, 7, 8, 9 federenin çalışması ve aradaki gecikmelerin saptanması işlemleri de yapılmıştır. Sistemde gönderilen paket sayısı ve boyutları da incelenen kriterlerdir. Sistemde bütün federeleri denemek için 2 makine kullanılmıştır.

Çalışma sonucunda artan federe sayısına bağlı olarak RTI yazılımı işlemlerinde aynı veri yolunu ve ağ kartını kullanmalarından dolayı 3-4 ms.'lik bir gecikme olduğu gözlemlenmiştir. Bununla birlikte DIS ile üretilen sistemde federe sayısının artmasıyla meydana gelen mesaj gönderme gecikmeleri 3-4 ms.'nin çok daha üstündedir. Bu da DIS ile gönderilen mesajların çok daha gecikmeli olarak hedefe ulaşması sonucunu çıkartır. Ağdan gelen mesajların federeye iletilmesi konusunda ise DIS çok hızlı çözüm üretmektedir. Yaklaşık 2-3 ms. içinde gelen mesajlar hedefe ulaştırılmış olurken RTI yazılımı ile ağdan gelen bir mesajın işlenip hedef federeye ulaştırılması 12 ms. civarında sürmektedir. Çalışmada geliştirilen yeni sistemin gecikme performansı olarak DIS'ten çok daha başarılı olduğu gözlemlenmiştir.

1.6.2 Gerçek zamanlı DMSO RTI, PRTI karşılaştırması

Sjöström ve arkadaşları, CelsiusTech Systems firmasında geliştirdikleri benzetim sistemlerinde kullanmak üzere performansı iyi bir altyapıyı seçmek için değişik testler uygulamışlardır[13]. Deneyler ağa bağlı 3 bilgisayar üzerinde gerçekleştirilmiştir. Deneylerde CPU yüklenmesi, gecikmeler ve ağ yoğunluğu değerleri hesaplanarak karşılaştırmalar yapılmıştır. Yapılan karşılaştırmalar için makinelerin sistem saatlerine göre kayıtlar alınarak yapılmıştır. Makinelerin sistem saatleri ise SNTP (Simple Network Time Protocol) ile senkron hale getirilmiştir. Bu sayede bütün makinelerde tutulan zaman kayıtlarının aynı türden olması sağlanmıştır. Araştırma sonucunda PRTI'nın JVM (Java Virtual Machine) kullanmasından dolayı CPU yükünün fazla olduğu, nesne güncellemelerinde oluşan gecikmelerin ise DMSO RTI'ya göre PRTI da daha az olduğu yönünde sonuçlar elde edilmiştir.

1.6.3 RTI 1.3NGv3.2, RTI 1.3v7, MAK RTI Karşılaştırması

Gerçek zamanlı 2 füze simülatörünün çalışması sırasında gönderdikleri mesajların hedefe varış gecikmelerinin altyapılara bağlı değişimini incelemek amacıyla LeSueur ve Burks bir çalışma yapmışlardır [14]. Çalışma aynı ağ ortamında SNTP sunucuya bağlı 2 makine üzerinde yapılan benzetim koşumlarına ait kayıtların değerlendirilmesiyle elde edilmiştir. Koşum sırasında füzelerin gönderdiği ve aldığı her mesaj, işlem öncesi ve sonrası ilgili zamanda kayıt bilgisi olarak tutulmaktadır.

Daha sonra bu kayıt bilgileri kullanılarak deęerlendirme grafikleri elde edilmiřtir. Bu arařtırma sonucu MAK RTI in dięer RTI yazılımlarından daha az gecikmeyle alıřtıęı sonucu elde edilmiřtir. alıřma iin tutulan kayıtların SNTP aracılıęıyla yapılması ve gecikmelerin bu saat zerinden hesaplanması analiz alıřmalarında SNTP'nin de kullanılabileceęini gstermiřtir.

2 DAĞITIK SİSTEM TEKNOLOJİLERİ

Tez kapsamında yapılacak çalışmalarda kullanılmak üzere değişik dağıtık sistem teknolojileri incelenmiştir. Özellikle analiz sistemi için geliştirilmesinde kullanılacak dağıtık benzetim altyapısı için bir teknoloji belirlenmesine ihtiyaç duyulmuştur. İncelemeler sonucunda bir dağıtık benzetim altyapısının CORBA, Java RMI (Remote Method Invocation), DCOM (Distributed Component Object Model) teknolojilerinden herhangi birisiyle geliştirilebileceği belirlenmiştir. Bu bölümde Java RMI ve DCOM bilgileri hakkında genel bilgiler verilecek sonra tez çalışmasında kullanılacak CORBA teknolojisi hakkında daha geniş bilgi verilecektir. Çalışma kapsamında dağıtık olarak hizmet verebilecek geniş bir servis setine ihtiyaç duyulduğundan CORBA teknolojisi diğerlerine göre daha kapsamlı olması dolayısıyla tercih edilmiştir.

2.1 Java RMI

Java RMI, Java içinde 1997 yılından sonra standart olarak gelen bir teknolojidir. RMI farklı bilgisayarlar üzerinde çalışan Java tabanlı sistemlerin etkileşebilmelerini, birbirine nesneler gönderebilmelerini sağlayan bir yapıdır. Fazla gelişmiş servisi yoktur; temel yapıda fonksiyonlar içerir. Benzetim sistemlerinin dağıtık çalışabilmesi için uygun bir yapı hazırlanabilir fakat zaman yönetimi servisi gibi birçok yapının temelden tanımlanması gerekir [7].

2.2 DCOM

DCOM Microsoft tabanlı bir dağıtık sistem yöneticisidir. Yerel COM (Component Object Model) nesnelerinin farklı bilgisayarlar üzerinden Windows ortamında çalıştırılabilmesini sağlamaktadır. Benzetim sistemleri çalışmalarında kullanılmıştır [6]. Sistem, Windows tabanlı olduğundan ancak Windows tabanlı geliştirilecek benzetim sistemlerinde kullanılabilir.

2.3 CORBA

CORBA, OMG gurubu tarafından 1990 yılında geliştirilmiş olan bir yapıdır. İlk çıkışından itibaren üzerine bir çok değişik servisin eklenmesiyle çok büyük bir ara katman haline gelmiştir. Her dilde yazılıma olanak sağlayan CORBA, dağıtık sistemi ağla alakalı bütün işlemlerden arındırır. Sistemin büyük olması kontrolünü zorlaştırmaktadır. Sağladığı servislerle dağıtık benzetim sistemlerinde kullanılabilecek bir yapıdır [7].

CORBA ortamında dağıtık benzetim tasarlanmasını sağlayacak yapılar IDL (Interface Definition Language), ORB (Object Request Broker) ve CORBA Servisleri olarak sınıflandırılabilir. CORBA servisleri içinden, istenilen dağıtık benzetim sistemi için, Olay, İsimlendirme ve Zaman Servisleri yeterli olarak görülmüştür. Ama istenildiği takdirde kullanılan ORB sağlayıcının desteklediği faydalı servisler de sisteme dahil edilebilir.

2.3.1 IDL

IDL, CORBA ortamında paylaşımlı bileşen ve servislerin tanımlamasının yapıldığı bir dildir. Şekil 2.1'de görüldüğü gibi dilin C++ yazımına benzer kendine özgü bir söz dizimi vardır.

```
module TimeServerService{

    interface TimeClient
    {
        attribute string name;
        attribute double currentTime;
        void TimeAdvaceGrant();
    };

    interface TimeServer{

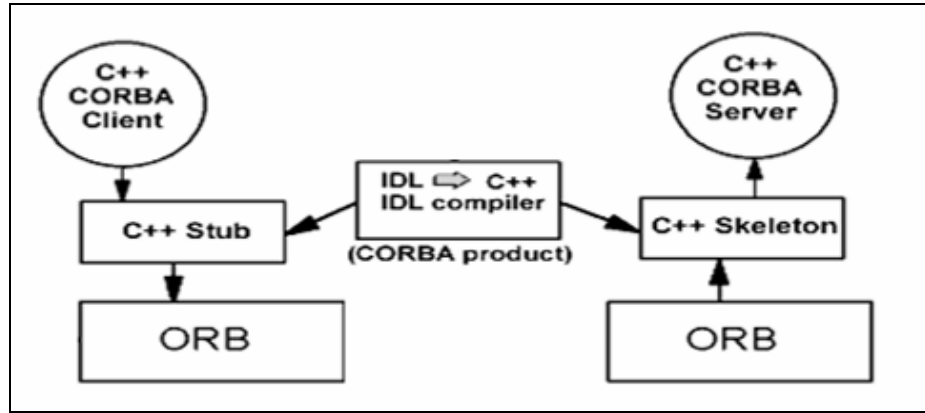
        double StartSimulationTime ( in TimeClient entityClient);
        double StopSimulationTime ( in string entity);
        void AdvanceTime(in string entity, in double time);
    };

};
```

Şekil 2.1 IDL Örneği.

CORBA alt yapısını kullanarak yapılan her çalışmada öncelikle kullanılacak ortak servisler ve bileşenler IDL ile tanımlanmalıdır. Bu tanımlamalardan sonra ilgili yapılara sunucu ve istemci taraflarından erişebilmek için gerekli bağlantı modüllerinin sınıf yapıları oluşturulmalıdır. İlgili yapılara erişim bu dönüştürülmüş bağlantı modülleri tarafından yapılır.

IDL ile tanımlanan nesnelerin geliştirme diline dönüştürülmesi işlemi IDL Dönüştürücü (IDL Parser) ile Şekil 2.2’te görüldüğü gibi yapılır. IDL Dönüştürücü, çalıştırıldığında IDL tanımlarında bulunan nesnelerin Sunucu (Server) tarafında çalışacak kısmı için geliştirme dilinde Skeleton, İstemci (Client) tarafında yine geliştirme dilinde Stub oluşturur. Bu üretilen yapılar altyapıda bulunan ORB üzerinden bağlantı işlemlerini yerine getirir ve sunucu-istemci bağlantısını sağlamış olurlar.



Şekil 2.2 IDL Dönüştürme İşlemi.

Dağıtık Benzetim Sisteminde de kullanılacak dağıtık servisler IDL ile tanımlanacak ve benzetim içinde bu servisler ile işlemler yürütülecek böylece sistem genelinde CORBA ile bağlı bütün yapılar aynı servisleri kullanabileceklerdir.

2.3.2 ORB ve CORBA Servisleri

ORB, CORBA mimarisin kaburgasıdır. Şekil 2.2’te görüldüğü gibi IDL ile ifade edilen arayüzlerin erişecekleri asıl nesnelerin erişimini sağlar. Bunu dağıtık makinelerin paylaşımlı yapılara ulaşması ve diğer makinedeki bilgileri almasını sağlayarak yapar. Ağa gönderilecek ve alınacak paketlerin takibi işlemlerini yürütür. Dağıtık sistemde çalışan dağıtılmış nesnelerin senkron bir şekilde bütün uç

birimlerde bulunmasını sağlar. Eğer nesneler farklı dillerde yazılmış iseler dillerin birbirine çevrilmesi işlemlerini yürütür. Uzak makinedeki metodların çağrılmasını sağlar. ORB sayesinde benzetim sisteminde alt bağlantı işlemleri ve nesne sorgulamalarının yapılabileceği bir yapı sağlanmış olur.

CORBA servisleri, CORBA ya fonksiyonellik sağlamak amacıyla yapılan eklenti servislerden oluşur. Bu servisler arasında bazı temel işlevi olanlar da vardır. Örneğin İsimlendirme Servisi birçok ORB içinde standart olarak bulunan bir servistir. Bu servis benzetim sistemlerinde de kullanılabilir. Ayrıca olay servisi ve zaman servisi dağıtık benzetim sistemlerde ihtiyaçlara cevap verebilecek servislerdir.

2.3.2.1 İsimlendirme Servisi

İsimlendirme Servisi, CORBA içinde standart haline gelmiş bir servistir. Servisin görevi isimlerden nesnelere referans döndürmektir. İsimlendirme servisinde yapılan bir isimde eğer bir referans daha önceden kaydedilmiş ise o zaman servis nesnenin referansını geri döndürür. Çalışması DNS (Domain Name Service) çalışmasına benzer. DNS tanımlamasında verilen bir ismin gerçekte hangi adrese karşılık geldiği tutulur ve geri döndürülür. Örneğin

www.hotmail.aate.nsatc.net 213.19.160.188

ikilisini içeren bir DNS'e www.hotmail.aate.nsatc.net adresi sorulduğunda 213.19.160.188 adresi geri döndürülür.

İsimlendirme Servisi ile değişik federasyon tanımlamaları yapılabilir. Böylece sistemin genişleyebilirliği ve incelenebilirliği artar. Tanımlanan federasyonlar ya hep birbirine bağlı ya da hiyerarşik bir yapıda bağlı olabilir. Hep birbirine bağlı olan yapıda bütün yayınlanacak nesneler her federasyon üyesine gönderilir ve her üye ilgili nesneye doğrudan erişebilir. Hiyerarşik yapıda ise nesnelere sadece sürekli erişim sağlayacak federelerin erişimine açıktır. Kendi erişimine kapalı olan bir nesneye, ilgili federe üzerinden erişim sağlanır.

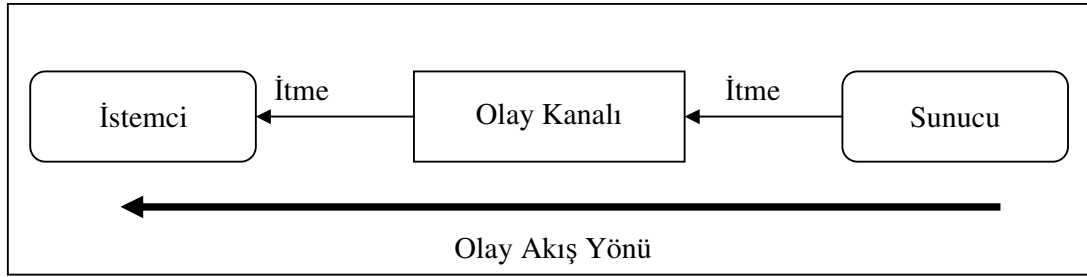
2.3.2.2 Olay Servisi

CORBA servislerinden dağıtık benzetimler içinde kullanılabilecek en önemli servislerinden birisi olay servisidir. Bu servis aracılığıyla sistem içinde her hangi bir

anda asenkron olay gönderilebilmektedir. Bu sayede benzetim bileşenleri birbiriyle koşum esnasında haberleşebilmekte karşılıklı mesajlaşmalarda bulunabilmektedir.

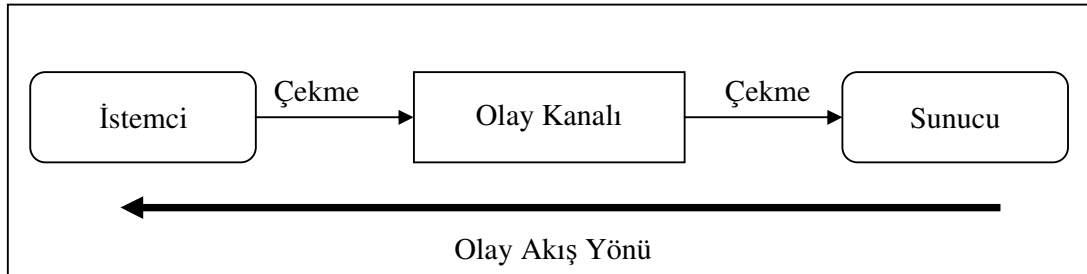
Olay servisinin çalışması temelde bir olay kanalının tanımlanması, bu olay kanalına alıcı ve verici olarak değişik bileşenlerin kayıt olması ve ilgili haberleşmelerin yapılarak olayların iletilmesi ile yapılır. Sistem içinde bir çok olay kanalı tanımlanabilir.

CORBA olay servisi içinde 2 değişik olay tanımlama modeli vardır. Bunlar Çekme stili ve İtme stili modellerdir. Bu iki temel model haricinde değişik karma modeller de mevcuttur.



Şekil 2.3 İtme Stili Olay Akışı.

İtme stili olay yönetimi, Üreticinin her hangi yayınlayacağı bir olay oluştuğunda Olay Kanalına bildirimde bulunması ve Olay Kanalının ilgili olay bilgisini Alıcıya iletilmesiyle yapılır. Şekil 2.3'te akış şeması görülmektedir. Akış üreticiden alıcıya asenkron olarak işler, Alıcı herhangi bir anda olay bilgisi alabilir. Bu stilde Üretici aktif çalışmada, Alıcı pasif çalışmadadır.



Şekil 2.4 Çekme Stili Olay Akışı.

Çekme stili olay yönetimi olayla ilgilenen Alıcının sürekli olay hakkında olay kanalını sorgulaması şeklinde yapılır. Olay kanalı kendisine alıcı tarafından yapılan olay bilgisi isteğini Üreticiye olay olup olmadığını sorgular ve sonucu

Alıcıya geri döndürür. Burada Alıcının olaylardan haberdar olması senkron güncellemelerle sağlanır. Şekil 2.4'teki gibi Akış yönü Alıcıdan Üreticiye sorgulama ve Üreticiden Alıcıya güncelleme gönderme şeklindedir. Bu stilde Üretici pasif, Alıcı aktif çalışmadadır.

2.3.2.3 Zaman Servisi

Zaman servisi, CORBA tasarımları içinde zaman akışının düzenli işlemlerini sağlar. CORBA ortamında gönderilen olayların senkronizasyonu, zamana bağlı olayların oluşturulması, oluşan olaylar arasındaki zaman farklarının hesaplanabilmesi amacıyla oluşturulmuştur. Benzetim döngüsündeki zaman akışının zaman servisi ile düzenlenebilmesi mümkündür. Bu servisle benzetim bileşenleri arasındaki olayların senkronizasyonu da çözülmüş olur.

Zaman servisi içinde 2 ana servisi barındırır. Bunlar Zaman Servisi ve Olay Zamanlayıcı Servistir. Zaman Servisi sistemdeki UTO(Universal Time Objects) ve TIO (Time Interval Objects)'ları kontrol eder. UTO'lar sistem saatini ifade eder. Olay Zamanlayıcı Servis ise Olay Zamanlama Denetleyicilerini kontrol eder. Olay Zamanlama Denetleyicilerinin her biri bir Olay Kanalı ile ilişkilendirilir. Bu kanal içinde gönderilen bütün olaylar zaman değeri taşırlar.

