T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

DAĞITIK SİMÜLASYON SİSTEMLERİ İÇİN YENİ BİR YÖNLENDİRME ALGORİTMASI VE UYGULAMASI

DOKTORA TEZİ

Yük. Elek.-Elektr. Müh. Ahmet ZENGİN

Enstitü Anabilim Dalı	**: ELEKTRİK – ELEKTRONİK MÜH.**
Enstitü Bilim Dalı	**: ELEKTRONİK**
Tez Danışmanı	**: Prof. Dr. Hüseyin EKİZ**

Temmuz 2004

T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

DAĞITIK SİMÜLASYON SİSTEMLERİ İÇİN YENİ BİR YÖNLENDİRME ALGORİTMASI VE UYGULAMASI

DOKTORA TEZİ

Yük. Elek.-Elektr. Müh. Ahmet ZENGİN

Enstitü Anabilim Dalı : ELEKTRİK – ELEKTRONİK MÜH.
Enstitü Bilim Dalı : ELEKTRONİK

Bu tez .. / .. /2004 tarihinde aşağıdaki jüri tarafından Oybirliği/Oyçokluğu ile kabul edilmiştir.

Prof. Dr. Hüseyin EKİZ — **Jüri Başkanı**
Prof. Dr. Ercan ÖZTEMEL — **Jüri Üyesi**
Prof. Dr. Etem KÖKLÜKAYA — **Jüri Üyesi**

Doç Dr. İsmail ERTÜRK — **Jüri Üyesi**
Doç. Dr. Osman GÜRDAL — **Jüri Üyesi**

ÖNSÖZ

Çalışmalarım esnasında değerli katkılarını hiçbir zaman esirgemeyen, yorum ve yönlendirmeleriyle çalışmalarıma ışık tutan ve takıldığım her noktada verdiği motivasyonla çalışmalarımın ilerlemesini ve tamamlanmasını sağlayan, Sayın Prof. Dr. Hüseyin EKİZ'e sonsuz şükranlarımı sunarım.

Çalışmamın uygulama kısmını birlikte tamamladığım, beraber çalıştığımız süre boyunca tecrübesinden ve önerilerinden büyük istifade ettiğim, ayrıca bana büyük vizyon sunan Sayın Prof. Hessam SARJOUGHIAN'a teşekkür ederim.

Tez konum ile ilgili yaptığı motivasyon ve yönlendirmeden dolayı, konu ile ilgili yazılımlar ve araçlar konusunda değerli destekleri ve katkıları nedeniyle Sayın Prof. Hasan ÇAM'a da teşekkürü bir borç bilirim.

TÜBİTAK'ın imkanlarını kullanımıma sunarak, yeni teknolojiler ve kaynaklara erişimime imkan sağladıklarından dolayı Sayın Prof. Dr. Ercan ÖZTEMEL'e ve Dr. Cüneyd FIRAT'a teşekkür ederim.

Bu tez çalışmasının başından sonuna kadar her zaman yanımda olan, moral motivasyonlarını her zaman yanımda bulduğum, Arş.Gör. Yavuz BAYAM ve Arş.Gör. Mehmet Recep BOZKURT'a da sonsuz teşekkürlerimi sunarım.

Çalışmam boyunca maddi ve manevi desteklerini aldığım Aileme teşekkürü bir borç bilirim.

Yanıkköy, Temmuz 2004 Ahmet Zengin

İÇİNDEKİLER

KISALTMALAR LİSTESİ

AI : Yapay Zeka (Artificial Intelligence)

BDI : Doktrinler, İstekler ve Niyetler (Beliefs, Desires, Intentions)

B-ISDN : Genişband Tümleşik Hizmet Sayısal Ağları (Broadband Integrated Services Digital Networks)

CM : Birleşik Model (Coupled Model)

CPU : Merkezi İşlem Birimi (Central Processing Unit)

CSV : Virgülle Ayrılmış Değerler (Comma Seperated Values)

DAI : Dağıtık Yapay Zeka (Distributed Artificial Intelligence)

DESS : Diferansiyel Denklemli Sistem Tanımı (Differential Equation System Specification)