3 PERFORMANS ANALİZİ

Performans analizini gerçekleştirebilmek için öncelikle değişik altyapılar üzerinde benzetime bakan yönlerini ele alan incelemeler yapılmıştır. Elde edilen sonuçlardan yola çıkarak benzetim sistem altyapıları için ortak özellikler belirlenmiştir. Bu özelliklerden yola çıkarak bütün altyapılar için performans analizlerinde kullanılabilecek metrik tanımlamaları yapılmıştır.

Altyapılar için ölçülecek metrikler belirlendikten sonra bu metriklerle altyapıların karakterize edilebilmesi için bir dizi test işlemi tanımlanmıştır. Bu işlemlerin yürütülmesi sayesinde altyapıya ait karakterin ortaya çıkması amaçlanmıştır. Performans analizi kapsamında yapılan ve genel hatlarıyla ifade edilen çalışmalara ait ayrıntılar ilerleyen başlıklarda verilecektir.

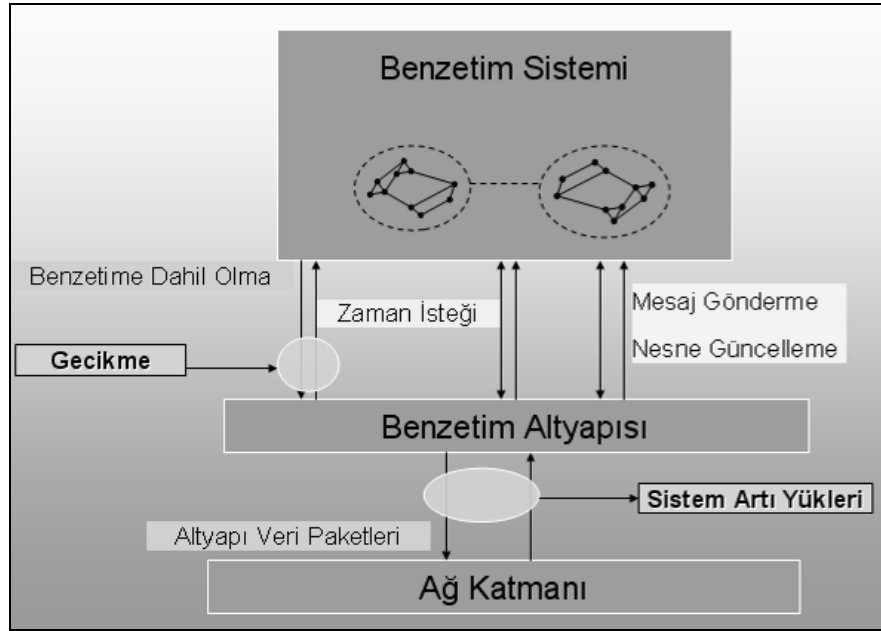
3.1 Performans Metrikleri

Ele alınan dağıtık sistem altyapılarıyla yapılan örnek geliştirmelerde benzetim altyapılarının Benzetim Sistemi ile Ağ Katmanı arasında bir ara katman görevi gördüğü belirlenmiştir. Altyapının her iki tarafla birlikte oluşturduğu arayüz fonksiyonları Çizelge 3.1’de verilmiştir.

Benzetim Sistemi-Benzeti Altyapısı	Benzeti Altyapısı-Ağ Katmanı
Benzetim gurubu oluştur Benzetime dahil olma isteği-kabulü Benzetimden çıkma isteği- kabulü Nesne yayımlama yapma isteği-kabulü Nene istek bildirimi yapma-kabulü Zaman yönetimi başlat Zaman artırımı isteği-kabulü Mesaj gönderim isteği Mesaj güncelleme bilgisi	Paket gönderme Paket alma

Çizelge 3.1 Benzetim Altyapısı Arayüzleri.

Daha önce yapılmış analiz çalışmaları da göz önünde bulundurularak altyapının oluşturduğu arayüzlerde benzetim performansına etki edebilecek 2 kriter seçilmiştir. Bu kriterler Şekil 3.1'da görüldüğü gibi benzetim sisteminin, benzetim altyapısına yapacağı her isteğin kabul edilmesi zamanına kadar oluşan *Gecikme* ve benzetim altyapısının ağ katmanına göndereceği her temel benzetim bilgisi için veri paketleri üzerine ekleyeceği *Sistem Artı Yükleridir*. Geliştirilecek performans analizi sisteminde bu iki metrik üzerinden ölçümlendirmeler yapılacak ve elde edilen ölçümlendirmelerle altyapılara ait karakterler çıkartılabilecektir.



Şekil 3.1 Benzetim Altyapıları Performans Metrikleri.

Performans Metriklerinin ölçülmesi ve altyapı karakterlerinin ortaya çıkarılabilmesi için bir dizi deney tasarlanmıştır. Bu deneylerin belirlenen metrikler üzerinden sonuçlarına göre altyapıları değerlendirmek mümkün olacaktır.

- Benzetim sisteminin zaman yönetimi performansının,
 - sisteme her yeni benzetim federesi eklendiğinde,
 - sistemde paylaşılan toplam veri boyutu arttığında,
 - sistemde paylaşılan toplam değişken sayısı arttığında değişimi.
- Benzetim sisteminin veri yönetimi performansının,
 - sisteme her yeni benzetim federesi eklendiğinde,

- sistemde paylaşılan toplam veri boyutu arttığında,
- sistemde paylaşılan toplam değişken sayısı arttığında değişimi.

Burada sıralanan benzetim sistemine yeni benzetim federelerinin eklenmesi, benzetim sisteminde birim zamanda paylaşılan toplam veri boyutunun değişmesi ve birim zamanda sistemde bulunan toplam değişken sayısının değişmesi, benzetim boyunca sürekli meydana gelebilecek ve sistem yoğunluğunu etkileyebilecek etkiler olduğundan dolayı değerlendirme de esas alınacak deneyler olarak seçilmişlerdir. Geliştirilen performans analizi sistemi yukarıda verilen değerlendirme deneylerinin gerçekleştirilebildiği bir ortam olarak hazırlamıştır.

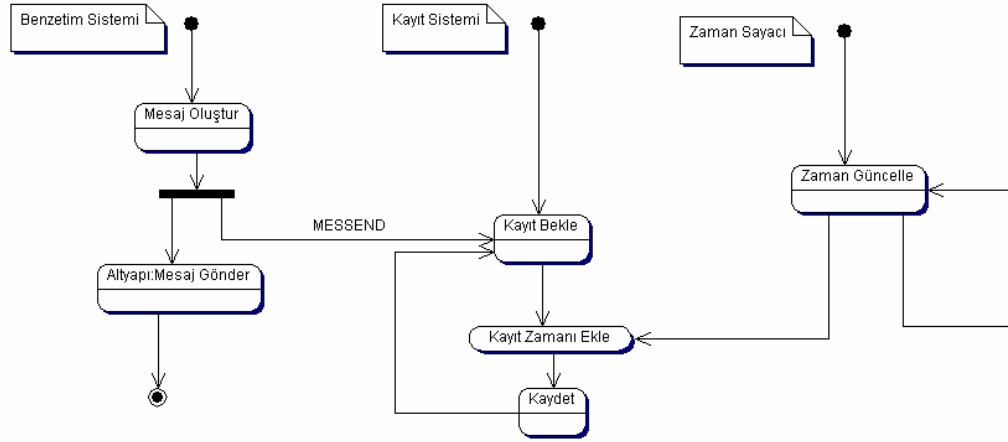
Metriklerin analiz çalışmalarında değerlendirilebilmesi için gerekli metrik kayıtlarını tutmak üzere, ağ ortamında dağıtık çalışabilen bir senkron kayıt sistemi tasarlanmıştır. Kayıt sistemi hakkında bilgiler geliştirilen sistemler başlığı altında verilecektir. Kayıt sistemi aracılığıyla benzetim koşumu sırasında analiz sistemi tarafından kullanılacak seçilmiş kayıtların tutulması sağlanacaktır. Tutulacak kayıtlar MESSEND, MESRECEIVED, TIMEREQUEST, TIMEGRANT tiplerini içerir. Bu mesaj tiplerinin açıklamaları Çizelge 3.2’te verilmiştir.

Kayıt Tipi	Açıklama
MESSEND	Gönderilen her mesaj için gönderirliği anda tutulur
MESRECEIVED	Alınan her mesaj için alındığı anda tutulur
TIMEREQUEST	Zaman artırım isteği anında tutulur
TIMEGRANT	Zaman artırımı kabul edildiğinde tutulur

Çizelge 3.2 Analiz için Tutulacak Kayıt Tipleri.

Bütün kayıtlar Şekil 3.2’de verilen genel işleyiş çerçevesinde kayıt sistemi zamanına uygun olarak etiketlenir. Örneğin benzetim sisteminin çalışması sırasında altyapıya gönderilen her mesaj için gönderme anında Kayıt Sistemine kayıt isteğinde bulunur. Kayıt Sistemi içinde bulundurduğu Senkron Zaman Sayacı aracılığıyla

güncellediği zamana uygun olarak kendisine gelen kayıt bilgisini etiketler ve dosyaya bilgiyi kaydeder.



Şekil 3.2 Mesaj Kayıt Sistemi.

Analiz sistemi kapsamında geliştirilen yapılara ait ayrıntılı tasarım ve kullanım bilgileri bir sonraki bölümde ele alınacaktır.

3.2 Performans Analizi İçin Geliştirilen Sistemler

Performans analiz sisteminin geliştirilmesinde belirlenen performans metrikleri ve deneyleri göz önünde bulundurulmuştur. Geliştirilen analiz sisteminden beklenen başlıca yetenekler şunlardır:

- Her altyapı sistemi için çalışabilmeli
- Çalışma sırasında ağ yoğunluğunu artırmamalı
- Dağıtık yapıda çalışabilmeli
- Metriklere ait kayıtları tutabilmeli
- Dağıtık makinelerdeki kayıtlar eş zamanlı olmalı
- Federe sayılarının değişimlerinde oluşan performans değişimleri ölçülebilmeli
- Değişik mesaj yoğunluklarında oluşan performans değişimleri ölçülebilmeli
- Metriklere ait değerlendirme grafiklerini oluşturabilmeli

Belirtilen yeteneklerin sağlanabilmesi için çok parçalı modüler bir yapı geliştirilmiştir. Yapı temel olarak Şekil 3.3'te görüldüğü gibi Altyapı Çevirici, Altyapı Adaptörü, Eş Zamanlı Kayıt Sistemi ve Kayıt Analiz Aracı modüllerini içermektedir. Altyapı Çevirici: Performans analiz sistemini altyapıdan bağımsız hale getiren yapıdır, adaptörler aracılığıyla altyapının kullanılmasına imkan sağlar. Altyapı Adaptörü: Çeviricinin altyapıyı kullanabilmesi için gerekli tanımlamaları içeren bir kütüphanedir. Her altyapının ilgili adaptörü yazılarak analizi yapılabilir. Eş Zamanlı Kayıt Sistemi: Benzetimden bağımsız olarak performans için kritik değerleri kayıt etmekle görevli, ağ üzerinde senkron kayıtlar tutabilen bir kayıt sistemidir., Kayıt Analiz Aracı: Kayıt edilen kritik değerlerin analiz edilmesini ve değişik grafiklerin elde edilmesine olanak sağlayan bir araçtır. Analiz sisteminin farklı bir benzetim altyapısı üzerinde işlevliğinin gösterilebilmesi için çalışmalar sırasında bir de RAM DBA (Dağıtık Benzetim Altyapısı) geliştirilmiştir. Ayrıca performans analizlerinin yapılabilmesini sağlayacak değişkenlerin güncellenebildiği bir molekül benzetim sistemi de analiz sistemine dahil edilmiştir. Proje çalışmaları nesne tabanlı bir yaklaşımla tasarlanmış ve Microsoft Visual Studio 8.0 ortamında geliştirilmiştir.

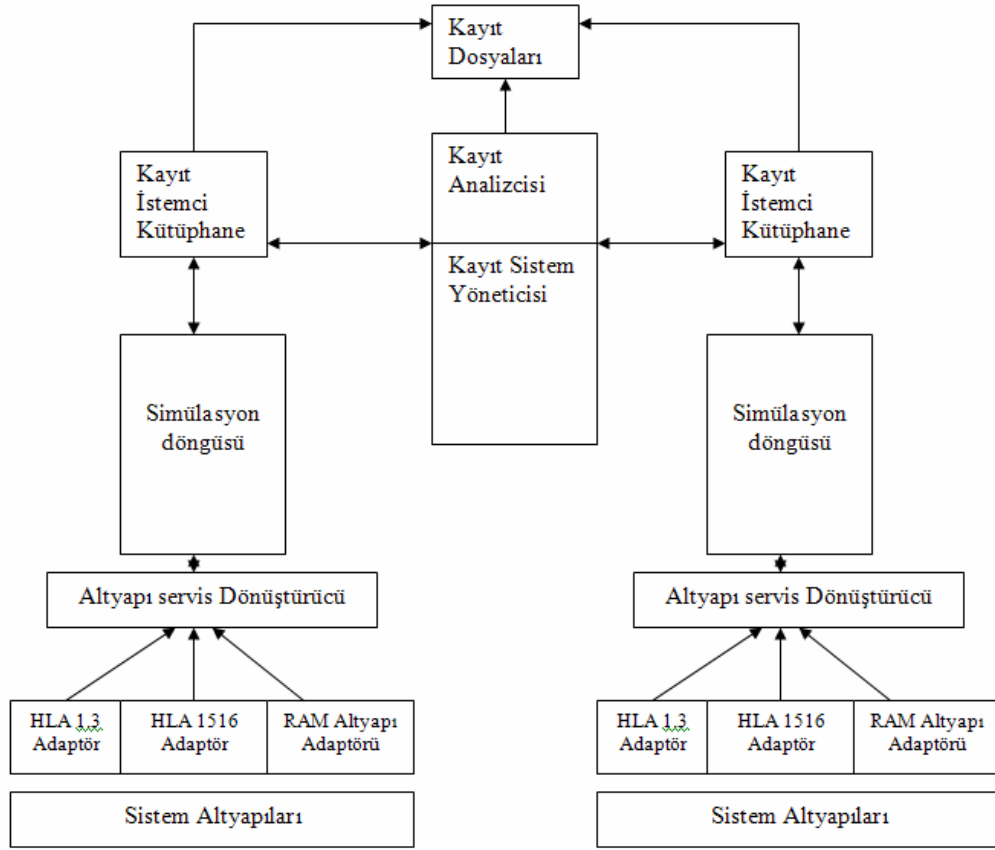
3.2.1 RAM Dağıtık Benzetim Altyapısı

Performans analizlerinde kullanılmak amacıyla, incelenen HLA Yazılımlarının ortak özellikleri ve dağıtık sistem metodolojisi göz önünde bulundurularak bir benzetim altyapısı geliştirilmiştir.

3.2.1.1 Beklenen Yetenekler

RAM DBA'dan beklenen temel yetenekler şu şekilde belirtilebilir.

- Ağ üzerinden haberleşme işlerini yapabilmeli
- Benzetimler için bir federasyon ortamı oluşturmali
- Federelerin federasyona katılmasını sağlamalı
- Federeler arası mesaj iletimini sağlamalı
- Federeler arası zaman senkronizasyonunu sağlamalı



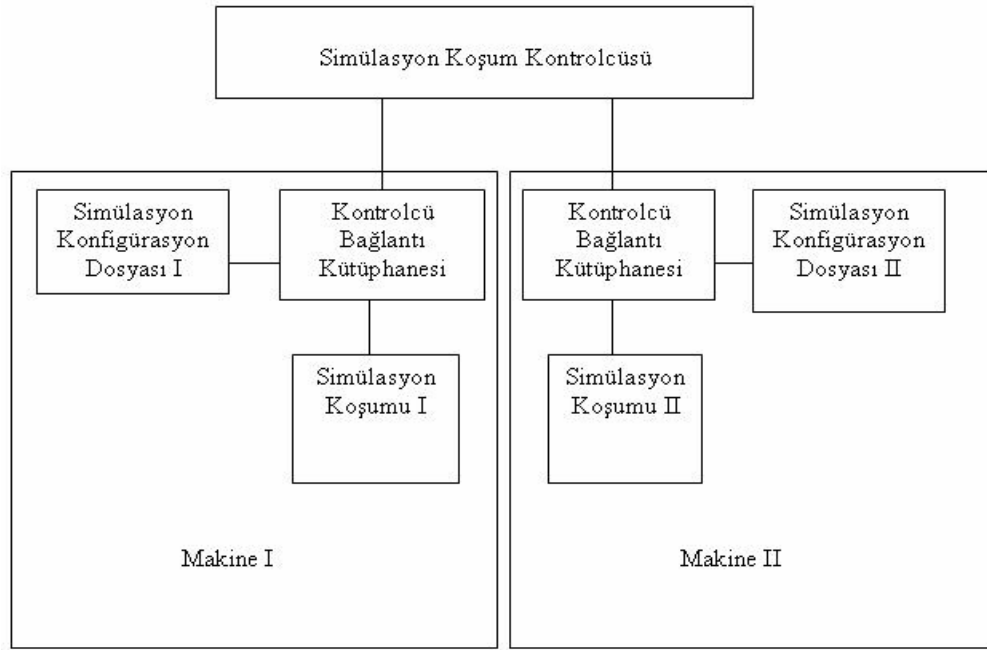
Şekil 3.3 Performans Analiz Sistemi Genel Şema.

3.2.1.2 Sistem Tasarımı

RAM Benzetim Altyapısı Microsoft Visual Studio 2005 ortamında C++ programlama dili ile, ACE (Adaptive Communication Environment)-TAO (The ACE ORB) [8] kütüphaneleri kullanılarak geliştirilmiştir. Geliştirilen sistem sağladığı genel amaçlı sistem tanımlama yapısıyla her türlü benzetimin geliştirilmesine olanak sağlayacak şekilde tasarlanmıştır. Sistem benzetim zaman yönetimini, veri iletim yönetimini, sisteme yeni bileşenlerin katılıp ayrılmasını sağlayan bileşen kontrol yönetimini başarıyla sağlayabilmektedir. Sistem ve alt bileşenleri bir kütüphane olarak geliştirilmiştir.

Sistemin genel şeması Şekil 3.4’de görüldüğü gibidir. Benzetim koordinasyon işlevi, benzetim koşum koordinatörü ve benzetimle koordinatör arasındaki bağlantı işlerini yapmak üzere hazırlanmış bağlantı kütüphanesiyle yerine getirilir. Koşum

kontrolcüsüne ve kontrolcü bağlantı kütüphanesine ait görevli birimler Şekil 3.5’da belirtilmiştir.



Şekil 3.4 RAM DBA Genel Bağlantı Şeması.



Şekil 3.5 RAM DBA Bağlantı Koordinatörü ve Bağlantı Kütüphanesi Şeması.

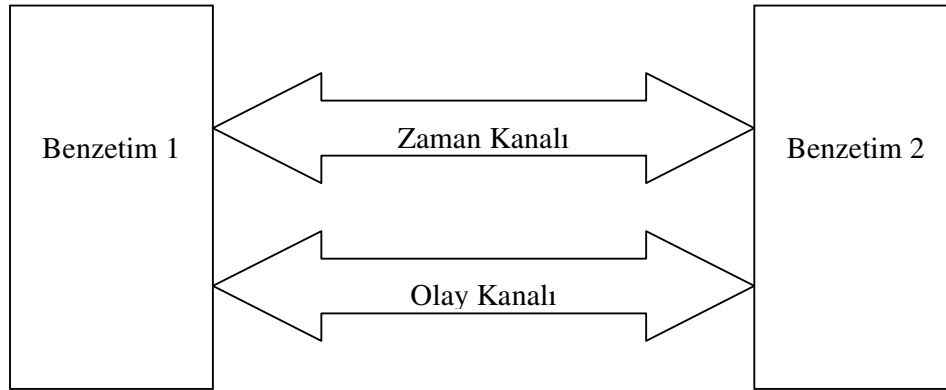
Benzetim koşum kontrolcü, birlikte çalışmak isteyen benzetim parçalarının kendilerini kayıt ettirebilecekleri ve gurubun paylaşımına sunacakları nesneleri işlettirebilecekleri bir mekanizmaya sahiptir. Zaman yönetimi de yine bu koordinatör aracılığıyla yapılır. Şekil 3.5’de görüldüğü gibi koordinatörün çalışması için hayati önem taşıyan 4 modül geliştirilmiştir. Bu modüller dağıtık benzetim koşumlarındaki zaman yetkilendirmesini yürüten zaman yöneticisi, ağ erişim yöneticisi, federasyona

yeni federelerin katılması ve ayrılmasını yürüten bileşen yöneticisi ve bileşenler arasındaki haberleşme mekanizmasını oluşturan veri yöneticisidir.

Benzetim koordinasyonunun diğer tamamlayıcı unsuru benzetimlerin koordinasyon çevrimine katılabilme ve veri paylaşabilme işlerini benzetimler için yürüten bağlantı kütüphanesidir. Bağlantı kütüphanesi, altyapının uygulanacağı benzetim sistemine ait birlikte koşum için gerekli yapılandırma bilgilerini okuyarak benzetim adına koordinasyon işlerini yürütür. Bağlantı kütüphanesi kapsamında geliştirilen modüller şu şekilde sıralanabilir:

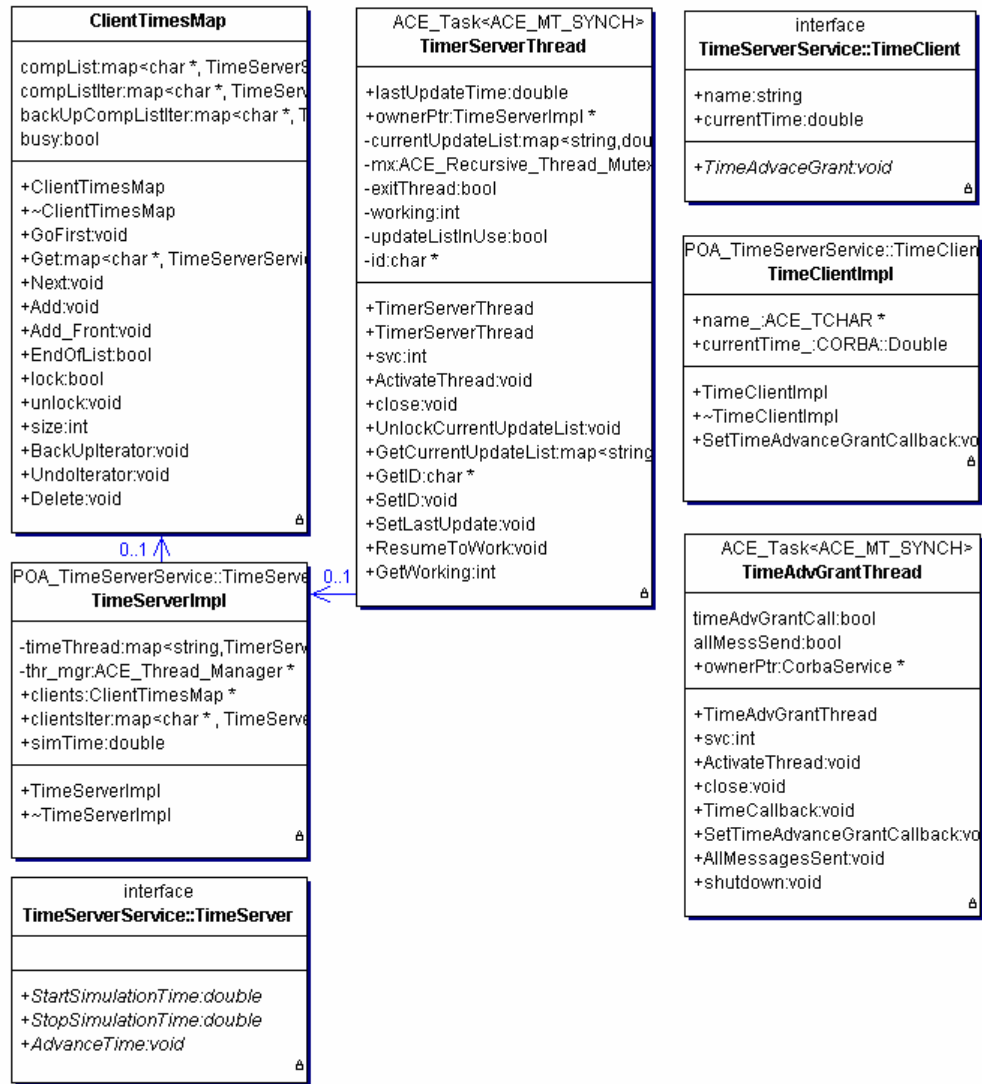
Altyapı arayüzü: Altyapının ağ bağlantıları hariç yürüttüğü bütün işler bu arayüz aracılığıyla yapılmaktadır. Benzetim sistemi bu arayüzle altyapıdan kendi işletimini soyutlayarak tasarımını basitleştirmektedir.

Zaman ve Veri Takipçileri: Benzetim sistemine ait güncelleme bilgilerinin alınması, benzetime iletilmesi ve benzetimin yaptığı güncelleme isteklerinin koordinatöre iletilmesi işlevlerini yürütürler. Şekil 3.6'da görüldüğü gibi benzetim altyapısının gerçekleştirilmesi için Zaman Yönetimi, Nesne Yönetimi için CORBA Olay servisi kullanılarak 2 olay kanalı tasarlanmıştır. Bu kanallar üzerinden birisi kullanılarak zaman yönetimi sağlanılmakta, diğeri kullanılarak nesne yönetimi sağlanılmaktadır.



Şekil 3.6 Altyapıda Kanal Kullanımı.

Zaman yönetimi için geliştirilen zaman servisi sınıf şeması Şekil 3.7'de görüldüğü gibidir. Zaman yönetimi ClientTimesMap, TimeServer, TimeClient, TimeServerThread, TimeAdvGrantThread nesneleriyle yapılır.



Şekil 3.7 Altyapı Zaman Yönetimi Sınıf Şeması.

ClientTimesMap: Zaman yönetimine bağlı bulunan istemcilerin adreslerinin ve zaman bilgilerini tutan listedir. Zaman güncellemeleri bu liste üzerinden yapılır.

TimeServer: Zaman yönetimini sağlayan CORBA zaman sunucusudur. İstemci istekleri doğrudan bu sunucu nesneye gelir.

TimeClient: İstemcilerin zaman artırım isteği yapabilmek için kullandıkları CORBA zaman istemcisidir.

TimeServerThread: Zaman artırım isteklerini kontrol eder ve gerekli artırım isteklerini kabul cevabı göndererek onaylar. Zaman bilgilerinin güncellenmesini ClientTimesMap listesi üzerinden yapar.

TimeAdvGrantThread: İstemci tarafında, zaman sucusundan gelen artırım isteği kabulünü beklerken yapılacak bütün mesaj güncellemelerinin yapılmasını sağlayan ve sonra da zaman artırımını gerçekleştiren nesnedir.

Her benzetim federesi bağlanmak istediği federasyona ait zaman sunucusuna kendi ismini ve zaman bilgisini kaydettirir. Sonra bu zaman ilerleme istekleri zaman olay kanalı aracılığıyla sunucuya iletilir ve güncelleme bilgisi gelene kadar diğer benzetim sistemlerinden gelen mesajlar alınır. Zaman sunucusu, federeye ait zaman ilerleme isteğini, ilgili zamana kadar bütün mesajlar alınıncaya veya gönderilinceye kadar vermez. Böylece zaman planında senkron çalışan benzetimlerin mesajlarını da ilgili zamanda almaları sağlanmış olur.

Veri yönetimi için hazırlanan sistemin sınıf şeması Şekil 3.8’de verilmiştir. Olay yönetimini yürüten nesneler EventManagerServer, EventManagerClient, EventBase, EventParameter, MessageSenderThread, MessageEraserThread’dir.

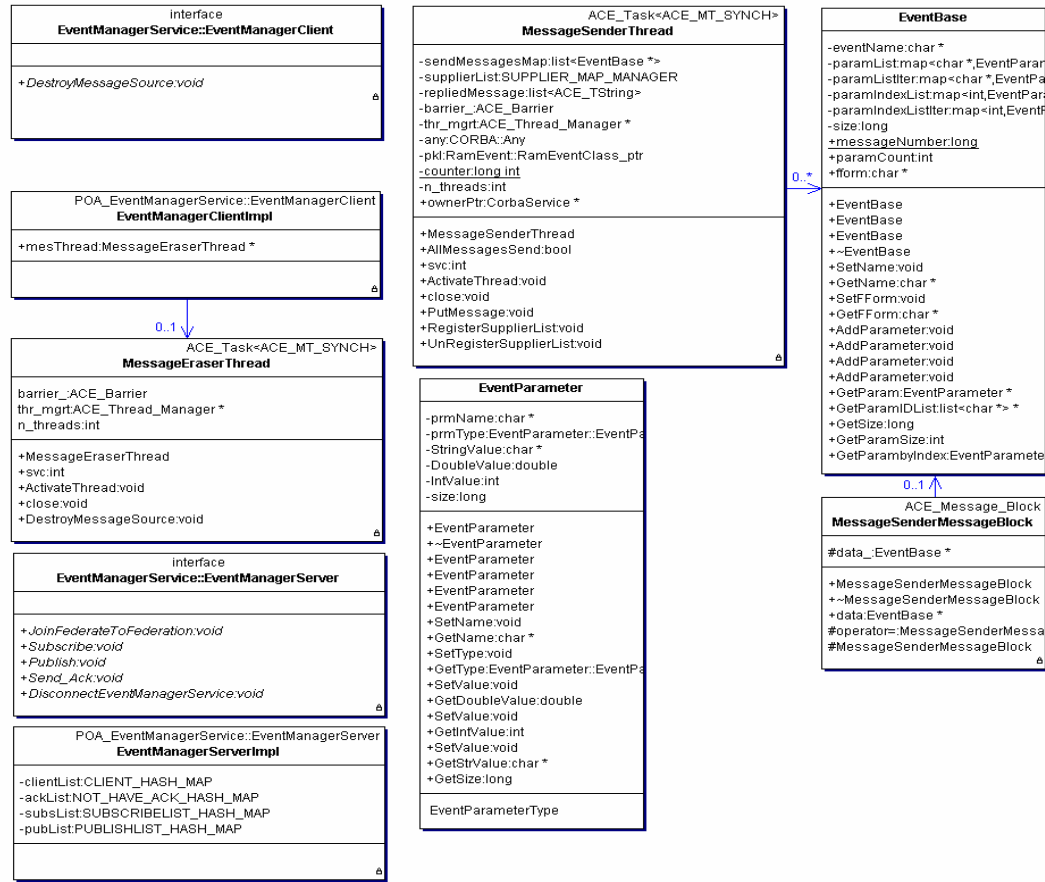
EventManagerServer: Federasyona katılan federe listesini, hangi federenin hangi nesneleri isteyip, hangilerini yayınlayacağı bilgilerini tutar. Her hangi bir federenin yayınlayacağı bilgi doğrudan bu nesneye gelir ve nesne tarafından ilgili federelere yönlendirilir.

EventManagerClient: Her federe için bir tane bulunur. Federeye ait yönetim işlerinde sunucuya kayıt olma, mesaj gönderme, alma işlerinde arayüz oluşturur. Aldığı her mesaj için sunucuya geri bildirim gönderir.

EventBase: Altyapıda gönderilen genel mesaj yapısıdır. Her federe aynı tip mesajlar gönderebilir.

EventParameter: EventBase nesnesi tipindeki mesajlar için parametre bilgilerini tutar.

MessageSenderThread: Mesaj gönderen bir federe için mesajın ihtiyaç duyan bütün federelere ulaştırılması görevini yapar.



Şekil 3.8 Veri Yönetimi Sınıf Şeması.

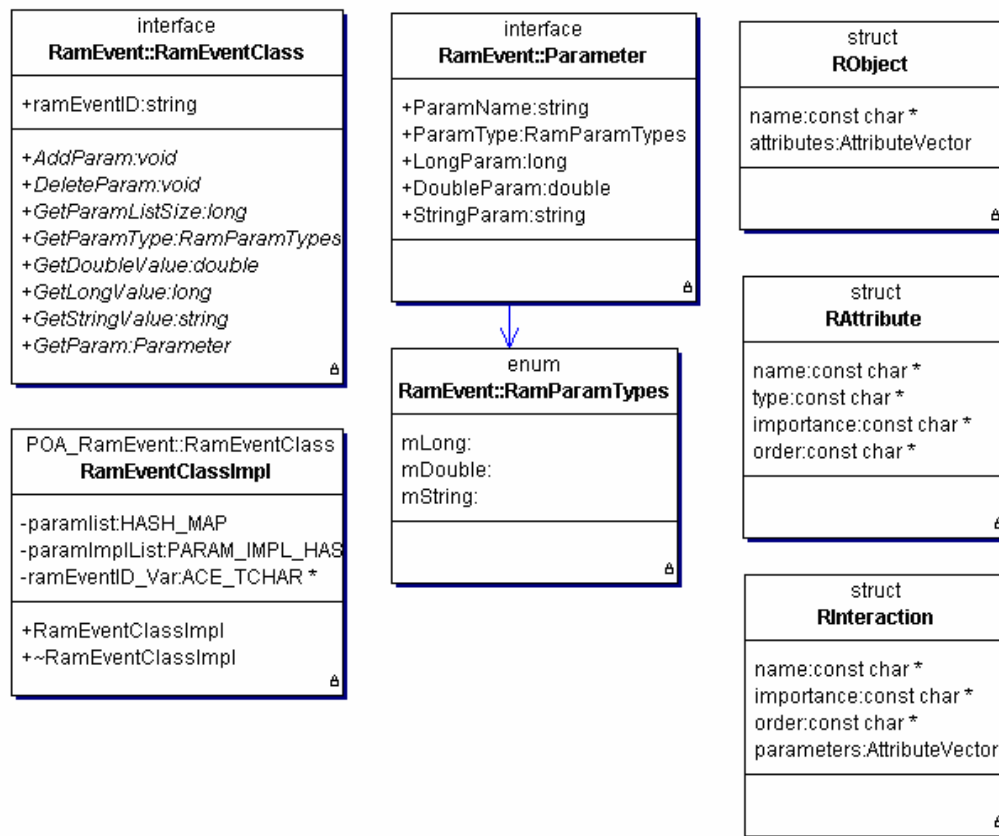
MessageEraserThread: Mesaj gönderen federede MessageSenderThread tarafından gönderilen ve bütün gerekli federelere ulaştırıldı bilgisi EventManagerServer tarafından bildirilen ve işi biten mesajların silinmesini sağlar.

Her federe veri göndermek ve güncellemek için bir veri sunucu nesnesi oluşturur. Bu nesne sürekli gönderilecek nesneleri ilgili formata çevirerek mesaj kanalına gönderir. Yine her federe federasyondan mesaj güncellemesi alacaksa o zaman bir de veri istemci nesnesi oluşturur. Bu nesne sunucudan gelen mesajı standart formattan bileşen mesajı haline dönüştürür ve bileşenin mesaj kutusuna koyar. Sunucu gönderilecek mesajın gerekli bütün hedeflere ulaştığını sahip olduğu yayınlama istek veri tabanından kontrol eder ve bütün uç birimlere gönderilmiş verileri temizler. Altyapı kendi bileşenleri arasında haberleşmesini kendi özel tanımlı genel amaçlı mesaj yapılarıyla yapar.

RAM DBA servisi içinde haberleşmelerde kullanılan temel mesaj yapısı Şekil 3.9'de görüldüğü gibidir. Burada RamEventClass ana mesaj sınıfını oluşturmaktadır.

RamEventClass, Parametre listeleri ile olay değişkenlerinin bilgilerini taşır. RObject, RAttribute, RInteraction nesneleri, konfigürasyon dosyalarından okunan tanımlama bilgilerini taşırlar. Burada parametrelerden oluşan ileti tanımlamaları yapılabilmektedir.

Konfigürasyon Dosyası ve Veri Analiz Yöneticisi: Benzetim sisteminin sağlıklı bir şekilde çalışabilmesi, dağıtık benzetime katılabilmesi için gerekli yapılandırma bilgileri xml tabanlı tutulan bir yapılandırma dosyası aracılığıyla altyapı tarafından veri analiz yöneticisi ile okunur ve işletilir. Bu yapılandırma bilgileri çalışacak benzetimin bir benzetim grubuna katılması için gerekli koşacak benzetime ait federe adı, katılacak benzetim gurubu (federasyon) kimlik bilgileri, dağıtık benzetimde haber alınması istenilen nesne ve güncelleme bilgileri yer alır. Deneme amaçlı benzetim için hazırlanan konfigürasyon dosyası Şekil 3.10'deki gibidir.