DEVS : Ayrık Olaylı Sistem Tanımı (Discrete Event System Specification)

DOC : Dağıtık Nesne Hesaplama (Distributed Object Computing)

DTSS : Ayrık Zamanlı Sistem Tanımı (Discrete Time System Specification)

EF : Deneysel Çerçeve (Experimental Frame)

FDG : Formalizm Dönüşüm Grafiği

FIFO : İlk Giren İlk Çıkar (First-In, First-Out)

FSM : Sonlu Durum Mekanizması (Finite State Machine)

GUI : Grafiksel kullanıcı arayüzü (Graphical User Interface)

HLA : Yüksek Seviyeli Yapı (High Level Architecture)

IC : Entegre (Integrated Circuit)

ID : Kimlik

I/O : Giriş ve Çıkış

IP : İnternet Protokolü

JVM : Java Sanal Mekanizması (Java Virtual Machine)

LAN : Yerel alan ağı (Local Area Network)

LSA : Link Durum İlanları (Link State Advertisement)

MAS : Çoklu Görevli Sistemleri (Multi-Agent Systems)

M&S : Modelleme ve Simülasyon

NIC : Ağ Arabirim Kartı (Network Interface Card)

OSI : Açık Sistem Bağlantısı (Open System Interconnection)

OS : İşletim Sistemi (Operating System)

OSPF : Açık En Kısa Yol İlk (Open Shortest Path First)

QoS : Hizmet Kalitesi (Quality of Service)

RIP : Yönlendirme Bilgi Protokolü (Routing Information Protocol)

SNMP : Basit ağ yönetim protokolü (Simple Network Management Protocol)

SPF : En Kısa İlk Yol (Shortest Path First)

TTL : Yaşama Zamanı (Time to Live)

UML : Birleşik Modelleme Dili (Unified Modeling Language)

VHDL : VHSIC donanım tanımlama dili (VHSIC Hardware Description Language)

VHSIC : Yüksek hızlı entegre devre (Very High Speed Integrated Circuit)

ŞEKİLLER LİSTESİ

TABLOLAR LİSTESİ

ÖZET

Anahtar Kelimeler: Modelleme ve Simülasyon (M&S), DEVS, Ağ Yönetimi, Ekoloji, İnternet

Dağıtık sistemler, çeşitli algoritmalar ve teknolojiler kullanarak birbirleriyle iletişim yapan birimlerden oluşur. İletişim içerisinde bulunan sistemlerin uyarlanabilirlik, ölçeklenebilirlik, güvenilirlik (sürdürülebilirlik) gibi bir takım niteliklere sahip olması gerekmektedir. Sürekli yeni servis türlerinin ve heterojen ağların bir bütün olarak dahil edilmesiyle, ağlar karmaşık bir hal almaktadır. Ağ sistemlerinin, sistemin büyüyerek daha karmaşık bir hal alması karşısında yeni ve daha gelişmiş servisleri sunması beklenmektedir. Artan karmaşıklık ve boyut nedeniyle ortaya çıkan sorunların üstesinden gelmek amacıyla geliştirilen çeşitli yöntemler bilgisayar ağlarının ihtiyaçlarına göre kullanılmaktadır. Günümüzde ağlar, hız ve işlem yapma gücündeki ihtiyaçlara cevap verebilmek için hesaplama işlevleri merkezi bir yapıdan dağıtık bir yapıya doğru kaymaktadır. Performans / maliyet oranının göz önünde tutulması zorunluluğu bu değişimi yeni işlem yapma kapasitelerinin tasarlanması üzerine daha fazla yöneltmektedir. Bu tezde, dağıtık sistemlerin karmaşıklık, ölçeklenebilirlik, vb. sorunlarının incelenmesi, tasarım alternatiflerinin araştırılması ve farklı çözüm yaklaşımlarının incelenmesi amacıyla modelleme ve simülasyon araçlarının kullanıldığı bir çalışma / uygulama gerçekleştirilmiştir.