Şekil 3.9 RAM DBA Mesaj Yapısı.

```

- <definitions>
- <head>
  <name>INFO</name>
</head>
- <body>
  <federation>MovingTopsFederation</federation>
+ <objects>
- <interactions>
  + <class>
  + <class>
  - <class>
    <name>NewFederateJoined</name>
    <importance>reliable</importance>
    <order>receive</order>
  - <parameter>
    <name>id</name>
    <type>string</type>
    <!-- Type String,Double,Int olabilir -->
  </parameter>
  </class>
</interactions>
</body>
</definitions>

```

Şekil 3.10 RAM Altyapı Konfigürasyon Dosyası.

3.2.1.3 RAM Kullanımı

RAM dağıtık benzetim altyapısı bir kütüphane olarak kolaylıkla değişik benzetim ortamlarında kullanılabilecek şekilde tasarlanmıştır. 'de ifade edilen dağıtık altyapı arayüzünü sağlamaktadır. Benzetim sisteminde RAM DBA'sını kullanmak istenildiğinde öncelikle konfigürasyon dosyasına nesne modellerine ilişkin tanımlamalar yazılmalıdır. Altyapı bu konfigürasyon dosyasını verilen bir adresten okuyabilmektedir. Benzetim tanımlamaları yapıldıktan sonra kütüphane açılıp içinden API kullanılarak Altyapı Arayüzünü sağlayacak bir nesne oluşturulur. Bu nesne üzerinden federasyona katılma, zaman isteği, mesaj iletimi gibi işlemler çalıştırılabilir. Dikkat edilmesi gereken önemli bir nokta Altyapının başlangıç atamalarını yapması için açılır açılmaz "InitializeRTI" komutu ile başlangıç işlemlerini yürütmesi sağlanmalıdır.

3.2.2 Altyapı Çevirici

Altyapı çevirici, performans analizleri için, geliştirilen benzetim sisteminin aynı şartlar altında değişik altyapıları kullanılabilmesini sağlayan ortak bir arayüz sunar. Bu ortak arayüze uygun geliştirilen ve sisteme kayıt olan adaptörler

aracılığıyla ilgili altyapı üzerinde deneylerin yapılması mümkün olmaktadır. Çevirici benzetim ortamına sunduğu arayüz ile Şekil 3.3’de görüldüğü gibi benzetim sistemini altyapıdan yalıtmaktadır. Sistem altyapısı ne olursa olsun, altyapı için yazılmış adaptör ile çevirici üzerinden altyapıdan beklenen işler yapılabilir. Çevirici, Microsoft Visual Studio 2005 ortamında C++ dili kullanılarak yazılmış bir kütüphanedir. Altyapı çevirici sahip olduğu arayüz ifadeleriyle temel anlamda herhangi bir benzetimin yapılmasına imkan tanımaktadır.

3.2.2.1 Beklenen Yetenekler

Altyapı çeviriciden beklenen, ortak altyapı arayüzüne uygun geliştirilmiş adaptörlerin belirli etiketlerle açılabilmesini sağlamaktır.

3.2.2.2 Sistem Tasarımı

Altyapı çeviricinin benzetim sistemine olan arayüzü ’de tanımlanan benzetim sistemi-benzetim altyapısı arayüz fonksiyonlarını içermektedir. Hazırlanan arayüz performans analizi için hazırlandığından altyapılara ait bütün fonksiyonel metotları içermemektedir. Altyapı çevirici aracılığıyla kullanılan altyapı arayüzü ’da görüldüğü gibidir. Altyapı Çeviriciye ait API açıklamaları EK-1 bölümünde verilmiştir.

```
#ifndef DistributionServiceInterfaceh
#define DistributionServiceInterfaceh
#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
class EventBase;
typedef void (*EventCallbackPtr)(EventBase *) ;
class DistributionServiceInterface{
public:
    virtual ~DistributionServiceInterface(){};
    virtual void SendEvent( char* _name, EventBase *eventMes,char * _mesunique ) = 0;
    virtual void SetTimeAdvanceGrantCallback( void f( bool ) ) = 0;
    virtual void Subscribe( char *id,char *_eventName, EventCallbackPtr ) = 0;
    virtual double StartSimulationTime( char* _id ) = 0;
    virtual double StopSimulationTime( char* entity ) = 0;
    virtual void InitializeRTI() = 0;
    virtual void CreateFederation(char *_name,char *_path) = 0;
    virtual void JoinFederation(char *_fedName, char *_fedName)=0;
    virtual void RegisterSyncPoint(char*,char*)=0;
    virtual bool IsSyncOk(char*) = 0;
    virtual void RequestTimeAdvance(double _timeStep)=0;
    virtual void Tick()=0;
    virtual double EnableTimeConstrained() = 0;
    virtual double EnableTimeRegulation(double _simTime,double _lookAhead) = 0;
    virtual bool IsTimeGranted() = 0;
    virtual double GetCurrentLBTS()=0;
    EventBase * CheckMessage(){ return NULL;};
};
#endif
```

Şekil 3.11 Dağıtık Altyapı Arayüzü.

```
extern "C" DLLIMPEX_Export DistributionServiceInterface * GetDistributionServiceObject(char * );
extern "C" DLLIMPEX_Export void DeleteDistributionServiceObject( );
```

Şekil 3.12 Altyapı Çevirici API.

3.2.2.3 Kullanımı

Altyapı çevirici kullanımında dikkat edilmesi gereken en önemli nokta kullanılacak adaptörün tanımlanmış ve ilgili kütüphanesinin hazır bulunması gerekmesidir. İlgili adaptöre ait kütüphane varsa o zaman geliştirilecek benzetim sistemine ait nesne tanımlamalarını adaptörün kullandığı konfigürasyon dosyalarına kuralına uygun bir şekilde yazmak gerekmektedir. Çevirici tarafından görevlendirilen adaptörler kendi konfigürasyon dosyaları üzerinden federasyon yönetimini sağlarlar. Her altyapının konfigürasyon dosyası farklılıklar içermektedir. Konfigürasyon dosyaları sisteme uygun şekilde güncellendikten sonra benzetim yazılımı içinde çeviri kütüphanesi içinden “GetDistributionObject(char * dll_name)” fonksiyonu kullanılarak ismi verilen kütüphanedeki altyapı adaptörünün oluşturulması sağlanır. Oluşturulan adaptör üzerinden dağıtık benzetim koşumu gerçekleştirilir.

3.2.3 Altyapı Adaptörleri

Benzetim altyapılarının kullanılması, dağıtık benzetim sistemlerinin geliştirilmesi sürecinde en yoğun iş yükünün gerektiği ve en yoğun kodlama ihtiyacı duyulan bölümlerdendir. Federelerin federasyona katılması ve ayrılması işlerinin yanında federasyonda yayımlanacak veya alınacak verilere ait yazılım modellerinin federe tanımlarına uygun bir şekilde oluşturulması, güncellemelerinin yapılması ve silinmesi işlemleri oldukça karmaşıktır. Her benzetimci kendisini sadece modelleme ve benzetim tasarımına verebilmek için genel amaçlı tasarlanmış federasyon arayüzlerine ihtiyaç duyar.

Geliştirilen Altyapı Adaptörleri hem benzetim tasarımcıları için ihtiyaç duydukları federasyon arayüzlerini sunmak hem de performans analiz sistemi bünyesinde altyapı çevirici aracılığıyla benzetim döngüsünde kullanılabiliyen yazılım kütüphaneleridir. Yazılımlar C++ programlama dili ile Microsoft Visual Studio 2005 ortamında geliştirilmiştir.

3.2.3.1 Beklenen Yetenekler

Altyapı adaptörlerinden beklenen yetenekler şunlardır:

- Altyapı tanımlarını benzetimden yalıtmalı
- Federe modellerine ait yazılım modellerini oluşturmalı
- Çevirici tarafından kullanılabilmeli.

3.2.3.2 Sistem Tasarımı

Performans Analiz Sisteminde kullanılacakları için geliştirilen adaptörler ‘de verilen altyapı arayüzüne uygun bir arayüz sağlamaktadırlar. Tez çalışması kapsamında DMSO RTI 1.3v6, PRTI1516le ve RAM DBA için altyapı adaptörleri yazılmıştır. Adaptörlere ilişkin genel yapısal bilgiler ve kullanım bilgileri EK -2’de verilmiştir.

3.2.3.3 Kullanımı

Adaptörler performans analizinde kullanılabilmek için analiz sistemi dosya yapısı içinde “library” klasörü altına koyulmalıdır. Bu sayede Altyapı Çeviri kütüphaneyi açabilir ve benzeti sistemini çalıştırabilir.

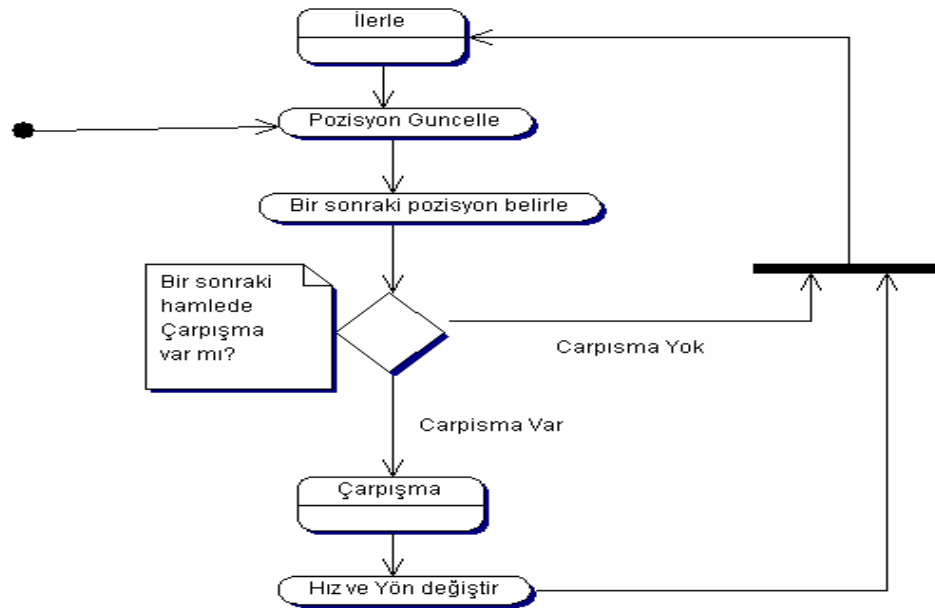
3.2.4 Örnek Hareketli Molekül Benzetimi

Performans incelemelerin yapılabilmesi için üzerinde değişiklik yapılabilecek bir benzetim ortamına ihtiyaç duyulmuştur. Bu ihtiyacı karşılamak için hareketli molekül benzetimi yapılmıştır. Hareketli molekül benzetiminde belirli bir hızla hareket eden moleküllerin çarpışmaları ve yön değiştirmeleri modellenmiştir. Benzetim zaman tabanlı ve durum bazlı olarak tasarlanmıştır. Benzetim sistemi Microsoft Visual Studio 2005 ortamında C++ dili ile kodlanarak geliştirilmiştir. Yazılım geliştirme sırasında ACE ve arayüz geliştirmeleri için QT kütüphaneleri kullanılmıştır.

3.2.4.1 Hareketli Molekül Kavramsal Modeli

Hareketli moleküller 2 durumlu olarak modellenmiştir. Bunlar Move (İlerleme) ve Crash (Çarpışma) durumlarıdır. Moleküllerin aktivite şeması Şekil 3.13’de

verilmiştir. Bu durumda benzetimi yapılacak her molekül benzetim boyunca sınırlı bir alan içinde hareket edecek ve çarpışma yapacaktır.



Şekil 3.13 Moleküllere Ait Aktivite Şeması.

3.2.4.2 Beklenen Yetenekler

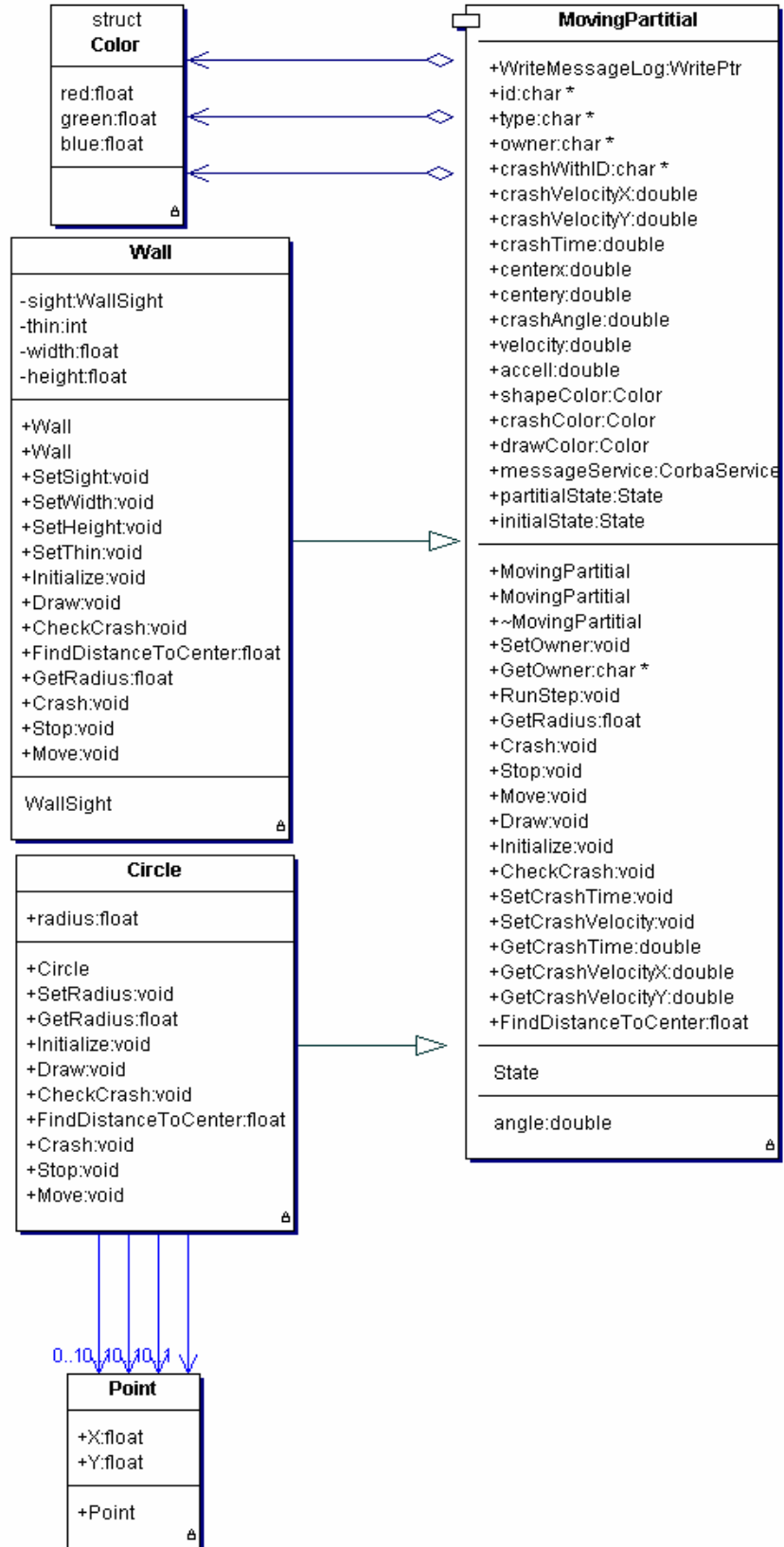
Geliştirilen benzetim sisteminden beklenen yetenekler şunlardır.

- Dağıtık çalışabilmeli
- Koşum sırasında bileşen oluşturup koşuma dahil edebilmeli
- Zaman tabanlı çalışabilmeli
- Sistem yoğunluğu değiştirilebilmeli
- Bileşen özellikleri değiştirilebilmeli

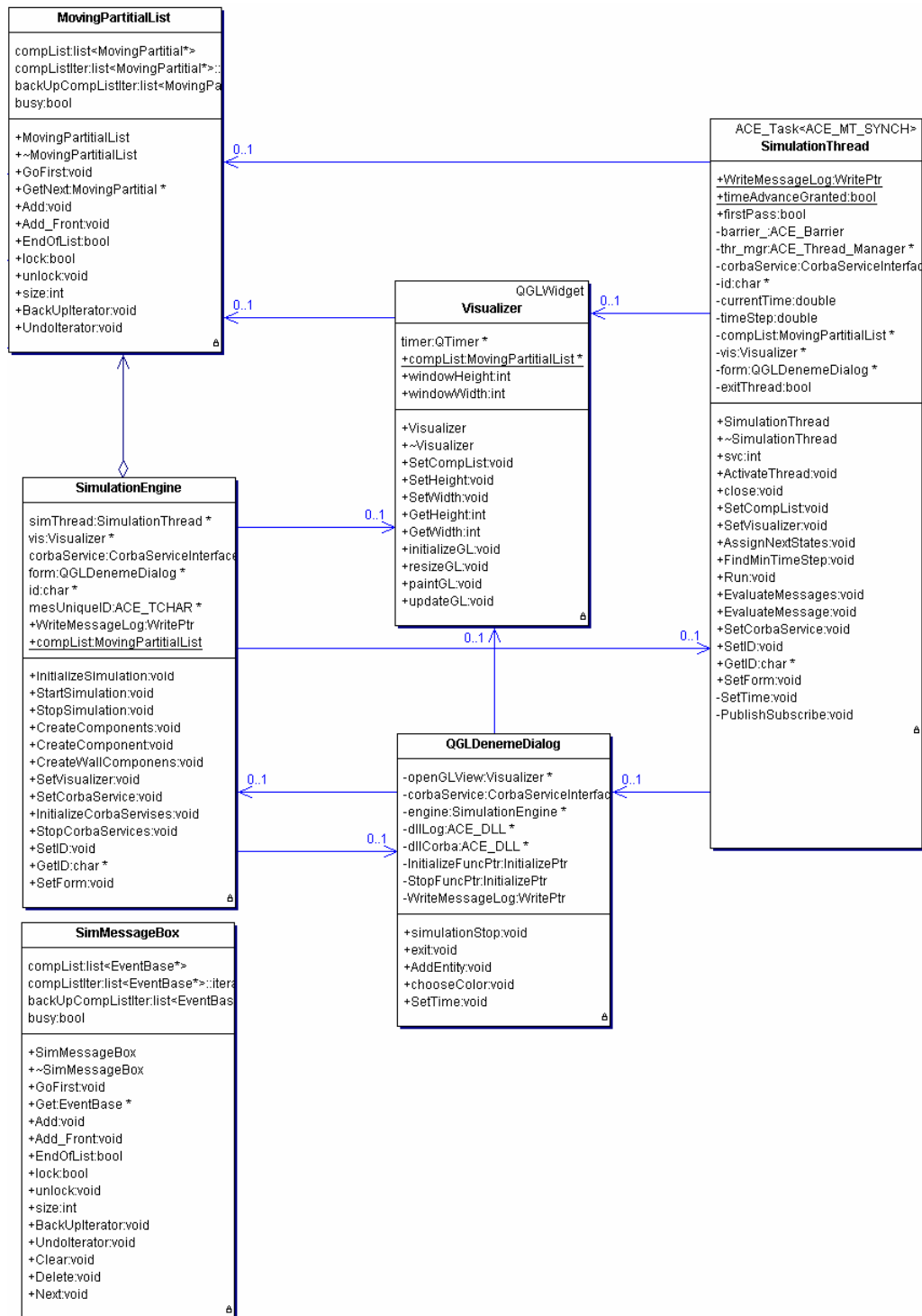
3.2.4.3 Sistem Tasarımı

Benzetim sistemi temelde 5 ana sınıf içermektedir. Sistem tasarımı Şekil 3.15’de ve Şekil 3.15’de verilmiştir. Dağıtık Altyapılar ve Mesaj Kayıt Sistemine ait nesneler kütüphaneler tarafından sağlandığından benzetim sistemi şemasında ele alınmamıştır.

- **MovingPartital:** Hareket eden bileşenler için temel sınıftır. Aktif olarak dairesel hareketli nesneler oluşturulabilmektedir. Benzetim sisteminde kullanılan hareketli bileşenler Dairesel Hareketli Parçalar ve Duvarlardır. Kapalı ortamı oluşturan ve dikdörtgen şeklinde olan duvarlar da hızları 0 olan birer hareketli parça olarak tasarlanmıştır. Sistemde bulunan bütün hareketli parçalar MovingPartitalList içinde tutulur.
- **MovingPartitalList:** Hareketli bileşenlerin listesini tutan ve SimulationEngine'nin güncellemeleri yapmak için ve Visualizer'in bileşen durumlarını görselleştirmek için kullandıkları listedir.
- **SimulationEngine:** Benzetime dahil edilmiş bileşenlerin durumlarını, kendilerine gelen mesajlara göre ayarlayan nesnedir.
- **SimulationThread:** Benzetim döngüsünün bulunduğu ana sınıftır. Bu sınıf içinde benzetim federasyona dahil olma, zaman ilerletme, federasyondan gelen mesajların bileşenlere iletimi ve koşum kayıtlarının alınması görevlerini yapan nesnedir.
- **QGLDenemeDialog:** QT Dialog formudur. Bütün arayüz tanımlamalarını içerir.
- **MessageBox:** Federasyondan benzetime gelen mesajların bulunduğu listedir.



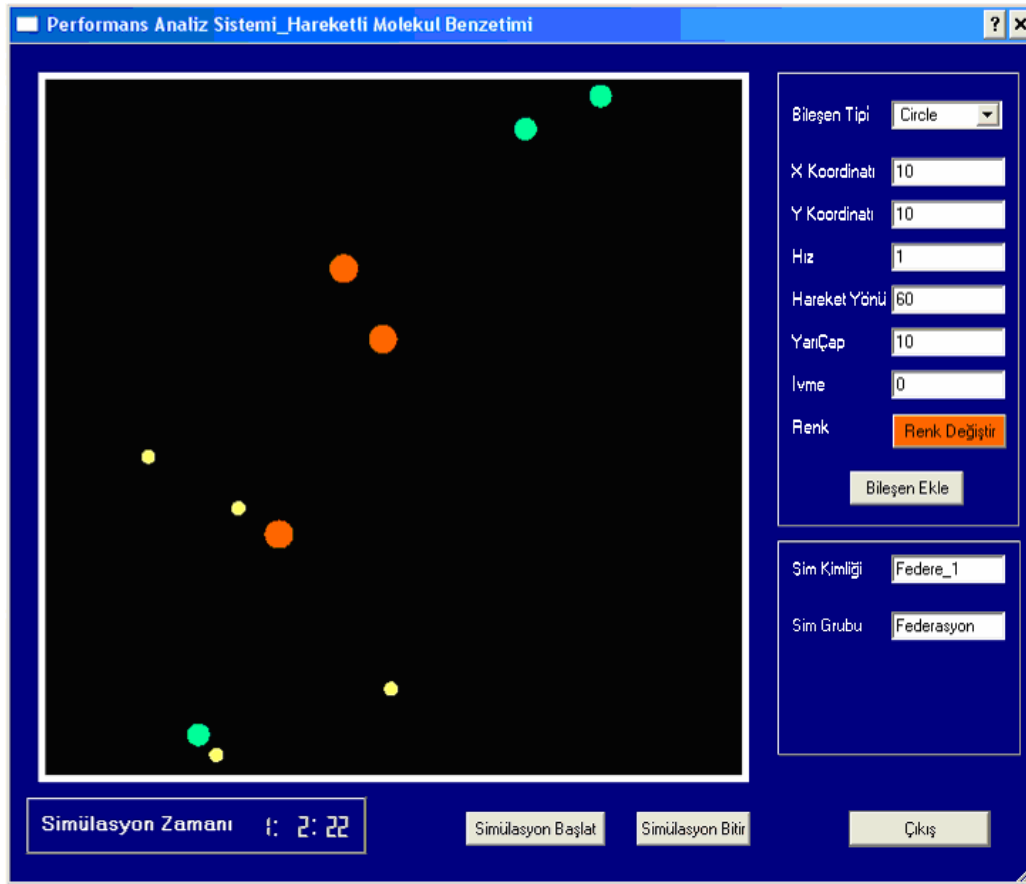
Şekil 3.14 Molekül Benzetimi Hareketli Bileşen Yapısı.



Şekil 3.15 Molekül Benzetimi Sınıf Şeması.

3.2.4.4 Kullanımı

Tasarlanan benzetim sistemi ve arayüz sayesinde benzetim sisteminde pratik değişiklikler yapılabilmekte ve benzetim performansının bu değişikliklerdeki değişimi gözlenebilmektedir. Hazırlanan arayüz sayesinde benzetim sistemi bir federasyona katılıp, bu federasyonda kendi hareketli molekül nesnelerine ait durum güncellemelerini federasyona yayabilir. Koşum sırasında kolaylıkla sisteme yeni moleküller eklenerek sistem yoğunluğu değiştirilebilir. Benzetime dahil edilecek yeni moleküllere ait öznitelik bilgileri Şekil 3.16'de verilen arayüzdeki tanımlama ekranı aracılığıyla girilebilir.



Şekil 3.16 Molekül Benzetimi Arayüzü.

Benzetim koşumunun başlatılabilmesi için performans analizi sistemi bünyesinde çalıştırılması gereken mesaj kayıt sistemi ve dağıtık altyapı servislerine ait yazılımlar çalıştırılmalıdır. Aksi taktide program hata vererek kapanacaktır. Mesaj Kayıt Sistemi ve kullanılacak dağıtık altyapı yazılımı sunucu yazılımı çalıştırıldıktan

sonra Benzetim sistemi arayüzünde Sim Kimliği bölümüne federe ismi olarak bir değer ve Sim Grubu bölümüne katılacak federasyon ismi değeri yazılmalıdır. Bu iki değer yazılmadığı sürece benzetim çalıştırılmayacaktır. Federe ismi ve federasyon ismi ilgili bölümlere yazıldıktan sonra Benzetim Başlat düğmesi kullanılarak benzetim başlatılır. Dağıtık altyapı ayarlamaları yapılır, eğer federasyon henüz tanımlanmamışsa federasyon tanımlaması yapılır ve federenin federasyona kaydı yaptırılır. Benzetim koşumunun başladığı, zamanın ilerlemesinden anlaşılabılır. Benzetim koşumu gerçekleştirildikten sonra Benzetim Arayüzünde sağ taraftaki panelde benzetime eklenecek bileşen özellikleri ayarlanarak Bileşen Ekle düğmesiyle benzetime hareketli moleküller eklenmesi sağlanır. Oluşturulacak hareketli bileşenlere ait başlama koordinatları(pixel), hızı (pixel/sn), hareket yönü(°), bileşen yarıçapı (pixel), ivme(pixel/sn²) ve renk bilgileri değiştirilebilir ve yeni tip bileşenler benzetime eklenebilir. Benzetim ağ ortamında her makinede istenilen sayıda çalıştırılabilir.

3.2.4.5 Benzetim Sırasında Koşum Kayıtlarının Alınması

Performans analizine yönelik koşum kayıtları benzetim sistemi tarafından alınır. Koşum kayıtları Mesaj Kayıt Sistemi İstemci Kütüphanesi kullanılarak sağlanır. Kayıt sisteminin çağırılma yapısı

WriteMessageLog(MESSAGETAG,MESUNIQUEID,MESSIZE)

Şeklinde dir. Mesaj Kayıt Sistemi bölümünde alınan kayıtlar hakkında ayrıntılı bilgi verilecektir.

3.2.5 Kayıt Sistemi

Performans analizinin doğru bir şekilde yapılabilmesi için metrik kayıtları hassas ve doğru bir şekilde tutulmalıdır. Bu amaçla Kayıt Sistemi geliştirilmiştir. Performans metriklerine ait tutulacak bütün kayıtlar ağ ortamında hassas bir senkron saat ile kayıt sistemi tarafından tutulur. Yapılan araştırmalarda performans karşılaştırma çalışmalarında tutulan kayıtlara ait zaman senkronizasyonun GPS saati ve SNTP sunucular aracılığıyla sağlandığı görülmüştür. Bu sistemlerin kullanılan analiz yazılımından çok test ortamının düzenlenmesiyle kullanıldıkları göz önüne bulundurulduklarında mevcut yazılımın başka bir ortamda analiz çalışmalarını

yürütebilmesi artı iş yükü getireceğinden geliştirilen kayıt sisteminin dahilinde senkron bir saat tutulması tercih edilmiştir.

Kayıt bilgileri yerel disklerde tanımlı dosyalarda tutulur. Kayıt bilgilerinin yerel dosyalarda tutulması kayıt sisteminin ağ ortamındaki yoğunluğa etkisinin azaltılmasını sağlamak içindir. Kayıt sistemi yazılımı Microsoft Visual Studio 2005 ortamında CPP programlama dilinde ACE, TAO kütüphaneleri kullanılarak yazılmıştır.

3.2.5.1 Beklenen Yetenekler

Kayıt sisteminden beklenen yetenekler şu şekilde sıralanabilir.

- Dağıtık çalışabilmeli
- Kayıt bilgileri senkron olmalı
- Her türlü uygulamadan çağrılabilmesi
- Sistem performansını düşürmemeli
- İstenilen formatta kayıt alınabilmeli

3.2.5.2 Yazılım Tasarımı

Kayıt sistemi zaman güncellemesi için yönetimi sağlayan bir sunucu ve güncellenen zamana göre kayıt bilgilerini tutan istemcilerden oluşmaktadır. Kayıt sisteminin yazılması için CORBA ara katmanının kolay kullanımı için bir kütüphane hazırlanmış ve CORBA özelinde yapılacak işlerin bu kütüphane aracılığıyla yapılması sağlanmıştır. Bu kütüphaneye ait API arayüzü Şekil 3.17’de verildiği gibidir. API açıklamaları EK-3’te verilmiştir.

Kayıt Sistemi Sunucu yazılımı CORBA olay servisleri kullanılarak hazırlanmış ve yüksek çözünürlüklü bir sayaca bağlanarak zaman güncellemelerinin yapılması sağlanmıştır.

Kayıt sisteminde 4 ana sınıf mevcuttur. Bunlar MessageLogger, MessageConsumer, MessageParser ve TimerThread sınıflarıdır.

MessageLogger: Kayıt Sistemi kayıt yönetim merkezi görevi görür. Kayıt bilgilerinin giriş noktasıdır. Kayıt için gelen mesajlara TimerThread tarafından

güncellenen en son zaman etiketi atanır. Kaydedilecek mesajlar kaydedilmek üzere MessageConsumer nesnesine gönderilir.

```

1
2 #define NAMING_SERVICE_NAME "NameService"
3 #define COS_EVENT_SERVICE_NAME "CosEventServiceForRam"
4 #define TIME_SERVICE_NAME "TimeServiceForRam"
5 #define EVENT_MANAGER_SERVICE_NAME "EventManagerServiceForRam"
6
7 class CorbaServiceInterface{
8
9 public:
10     virtual ~CorbaServiceInterface(){};
11     virtual void Initialize( int argc, ACE_TCHAR* argv[] ) = 0;
12     virtual void StartWorkerThreads() = 0;
13     virtual void FindNameService() = 0;
14     virtual int GetRootPoa() = 0;
15     virtual void ActivatePoaManager() = 0;
16     virtual void DeclareNewCorbaEnv() = 0;
17     virtual void ResolveEventChannel() = 0;
18     virtual void ResolveEventChannelByName( char* _name ) = 0;
19     virtual CORBA::Object_var ResolveService( char* _name ) = 0;
20     virtual void ExportRootPoa( char* filename ) = 0;
21     virtual bool ConnectServer( char* name ) = 0;
22     virtual void ConnectAsConsumer( char *consumerId,char* _eventChannelName = COS_EVENT_SERVICE_NAME ) = 0;
23     virtual void ConnectAsSupplier( char* _name = "EventSupplier", char* _eventChannelName = COS_EVENT_MANAGER_SERVICE_NAME ) = 0;
24     virtual void DisconnectAsSupplier( char* _eventChannelName ) = 0;
25     virtual void SendEvent( char* _name, EventBase *eventMes,char * _mesunique ) = 0;
26     virtual void Run() = 0;
27     virtual void SetTimer( PortableServer::Servant timer_servant ) = 0;
28     virtual void Stop() = 0;
29     virtual void RunInThread() = 0;
30     virtual void StopOrbThread() = 0;
31     virtual void ConnectTimeService(char* _name =TIME_SERVICE_NAME) = 0;
32     virtual void DestroyEventChannel( char* _name ) = 0;
33     virtual double StartSimulationTime( char* _id ) = 0;
34     virtual double StopSimulationTime( char* entity ) = 0;
35     virtual void AdvanceTime( char* id, double _time ) = 0;
36     virtual void SetTimeAdvanceGrantCallback( void f( bool ) ) = 0;
37     virtual void Subscribe( char *id,char * _eventName, EventCallbackPtr ) = 0;
38     virtual void JoinEventManagerService(char *id) = 0;
39     virtual void DisconnectEventManagerService(char *id) = 0;
40     virtual void ConnectTimeManagerService(char *federationID, char *id , void ( *ptr )( const
41     virtual void DisconnectTimeManagerService(char *federationID, char *id) = 0;
42     virtual void CreateTimeService(char * _name = TIME_SERVICE_NAME) = 0;
43     virtual void BindService( char* _name, CORBA::Object_var p_servant ) = 0;
44     virtual void ActivateObject( PortableServer::Servant p_servant, char* ) = 0;
45     virtual void BindToEventChannelByName( char* _name ) = 0;
46     virtual void BindToEventChannel() = 0;
47     virtual void DeActivateObject( PortableServer::ObjectId_var oid, char* name ) = 0;
48     virtual void StartFederationInterOperate() = 0;
49     virtual void CreateTimeManagerService(char * _name) = 0;
50 };
51 #endif

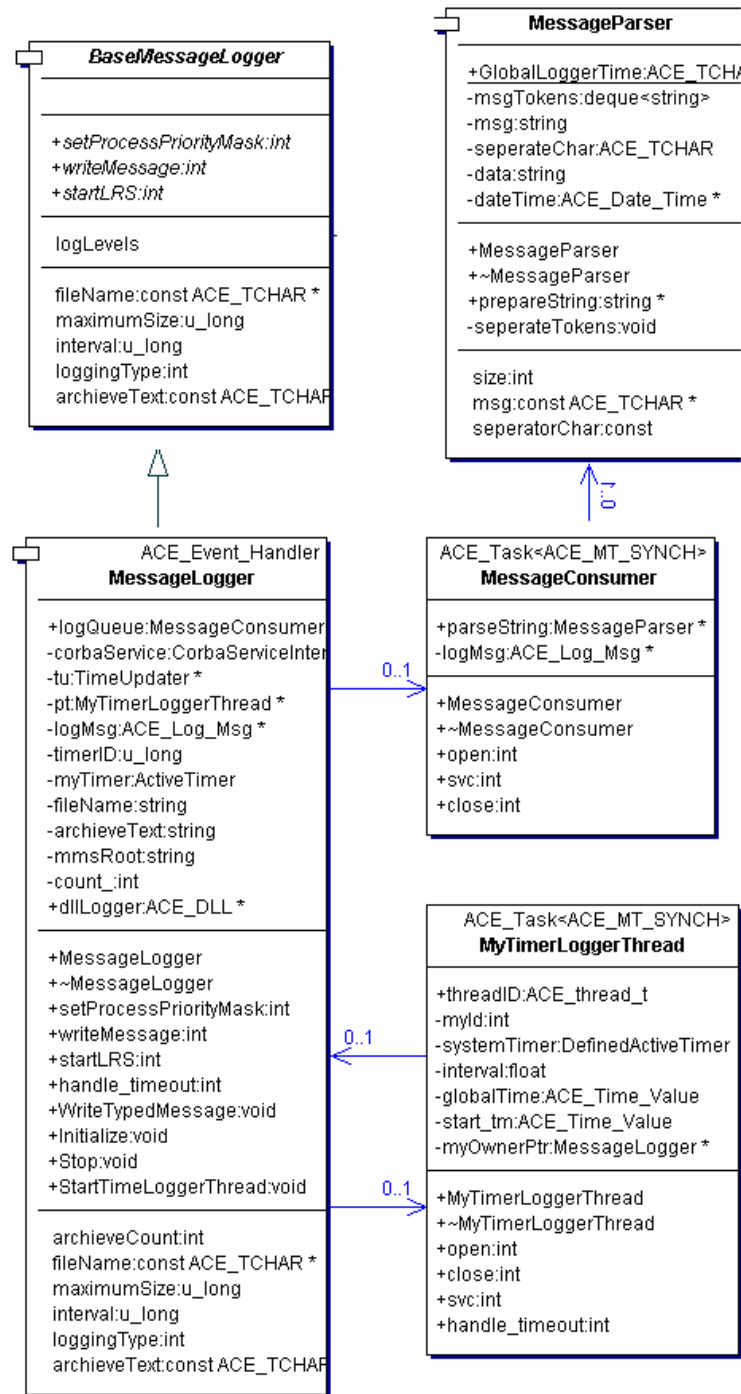
```

Şekil 3.17 CORBA Kütüphanesi.