İnternetin yakın gelecekte 1 milyar düğüme erişeceği düşünülürse, ölçeklenebilirlik kavramının ağların yönetiminde, modellenmesinde ve simülasyonunda yeni boyutlar kazanacağı açıktır. Statik topoloji üreten simülatörler (COMNET, NS2, OPNET, vb.) küçük ağları çalışmak için ideal platformlarken, günümüzde üstel olarak artan ağ sistemlerini modellemede ve değişken yapılı ağ sistemlerinin performansını test etmede yetersiz kalmaktadırlar. Ayrıca, bu simülatörlerin mimarilerinin bir çoğu soyutlama ve hiyerarşiden yoksun olmaları yanında çok büyük hesaplama maliyeti oluşturmaktadırlar. Yapılan çalışmada, belirtilen kısıtlamaları / sakıncaları ortadan kaldırmaya yönelik olarak DEVS metodolojisi kullanılarak bir ağ simülatörü geliştirilmiştir.

Tasarlanan ağ sisteminin modellenmesi; ağ bileşenlerinin tanımlanmasını, bu bileşenlerde çalışacak yazılım nesnelerinin, etkileşimlerinin ve bu varlıkların işlem yapan düğümlere dağıtılmalarını, ağ topolojilerinin ve iletişim protokollerinin tanımlanmasını içermektedir. Düğümler ve linkler temel ağ bileşenleri olarak tanımlandıktan sonra, DEVS birleşik model tanımı kullanılarak temel bileşenler birbirine bağlanıp birleşik ağ modelleri oluşturulmuştur. Geliştirilen ağ ortamı farklı yönlendirme algoritmalarını (en kısa yol, uzaklık vektörü, oğul zekası, vb.) modelleyebilme yeteneğine sahiptir.

Geliştirilen simülatörün üstünlüklerini ve performansını göstermek amacıyla binlerce düğümden oluşan ağlar modellenmiştir. Modellenen ağlar farklı trafik yükleri altında çalıştırılarak, çalışma sırasında geliştirilen yönlendirme algoritmasının performansı incelendi. Gerçekleştirilen uygulamalardan, geliştirilen simülatörün son derece paralel, esnek ve hızlı çalıştığını, değişik teknolojileri barındıran uygulamaları geliştirebilme yeteneğine sahip olduğu gözlemlendi.

Tez içerisinde yapılan çalışmalar dört grup altında özetlenebilir:

i- Farklı yönlendirme algoritmalarının incelenebilmesine olanak tanıyan bir ortam oluşturulması amacıyla örnek bir ağ modelinin DEVS (Discrete Event System Specification) kullanılarak modellenmesi ve simülasyonu işlemleri gerçekleştirildi.

ii- Büyük ölçekli biyolojik sistemlerde (karıncalar, balarıları, termitler, vb.) kullanılan optimizasyon düzeneklerinden esinlenerek ağlarda kullanılacak kural-tabanlı yeni bir yönlendirme algoritması geliştirildi.

iii- Oluşturulan ağ modeline hali hazırda kullanılmakta olan yönlendirme algoritmaları ile çalışma sırasında geliştirilen biyolojik-tabanlı yönlendirme algoritması uygulanarak, özellikle biyolojik-tabanlı yönlendirme algoritmalarının klasik yönlendirme algoritmalarıyla karşılaştırılması yapıldı.

iv- Geliştirilen algoritmanın büyük ölçekli ağlarda kullanılabilirliğini göstermek amacıyla, çeşitli boyutlarda ağ modelleri oluşturularak kural-tabanlı algoritmanın performansı incelendi.