MessageConsumer: Dosyaya yazdırılacak kayıtların hazırlanması için mesajları MessageParser nesnesine gönderir. Hazırlanan mesajları dosyaya kaydeder.

MessageParser: Mesaj kalıplarını gelen formata göre düzenler basit karakter dizileri oluşturur.

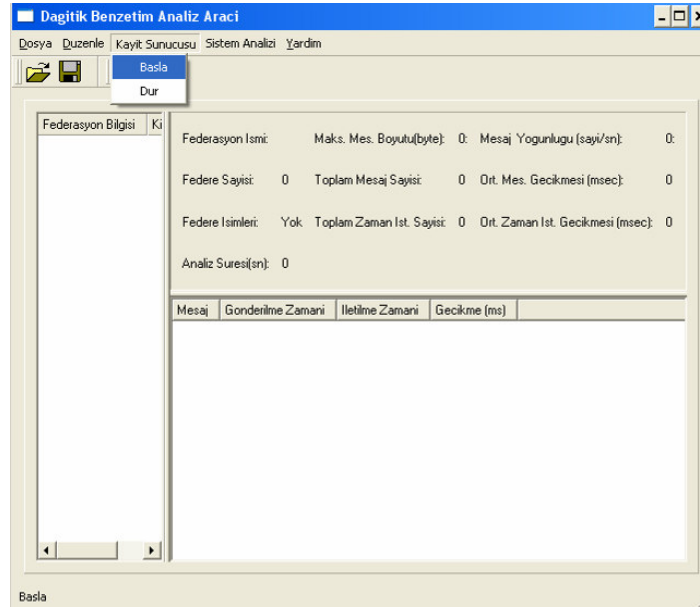
TimerThread: Sunucudan gelen en son zaman bilgilerini kontrol eder ve Kayıt Sistemi İstemi zamanını günceller.



Şekil 3.18 Kayıt Merkezi Sistem Yapısı.

3.2.5.3 Kullanımı

Kayıt sistemi sunucu yazılımı ayrıca analiz sistemi olarak da kullanılacak Analiz Formu aracılığıyla çalıştırılır. Analiz formu ve Kayıt Sunucusu başlatma ve durdurma menüsü Şekil 3.19’da görülmektedir.



Şekil 3.19 Kayıt Sunucusu Başlatma ve Durdurma.

Kayıt Sunucusu çalıştırıldıktan sonra her hangi bir benzetim koşumu gerçekleştirildiğinde bu benzetimin kaydı kayıt kütüphanesi aracılığıyla alınabilmektedir. Kayıt dosyaları kayıt istemcinin adında ilgili dosyanın altında oluşturulmaktadır. Kayıt İstemi Kütüphanesi kullanıcı arayüzü basit bir API den oluşmaktadır. API de bulunan fonksiyonlar,

void Initialize(char *id)

id Kayıt İstemcisi ismi

Kayıt sistemi başlangıç işlemlerini yürütür. Kayıt sistemi başlatılması için ilk çağırılması gereken fonksiyondur.

void WriteTypedMessage(char *mesType, char *msg, long size)

mesType performans sitemine özgü kayıt tipi. Kayıt tipleri MESSEND, MESRECEIVED, TIMEADV, TIMEADVGRANT şeklinde sıralanmaktadır.

msg mesaj içeriği

size mesaj boyutu

Kayıt alınması istenilen her noktada çağrılacak kayıt fonksiyonudur. Performans analizi sadece ilgili kayıt tipleri için kayıt alabilir.

void Stop(char *id)

İd Kayıt İstemci ismi

Kayıt istemcinin sunucudan bağlantısını keser.

3.2.6 Analiz Aracı

Kayıt sistemi aracılığıyla tutulan yerel diklerdeki dosyalar geliştirilen Analiz Aracı ile incelenebilir ve altyapılara ait karakterizasyon grafikleri elde edilebilir. Analiz Aracı arayüzü Şekil 3.19'daki gibidir. Analiz Aracı aracılığıyla bir federasyonda bulunan bütün federelere ait kayıt bilgileri aynı anda açılıp federasyon bazında inceleme yapılabilir. Ayrıca federe kayıt dosyaları ardışık olarak analiz sistemine yüklenip incelenebilir. Analiz Aracı ile,

- Zaman İlerleme Grafiği
- Mesaj Gecikmesi(msec)-Sim Zamanı (sn)
- Zaman Gecikmesi(msec)-Sim Zamanı (sn)
- Mesaj Sayısı-Sim Zamanı (sn)
- Mesaj Yagunlugu(sayi/sn)-Mesaj Gecikmesi (msec)
- Mesaj Boyutu(byte)-Sim Zamanı (sn)
- Mesaj Boyutu(byte)-Mesaj Gecikmesi (msec)
- Federe Sayısı-Sim Zamanı (sn)

Grafikleri elde edilebilmektedir. Analiz aracı Microsoft Visual Studio 2005 ortamında CPP diliyle ACE ve QT kütüphaneleri kullanılarak hazırlanmıştır. Grafiklerin çizdirilmesi için QPLOT3D isimli açık kaynak kodlu bir QT eklentisi kullanılmıştır.

3.2.6.1 Beklenen Yetenekler

Analiz aracından beklenen analiz için alınan analiz kayıtlarından ihtiyaç duyulan grafiklerin elde edilebilmesini sağlamasıdır.

3.2.6.2 Sistem Tasarımı

Analiz Sistemi ShowFileTask, FormUpdate, PaintFormMessageBlock, FederateInfo, FederationInfo, FedMessage, FedTimeAdv, SurfacePlot sınıflarından oluşmaktadır.

ShowFileTask : Sisteme yüklenecek dosyaların yüklenmesini sağlar.

FormUpdate: Form bilgilerinin güncellenmesini sağlayan nesnedir.

PaintFormMessageBlock: Yeni yüklenen dosya bilgilerine göre analiz veritabanının güncellenmesini sağlar.

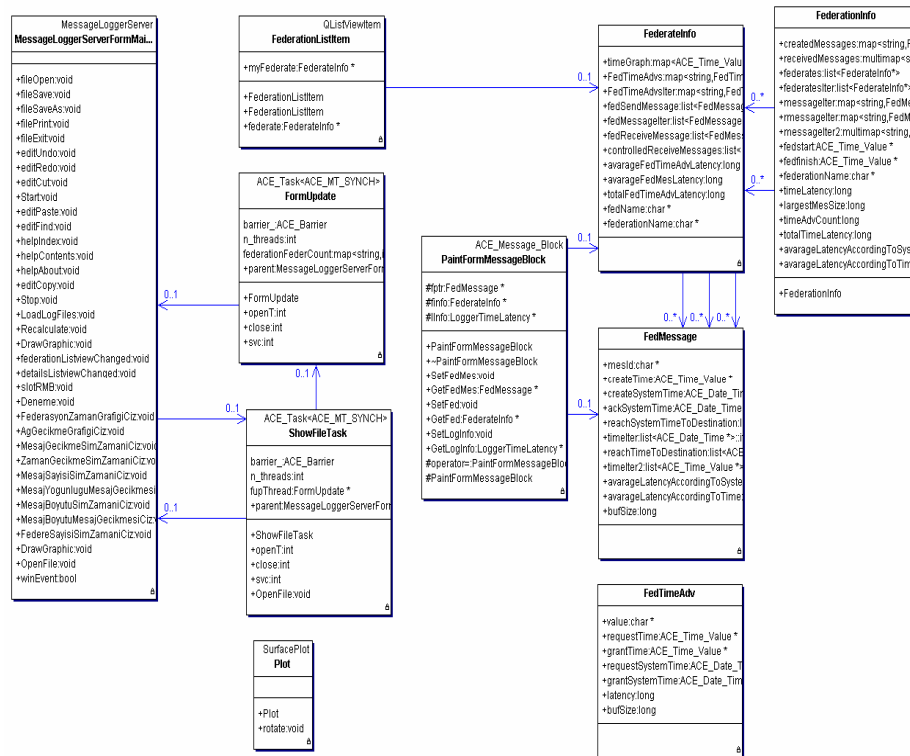
FederateInfo: Sisteme yüklenen federeye ait bilgilerin tutulduğu nesnedir.

FederationInfo: Sisteme yüklenen federasyon bilgilerini tutan nesnedir.

FedMessage: Federe mesaj gönderilme ve alınma kayıt bilgilerini tutar.

FedTimeAdv: Federe zaman güncelleme kayıt bilgilerini tutar.

SurfacePlot: Grafik çizimleri için kullanılan nesnedir.



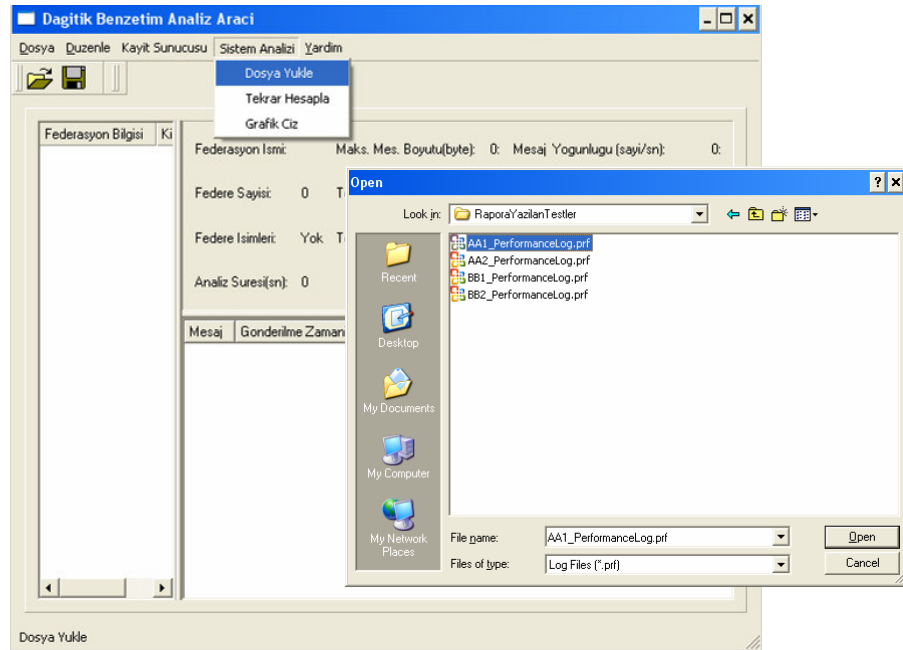
Şekil 3.20 Analiz Sistemi Yapısı.

3.2.6.3 Kullanımı

Analiz Sistemi ile dosyaların yüklenmesi ve yüklenen dosyalardaki bilgilere ait değişik grafikleri alınması işleri yapılabilmektedir.

Yeni Kayıt Dosyası Yükleme:

Yeni kayıt dosyası yüklemek için Sistem Analizi menüsünden Dosya Yükle sekmesi seçilerek dosya açma formu açılır. Açılan formdan ilgili kayıt dosyası bulunup Şekil 3.21’deki gibi Open tuşuna basılır. İlgili dosyanın sisteme yüklenmesi tamamlandıktan sonra ilgili federasyon bilgisi ekranda güncellenerek listelenir. Dosya yükleme işlemi ardıl olarak yapılabildiği gibi aynı anda bütün dosyaların seçilerek yüklenmesi sağlanabilir.



Şekil 3.21 Analiz Sistemi Dosya Yükleme.

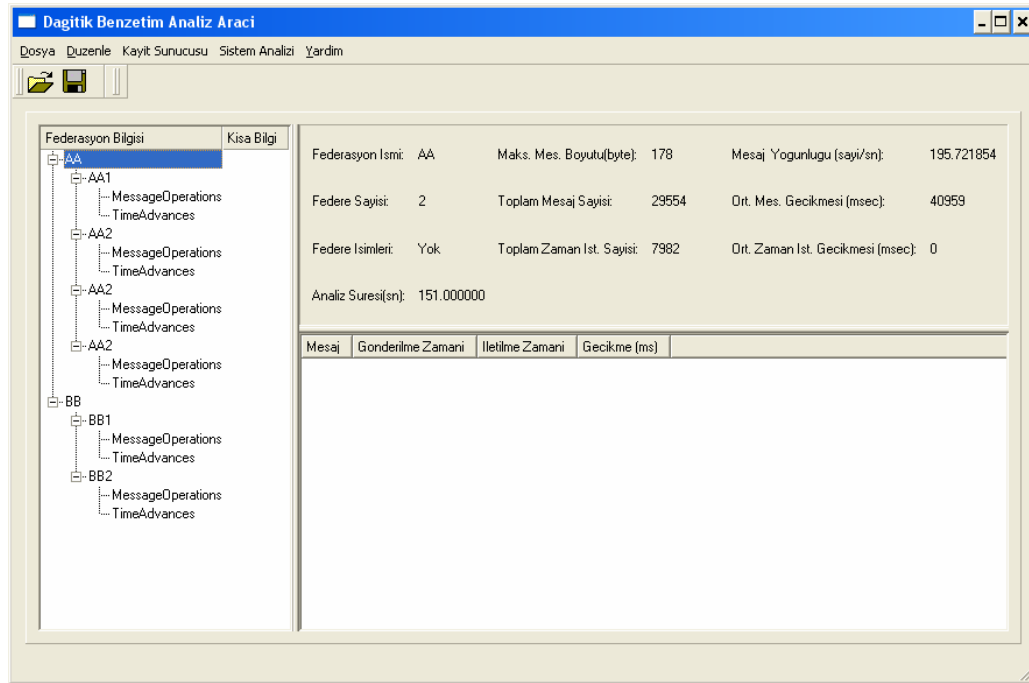
Federasyona Ait Bilgilerin Alınması:

Şekil 3.22’de yüklenen 2 federasyona ait bilgiler görülmektedir. Sisteme yüklenen federasyona ait kısa bilgileri ilgili federasyonun üzerine sol tuşla tıklayarak görülebilir. Şekil 3.22’de AA federasyonuna ait ortalama bilgiler görülmekte.

Ekran Bilgilerine ait açıklamalar şu şekildedir:

Federasyon İsmi: Seçili federasyon ismi

Federe Sayısı: Federasyonda kayıtlı federe sayısı



Şekil 3.22 Analiz Sisteminde Listelen Federasyon Bilgileri.

Analiz Süresi: Analiz işlemi süresi

Mak. Mes. Boyutu: Federasyonda gönderilen en büyük mesajın boyutu

Toplam mesaj sayısı: Federasyonda gönderilen toplam mesaj sayısı

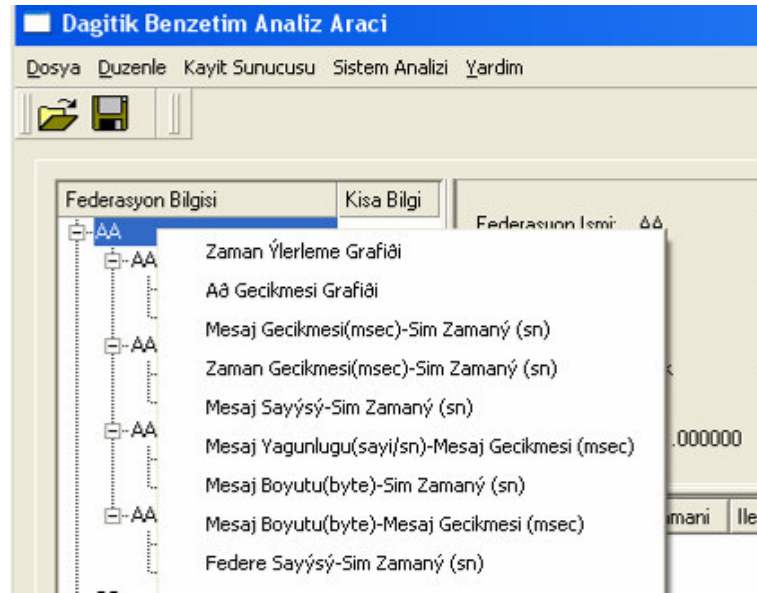
Toplam Zaman İst. Sayısı: Federasyonda gönderilen toplam zaman isteği sayısı

Mesaj Yoğunluğu: Birim zamanda gönderilen mesaj sayısı

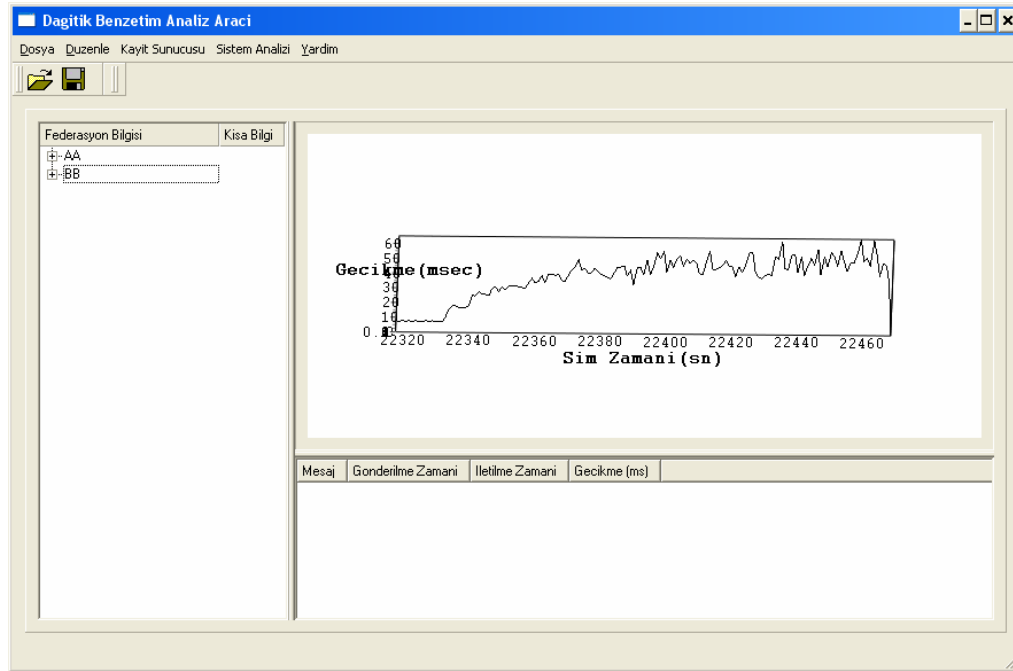
Ort. Mesaj Gecikmesi: Analiz boyunca ortalama mesaj gecikmesi

Ort. Zaman İst. Gecikmesi: Analiz boyunca ortalama zaman isteği gecikmesi.

Federasyona ait çizdirilebilecek grafikler listeden ilgili federasyona sağ tıklanarak görülebilir. Şu anda sistemde tanımlı 8 adet grafik mevcuttur. Sisteme yeni grafikler eklenilebilmesi mümkündür. Mevcut seçilebilecek grafikler giriş bölümünde verilmiştir. İlgili liste Şekil 3.23'te görülebilir. Seçili grafik ekranda federe bilgileri bulunan panel üzerine çizdirilir. Örnek olarak çizdirilen bir grafik Şekil 3.24'de görülebilir.



Şekil 3.23 Federasyona Ait Çizdirilebilecek Grafikler.



Şekil 3.24 Federasyon Gecikme-Benzetim Zamanı Grafiği.

4 DENEYSEL ÇALIŞMALAR

Tez kapsamında yapılması planlanan analiz sistemine ait beklenen yetenekler geliştirilen sistemler bölümünde verilmiştir. Bu hedefler, performans incelemesi için değişen mesaj yoğunluklarına, federe sayılarına ve mesaj boyutlarına göre gecikme analizlerinin yapılabilmesidir. Hedeflerin gerçekleştirilebilirliğini göstermek amacıyla yapılan 3 deney bu bölümde açıklanacaktır. Birinci deney iki farklı altyapı için değişen mesaj yoğunluklarında oluşan gecikmeleri incelemek, ikinci deney iki farklı altyapı üzerinde çalışan benzetime yeni federeler eklenmesiyle oluşacak gecikmeleri gözlemlemek, üçüncü deney iki farklı altyapıyla değişen mesaj boyutlarına göre gecikmeleri analiz etmek için tasarlanmıştır.

4.1 Deney1 - 2 Altyapının Mesaj Sayısı X Gecikme Analizi

Bu deneyde, DMSO RTI ve RAM DBA altyapılarının, 2 Federeden oluşan bir benzetimde mesaj yoğunluğuna göre performans ölçümlerinin alınması ve analiz edilebilmesi amaçlanmıştır.

Deneyler için Performans Analiz Sistemi kapsamında geliştirilen Hareketli Molekül Benzetimi kullanılmıştır. Deneyler boyunca benzetim sistemine ön tanımlı bileşenler eklenmiştir. Bileşen özelliklerinde herhangi bir değişiklik yapılmamıştır. Bu kapsaman benzetim koşumu boyunca eklenen bileşenlere ait temel özellikler Şekil 4.1'deki gibidir.

Özellik	Değer
Bileşen Tipi	Circle
X Koordinatı	10 pixel
Y Koordinatı	10 pixel
Hız	1 pixel / sn
Hareket Yönü	30°

Yarıçap	5 pixel
İvme	0
Renk	Sarı

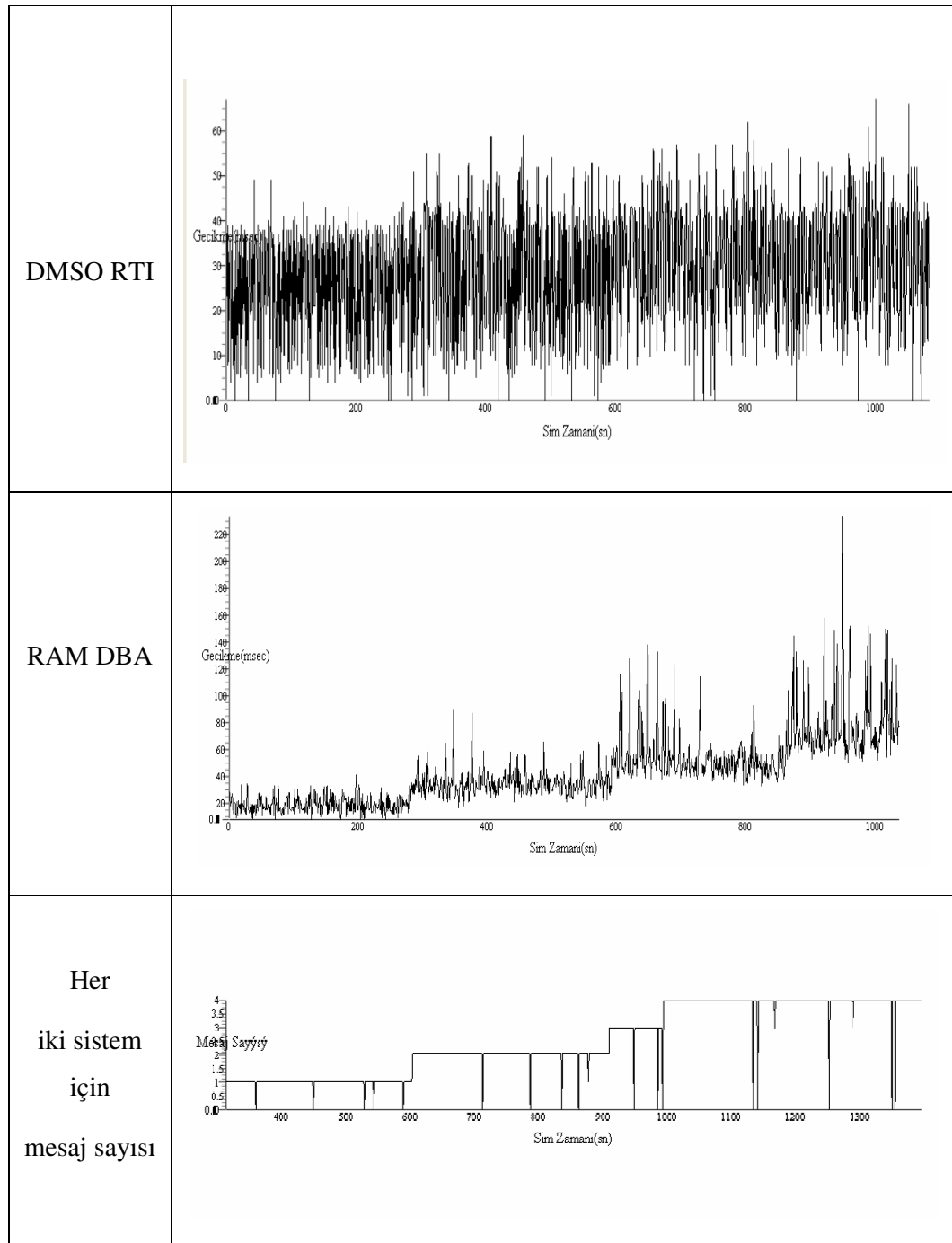
Şekil 4.1 Deneylerde Kullanılan Bileşen Özellikleri Tablosu

Deney benzetim zamanı olarak 1400 sn boyunca çalıştırılmıştır. Sisteme benzetim başlangıcından sonra her 300. sn içinde bir bileşen eklenmiştir. Deney için benzetim koşumları önce DMSO RTI için sonra RAM DBA için tekrar edilmiştir.

	DMSO RTI 1.3	RAM DBA
Federe Sayısı	2	2
Benzetim Zamanı	1400 sn	1400 sn
Deney Süresi	40 sn	72 sn
Toplam Mesaj Sayısı	2723	2415
Mesaj Boyutu	188 byte	188 byte

Şekil 4.2 DMSO RTI 1.3, RAM DBA Karşılaştırma Tablosu

Benzetim koşumu sonrasında yapılan analizde kısaca Şekil 4.2’de görülen sonuçlar çıkmıştır. 1400 saniyelik benzetim koşumunu DMSO RTI 1.3 altyapısı 40 saniyede sağlamıştır. RAM DBA altyapısı daha düşük bir performansla 1400 saniyelik koşumu 72 sn de tamamlamıştır. Burada zaman ilerletme performansı RAM DBA altyapısı için çok düşüktür, bu da henüz altyapının tam olgunluğa ulaşmamasından kaynaklanmaktadır. DMSO RTI için iletimi gerçekleşen toplam mesaj sayısı da zaman artırım isteğine paralel olarak RAM DBA’na göre daha fazladır. Deneme sürecinde oluşan gecikme grafikler Şekil 4.3’de görüldüğü gibidir.



Şekil 4.3 DMSO RTI 1.3, RAM DBA Gecikme Karşılaştırma Grafikleri

Gecikme grafiklerinde görüldüğü gibi DMSO RTI sisteminde başlangıç itibariyle mesaj gecikmesi yaklaşık 10 ms iken, mesaj yoğunluğunun artmasıyla gecikme ortalama 36 mili saniyelere doğru tutarlı bir oranda yükselmekte. Çalışma sırasında gecikmelerin sürekli iniş çıkışlar yaşamasının sebebi DMSO RTI'nın

performansı artırmak için yük dengelemek amacıyla koyduğu kontrol mekanizması olabilir. Gecikme başlangıç safhalarında 25 ms olduğu gibi koşumun son safhalarında da yine 25 ms gecikmelerle iletim sağlanabilmekte. Buna karşın RAM DBA altyapısında mesaj yoğunluğunun her artırılmasında gecikmelerde kritik yükselişler meydana gelmekte. Öyle ki, 20 ms ler civarında oluşan başlangıç durumu mesaj gecikmelerine karşılık, artan mesaj yoğunluğuyla birlikte 80-100 ms civarında gecikmeler gözlemlenmektedir. RAM DBA altyapısının yoğun mesaj trafiğinde performans eksiklikleri olduğu görülmekte.

4.2 Deney2 - Altyapının Federe Sayısı X Gecikme Analizi

Bu deneyde, DMSO RTI altyapısının, hareketli molekül benzetiminde federe sayılarının değişiminde göre performans ölçümlerinin alınması ve analiz edilebilmesi amaçlanmıştır.

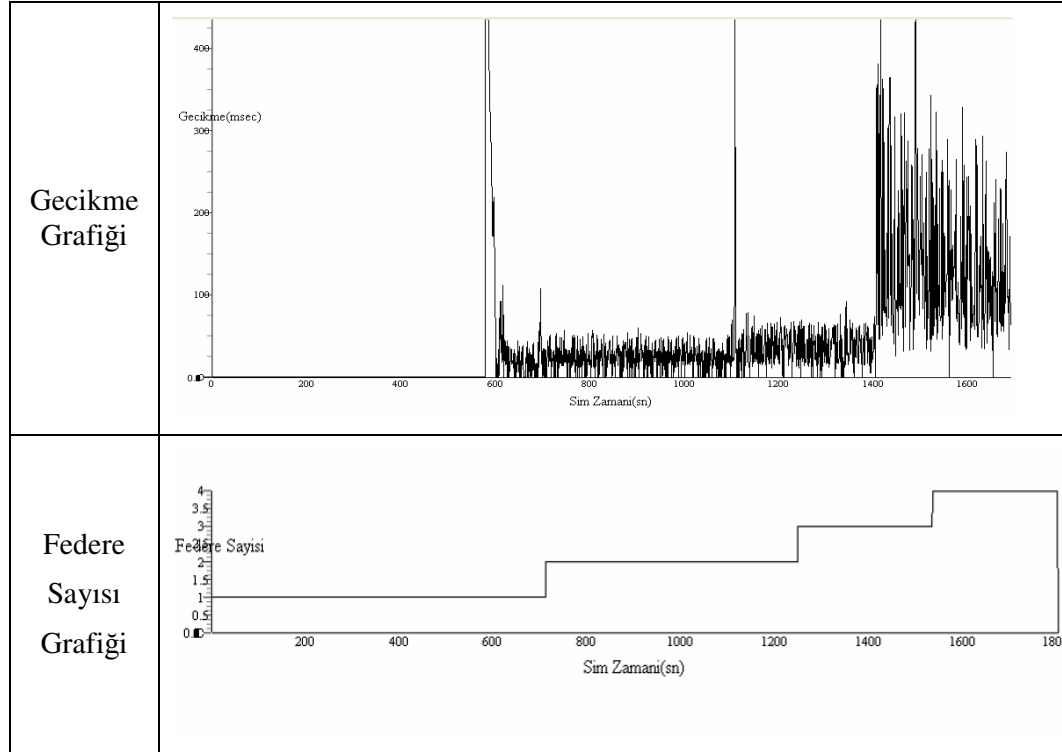
Deney benzetim zamanı olarak 1800 sn boyunca çalıştırılmıştır. Birinci federe çalıştırıldığında diğer federeleri başlatmadan 5 bileşen oluşturulmuştur. Oluşturulan bileşenler Şekil 4.1'deki gibi temel özelliklerinde değişiklik yapılmadan sisteme dahil edilmişlerdir. Böylece diğer federeler başladığında hepsi için aynı mesaj yoğunluğunda bir akış gerçekleşecektir. Sisteme benzetim 600. saniyeden sonra her 300. sn içinde yeni bir federe katılmıştır.

Federe sayılarının artırılmasıyla sistem çalışması yavaşlamış sonuç deney süresinin toplamda 156 saniyelere çıkmasına neden olmuştur. Koşum sonucunda elde edilen son durum bilgisi Şekil 4.4'de görüldüğü gibidir.

Federe Sayısı	4
Benzetim Zamanı	1800 sn
Deney Süresi	156 sn
Mesaj Sayısı	8143
Ort. Mesaj Gecikmesi	73 msn
Mesaj boyutu	174 byte

Şekil 4.4 Deney2 DMSO RTI Koşum Raporu Tablosu

Deney sonucunda elde edilen analiz grafikleri Şekil 4.6'daki gibidir. Şekilde görüldüğü gibi Federe sayısının dörde çıkması mesaj gecikmesinin 100 milisaniyelerin üzerine çıkmasına neden olmuştur. Yeni federelerin sisteme dahil olma çalışmaları nedeniyle o noktalarda gecikmeler çok yüksek çıkmaktadır.



Şekil 4.5 Deney2 Federe Sayısına Bağlı Gecikme Grafikleri

4.3 Deney3 - Altyapının Mesaj Boyutu X Gecikme Analizi

Bu deneyde, DMSO RTI altyapısının, hareketli molekül benzetiminde değişen mesaj boyutlarında oluşturduğu performans ölçümlerinin alınması ve analiz edilebilmesi amaçlanmıştır.

Deney her seferinde benzetim zamanı olarak 2410 sn olan 9 koşum gerçekleştirilmiştir. Her benzetim 2 federeden oluşmakta ve 2. federe sisteme 300. saniyede katılmaktadır. Yukarıdaki prosedür bütün koşumlar için aynı olmak kaydıyla her koşumda ilk federe 300. saniyeye gelmeden 1 bileşen oluşturmaktadır.

Oluşturulacak bu bileşenin yarıçapı;

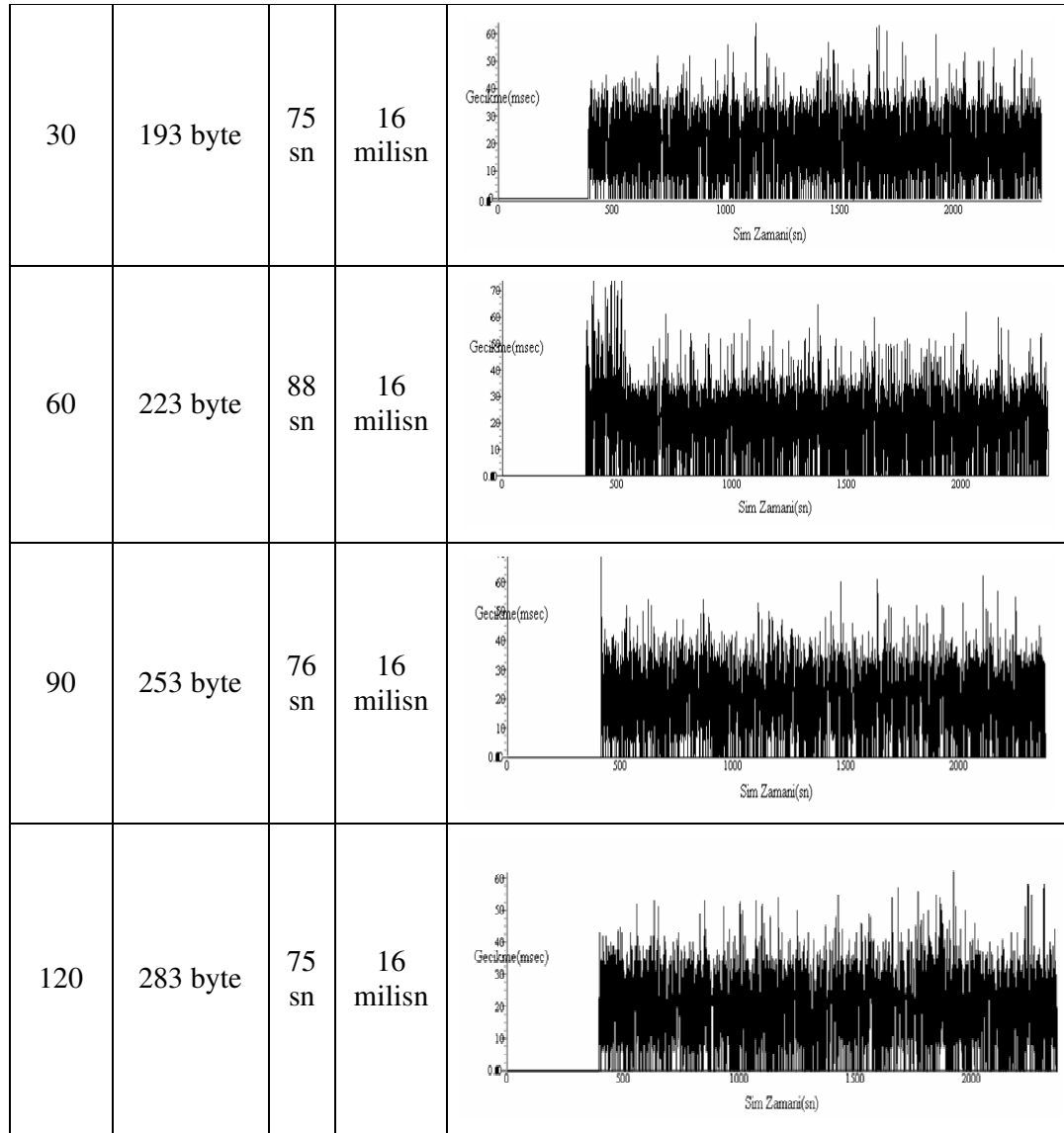
1. Koşum - 5
2. Koşum - 10
3. Koşum - 15
4. Koşum - 20
5. Koşum - 25
6. Koşum - 30
7. Koşum - 60
8. Koşum - 90
9. Koşum -120

olarak atanmış ve benzetime gönderilecek mesaj boyutu bu yarıçapın bir çarpanı oranında artırılmıştır. Bileşene ait diğer özellikler Şekil 4.1'deki ile aynı değerlerdedir. 9 Koşum tekrar edildikten sonra bütün kayıt bilgileri kayıt sisteminde analiz edilmiştir.

Gerçekleştirilen 9 koşum için elde edilen özet veriler Şekil 4.6'da ve Şekil 4.7'de verilmiştir. Tablodaki verilerden ve grafiklerden anlaşılabileceği gibi mesaj boyutunda 50- 60 byte lık artışlar olmasına rağmen 2 federe için DMS RTI gecikmelerinde ve çalışma performansında bir değişme olmamıştır. Oluşan küçük çaptaki değişiklikler sistem performansından kaynaklanan değişiklikler olarak yorumlanabilir.

Bileşen	Gönderilen Mesaj Boyutu	Deney Süresi	Ortalama Gecikme	Grafik
5	168 byte	77 sn	16 milisn	
10	173 byte	77 sn	16 milisn	
15	178 byte	75 sn	16 milisn	
20	183 byte	77 sn	16 milisn	
25	188 byte	77 sn	16 milisn	

Şekil 4.6 Deney3 DMSO RTI Koşum Sonuçları 1



Şekil 4.7 Deney3 DMSO RTI Koşum Sonuçları 2

5 SONUÇ VE ÖNERİLER

Bu tez kapsamında bütün benzetim altyapılarında kolaylıkla uyarlanabilecek bir analiz sistemi geliştirilmiştir. Sistem sayesinde farklı amaçlı benzetimlerde kullanılacak altyapının seçiminde altyapılar arası karşılaştırmalar yapılabilmekte, yeni geliştirilecek altyapılar için benzer altyapıya kıyasla performans kontrolleri yapılabilmekte ve benzetim sisteminin tasarım aşamasında en uygun performanslı tasarımın elde edilmesi için deneylerin yapılabilmesi sağlanabilmektedir.

Sistem genelde kapalı bir kutu olarak tasarlandığından kullanım için ekstra iş yükü gerektirmemektedir. Özellikle büyük ölçekli benzetim projelerinde ve gerçek zamanlı benzetim projelerinde kullanılacak altyapıların denenebilmesi için uygun şartların oluşturulabileceği ve analiz işlemlerinin gerçekleştirilebileceği bir ortam sunmaktadır. Analiz sistemi bünyesinden geliştirilen benzetim sistemi tasarlanacak benzetim sistemleri yerine kullanılabilecek şekilde düzenlenebilmektedir.

Temel anlamda gecikme analizi yaparak altyapıya ait karakterizasyonun yapılabilmesiyle birlikte sisteme daha fazla ölçüm yeteneği kazandırarak oluşturulacak karakterizasyonun etkinliği artırılabilir. İlerleyen çalışmalarda CPU kullanımı, dosya kaynak kullanımı ekstra ölçüm metrikleri üzerinden ölçümler alınabilir. Ayrıca çalışma kapsamında ele alınan gecikmeler için çizilen mesaj yaşam döngüleri daha ayrıntılandırılarak analizin derinleştirilmesi sağlanabilir.

Bir bütün olarak performans analiz sistemini oluşturan parçalar, modüler olarak tasarlandıklarından farklı sistemlerde de kullanılabilirler. Örneğin dağıtık olarak çalışan senkron mesaj kayıt sistemi her sisteme ait kayıtların tutulabilmesini sağlar. CORBA ara katmanını kullanmak için yazılan kütüphane ile CORBA öğrenmeye gerek kalmadan CORBA üzerinden çalışan yazılımlar geliştirilebilir. Hazırlanan kütüphane basit servislerin yayımlanmasını ve mesaj göndermek için olay kanalları açılmasını sağlayabilmektedir. DMSO RTI 1.3 ve pRTI1516 için yazılan altyapı adaptörleri benzetim sistemlerini kolaylıkla geliştirmek için uygun birer ortam sunmaktadırlar. RAM DBA performans analizi için geliştirilmiş bir prototip dağıtık benzetim altyapısı olmakla birlikte temel manada dağıtık benzetim sistemi geliştirilmesi için uygundur. Yapılacak düzenlemelerle daha performanslı ve fonksiyonel bir altyapı haline gelebilir. Altyapı Çevirici performans analizi

çerçevesinde altyapılara ait ortak temel yeteneklere ait arayüz sunmakta ve değişik altyapılar arasında değişim yapabilmeyi sağlamaktadır. Desteklenen altyapılara ait daha geniş çerçevede ortak noktaları ele almak ve arayüze eklemek suretiyle tam anlamıyla fonksiyonel bir altyapı çevirici elde edilebilir ve değişik benzetim projelerinde altyapıdan bağımsız benzetimler geliştirilebilir.

KAYNAKLAR DİZİNİ

- [1] Davis P.K., “Distributed interactive simulation in the evolution of DoD warfare modeling and simulation”, Proceedings of the IEEE, Sayı 83-8, pp. 1138-1155, 1995.
- [2] Xiaoxia Shi., Qiuhai Z., “The Introduction on High Level Architecture (HLA) and Run-Time Infrastructure (RTI)”, SICE Annual Conference in Fukui, 2003.
- [3] Kounev S., “Performance Modeling and Evaluation Distributed Component-Based Systems Using Queueing Petri Nets”, IEEE Transactions On Software Engineering, Sayı. 32-7, pp. 486-502, 2006.
- [4] Institute of Electrical an Electronics Engineers, Doc. No: 1516, 2000.
- [5] HLA Status Board, <http://www.dmsomil/public/transition/hla/statusboard>, 2007
- [6] Guofu D., “Study of DCOM-based simulation applied in locomotive & track coupling dynamics”, Parallel and Distributed Computing, Applications and Technologies, pp. 652- 654, 2003.
- [7] Buss A., Jackson L., “Distributed Simulation Modeling: A Comparison Of HLA, Corba, And Rmi”, Winter Simulation Conference, 1998.
- [8] OMG (Object Management Group), “The Common Object Request Broker: Architecture and Specification—Revision 2.2”, Technical Report, 1998.
- [9] Graffagnini, J., “What is ‘True’RTI Performance? And How is it Measured?”, Spring Simulation Interoperability Workshop, 1997.
- [10] Fischer, J., “RTI Performance Testing Concepts”, Fall Simulation Interoperability Workshop, 1998.
- [11] Harless W. N., “Performance and Interoperability Observations of the Gateway Approach to HLA Compliance for Legacy VR Simulations”, Fall Simulation Interoperability Workshop, 1998.
- [12] Purdy S., Wuerfel R. D., “A Comparison Of HLA And DIS Real-Time Performance”, Spring Simulation Interperability Workshop, 1998.
- [13] Sjöström R., Johansson U., “RTI NG and pRTI Performance for Simulated Aircraft Real Time Updates”, Fall Simulation Interperability Workshop, 1998.

- [14] LeSueur K., Burks T., “Latency Performance of Various HLA RTI Implementations”, Spring Simulation Interperability Workshop, 2001
- [15] Fujimoto R., Hoare P., “HLA RTI Performance in High Speed LAN Environments”, Spring Simulation Interperability Workshop, 2000
- [16] Fujimoto R., Ferenci S., “RTI Performance on Shared Memory and Message Passing Architectures”, Spring Simulation Interperability Workshop, 2000
- [17] Pitch, <http://www.pitch.se/products/pitch-prti/specifications-pitch-prti.html>, 2007
- [18] RTI Next Generation <http://www.virtc.com/rtingpro-full.jsp>, 2007

ÖZGEÇMİŞ

Ramazan Cengiz, 1980 yılında Afyon'un İscehisar ilçesinde doğdu. 2003 Haziran ayında İTÜ Elektrik-Elektronik Fakültesi'nden Bilgisayar Mühendisi olarak mezun oldu. 2003 Eylül ayında Gebze Yüksek Teknoloji Enstitüsü'nde Bilgisayar Mühendisliği bölümünde Yüksek Lisans Eğitimine başladı. Yine 2003 Eylül ayında TÜBİTAK Marmara Araştırma Merkezi, Bilişim Teknolojileri Enstitüsünde "Araştırmacı" olarak çalışmaya başladı. TÜBİTAK tarafından başarıyla tamamlanan GENESIS prototip geliştirme projesinde veri kayıt sistemleri ve ağ haberleşme kütüphaneleri görev yazılımlarını geliştirdi. 2004 Ekim ayında SGKS C4ISR-MOS simülasyon projesiyle birlikte dağıtık simülasyon sistemleri üzerinde çalışmaya başladı. Proje kapsamında dağıtık altyapı hizmetlerinin sağlanması, veri madenciliği ile karar destek yazılımlarının geliştirilmesi, simülasyon sistemi çerçeve geliştirilmesi ve C4ISR sistemleri etkinlik analiz yazılımları çalışmalarında görevler aldı. Halen TÜBİTAK Marmara Araştırma Merkezi bünyesinde Modelleme ve Benzetim Sistemleri konularında çalışmalarına devam etmekte.

EK 1 Altyapı Çevirici API Açıklamaları

Altyapı Çevirici API:

DistributionServiceInterface *

GetDistributionServiceObject(char *DLL_NAME);

Dağıtık Altyapı Arayüzü nesnesini döndürür.

DLL_NAME kullanılmak istenen altyapıya ait adaptör kütüphanesinin ismidir. Benzetim Koşumu sırasında yürütülecek altyapı haberleşmeleri döndürülen bu nesne ile yapılır.

void DeleteDistributionServiceObject();

Oluşturulan altyapı arayüz nesnesinin silinmesini sağlar.

Altyapı Arayüz API:

void SendEvent(char* _name, EventBase *eventMes,char * _mesunique)

_name Mesajı gönderen ferde ismi

eventMes Gönderilen mesaj

mesunique Gönderilen mesaja ait tekil isim

Mesaj Gönderme metodudur.

void SetTimeAdvanceGrantCallback(void f(bool)) ;

f Zaman Artırım Kabul Geri Bildirim fonksiyonudur

Zaman artırım isteklerinin kabul edilmesi halinde çağrılacak fonksiyonu altyapı bildirmek için kullanılır

void Subscribe(char *id,char *_eventName, EventCallbackPtr ECPtr)

id mesaja kaydolan federe ismi

eventName kayıt olunan mesaj ismi

ECPtr Mesaj geldiğinde çağrılacak metod

double StartSimulationTime(char* _id)

_id Zamana yönetimine kaydolan federe ismi

double StopSimulationTime(char* _id)

_id Zaman yönetimini durduran federe ismi

void InitializeRTI()

Altyapılar için başlangıç işlemlerinin yapılmasını sağlayan metod

void CreateFederation(char *_name,char *_path)

_name Oluturulacak federasyon ismi

_path Federasyon tanımlamalarını içeren konfigüreasyon dosyasının adresi

Yeni bir federasyon oluşturmak için çağrılan metod

void JoinFederation(char *fedName, char *fedrName)

fedName Federasyona katılacak federe ismi

fedrName Katılınacak federasyon ismi

Oluşturulmuş bir federasyona katılmak için çağrılan metod

void RegisterSyncPoint(char*_syncName,char*syncText)

_syncName Senkronizasyon noktası ismi

syncText Senskronizasyon noktası açıklaması

Senkronizasyon noktasına kaydolma metodu

bool IsSyncOk(char*_syncName)

_syncName Senskronizasyon noktası ismi

Sekronizasyon noktasına gelinildi mi kontrolü

void RequestTimeAdvance(double _timeStep)

_timeStep Atılacak adım, saniye cinsinden

Zaman artırım isteği

void Tick()

Güncellemelerin yapılması için tetikleme metodu

double EnableTimeConstrained()

Constrained olarak zaman yönetimine katılma

double EnableTimeRegulation(double _simTime,double _lookAhead)

_simTime Atanacak Zaman

_lookAhead en yakın güncelleme zaman aralığı

Regulated olarak zaman yönetimine katılma metodu.

bool IsTimeGranted()

Zaman artırım isteği kabul edil di mi sorgusu

EventBase * CheckMessage()

Gelen mesajları alınması için çağrılan metod.

FederateInterface: Benzetim altyapısının, benzetim sistemine olan arayüzüdür. Benzetim ortamından federasyona katılma-çıkma, nesne yayımlama-kaydolma, mesaj gönderme-alma, zaman artırım isteği-kabulü gibi bütün işlemler bu sınır aracılığıyla yapılır.

COwnerMgmt, CDataDistMgmt, CObjectMgmt, CTimeMgmt, CFedAmb, CDeclarationMgmt, CFedMgmt sınıfları HLA servislerini uygulamak amaçlı oluşturulmuş sınıflardır. Her sınıf ilgili olduğu sınıfa ait işlemleri yürütür.

Callback: Altyapı içinde gelen mesajların iletilmesi için tanımlanan geri bildirim nesnesi. CallbackList: Son güncellemede gelen bütün geri bildirimleri tutan liste

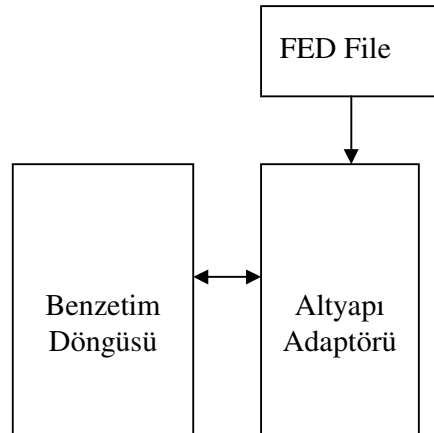
simSOMList: Federe tanımlarında bulunan nesnelere ait yazılım modelleri listesi.

simObjectClass: Tanımlı nesne sınıfları listesi.

simInteractionClass: Tanımlı ileti sınıfları listesi.

simData: Federe tanımlarına uygun hazırlanan yazılım modeli temel veri nesnesi. İşlemler simData tipi üzerinden yapılır.

DMSO RTI 1.3v6 Adaptörünün Kullanımı



Şekil 5.2 DMSO RTI 1.3v6 Adaptör Kullanım Şeması

Adaptör Benzetim döngüsünde benzetim yönetimiyle alakalı bütün işleri yapabilecek yetenektedir. Sahip olduğu arayüze ait API tanımlamaları EK-1 de ifade edilmiştir.

Bu kapsamda herhangi bir benzetim sistemi geliştirmesinde RTI kullanılacaksa, benzetim modellerimize göre FED dosyası HLA kurallarına uygun bir şekilde doldurulur. Bu dosya RTI 1.3 yazılımının standart federasyon tanımlamalarını okuduğu dosyadır. Dosya doldurulduktan sonra dosya sisteminde kayıtlı olduğu adres bilgileri adaptöre gönderilecek federasyona katılma işlemleri yürütülür.

EK 3 CORBA Kütüphane API Açıklamaları

void Initialize(int argc, ACE_TCHAR* argv[])

Corba servisinin başlatılması ve yeni bir ORB nesnesi oluşturmak için çağrılır.

void FindNameService()

Yayınlanan nameservice varsa bulmak için kullanılır

int GetRootPoa()

Ana objeyi almak için kullanılır

void ActivatePoaManager()

Pro manageri aktif hale getirmek için kullanılır.

void DeclareNewCorbaEnv()

Yeni bir ENV tanımlaması yapılır

void ResolveEventChannel()

Ön tanımlı event kanalına bağlanılır

void ResolveEventChannelbyName(char* _name)

_name bağlanılacak event kanlının ismi

İsmi verilen kanala bağlanılır

CORBA::Object_var ResolveService(char* _name)

_name Alınacak servis ismi

İsmi verilen servis nesnesi alınır

void ExportRootPoa(char* filename)

RootPoa nın bilgileri fileName isimli dosyaya kaydedilir.

bool ConnectServer(char* name)

name İsimli sunucuya bağlanılır

**void ConnectAsConsumer(char *consumerId,char* _eventChannelName =
COS_EVENT_SERVICE_NAME)**

consumerId Bağlantı ismi

eventChannelName Event kanalı

Event kanınla istemci olarak bağlanılır

**void ConnectAsSupplier(char* _name = "EventSupplier", char*
_eventChannelName = COS_EVENT_SERVICE_NAME)**

name Bağlantı ismi

eventChannelName EVent kanalı

Event kanlına sunucu olarak bağlanılır.

void DisconnectAsSupplier(char* _eventChannelName)

Event kanlından bağlantı koparılır.

void Run()

ORB koşumu başlatılır

void Stop()

ORB koşumu dururulur.

void RunInThread()

ORB başka bir thread üzerinde koşturulur

void StopOrbThread()

ORB koşum threadini durdur

void DestroyEventChannel(char* _name)

_name isimli event kanlını siler

void BindService(char* _name, CORBA::Object_var p_servant)

_name Yayınlama ismi

P_servant Yayınlanan nesne

Bir servisin yayınlanması

void ActivateObject(PortableServer::Servant p_servant, char* _name)

p_servant yayımlanan nesne

_name yayımlama ismi

void BindToEventChannelbyName(char* _name)

_name isimli event kanlını oluşturur

void BindToEventChannel()

Ön tanımlı bir event kanlı oluşturur

void DeActivateObject(PortableServer::ObjectId_var oid, char* name)

name isimli objeyi kaldırır.