A ROUTING ALGORITHM FOR DISTRIBUTED SIMULATION SYSTEMS

SUMMARY

Keywords**:** Modeling & Simulation, DEVS, Network Management, Ecology, Internet

Network systems must communicate with one another using a variety of algorithms and technologies. Many systems supporting interconnectivity are required to exhibit essential traits such as adaptability, scalability, and reliability (survivability). At the same time, these networked systems are expected to offer new and more sophisticated services in the face of increasing system heterogeneity. Consequently, to cope with the management of such networks in the presence of ever increasing complexity, various decentralized and centralized approaches are being used to address the needs of private and public organizations. In this thesis, in order to examine some problems of distributed systems such as complexity and scalability, search for design alternatives and propose various resolution levels, a modeling and simulation study is performed in which modeling and simulation methodologies and tools are used. Scalability issue has become very crucial concept for modeling and simulation of the Internet. Simulators generating static topology such as COMNET, NS2, and OPNET are ideal platforms for studying small networks, but incapable of modeling and testing large-scale and dynamic structure networks. Furthermore, due to lack of hierarchy and abstraction in their structure, it is difficult to create and manage large model families. In this study, a network simulator is developed to bring solutions to above problems by using DEVS modeling and simulation methodology.

To develop and study dynamic and adaptive swarm-based routing protocols of biological, we have devised a DEVS (Discrete Event System Specification) network model. The nodes and links are characterized as the elementary network components. These models and networks are implemented in the DEVSJAVA modeling and simulation environment which is an implementation of the DEVS framework. The developed network environment is capable of representing behavior of different routing algorithms (e.g. shortest-path, distance vector and other swarm algorithms). For the purpose of modeling of a distributed networked system, we define a network in terms of its components (e.g. nodes and links) and their hierarchical structure. The network is then modeled as DEVS atomic and coupled models. To do this we closely examine biological aspects of honeybee colony and their mappings into DEVS models. For example, the routing policy of the network which is embedded in every node is similar to a honeybee having its own capability to search for food and communicate with other honeybees. Using the DEVS hierarchical model composition concept, we develop simulation models of networks with varying topologies and scales. For example, we will use clusters to study its impact on reducing communication and increasing performance. The explicit and hidden behaviors of these networks are observed under various experimental configurations – e.g., nodes and links are assigned different capacities.

In this thesis, performed operations can be summarized into four steps:

i- Modeling and simulation of distributed systems together with nodes and links components using DEVS.

ii- Developing a rule-based swarm intelligence routing algorithm by inspiring from honeybee scout-recruit system.

iii- Modeling a state-of-the-art routing algorithm in order to depict advantages of developed algorithm.

iv- Creating large-scale networks models having from tens to several thousands of components and observing the behavior of developed algorithm.

BÖLÜM 1. GİRİŞ

Bilgisayar ağları, yeni servis türlerinin ve heterojen ağların bir bütün olarak mevcut ağlara dahil edilmesiyle sürekli büyümekte ve gittikçe daha karmaşık bir hal almaktadır [1]. Ağ hızında ve işlem yapma gücündeki hızlı artış gereksinimi, ağ içerisinde gerçekleştirilen yönlendirme ve yönetim işlemlerinin, merkezi bir yapıdan dağıtık bir yapıya doğru kaymasını zorunlu kılmaktadır. Performans / maliyet oranının göz önünde tutulması zorunluluğu, araştırmacıları ağ üzerinde yeni işlem kapasitelerinin eklenmesi ve yeni yöntemlerin tasarlanması konusuna daha fazla yöneltmektedir. Bu yönelim, nesneye yönelik teknolojilerle birlikte yazılım endüstrisinde yoğunlaşmakta ve yazılım endüstrisindeki yoğunlaşma sonucunda nesneye yönelik eğilim birbirleri ile etkileşen yazılım bileşenlerinin modellenmesi ve tasarlanması popüler konular olarak ortaya çıkmaktadır.

Milyarlarca insanın global ağ içerisinde çalışan uygulamalara günlük yaşamlarının bir parçası olarak düzenli olarak erişecek olması, yakın gelecekte gerçekleşecek olaylardan birisi olacağı tahminleri yapılmaktadır [2]. Bu tahminleri gerçekleştirmek için, ağ uygulamalarının üç özelliğe sahip olması gerekmektedir:

i) Büyük boyutlu talepleri karşılayacak bir ölçekte olmalıdırlar.

ii) Dinamik kullanıcı taleplerine ve ağ şartlarına kolayca uyum sağlamalıdırlar.

iii) Kısmi hatalar karşısında ayakta kalabilmeli ve kullanıcılara hizmet vermeye devam edebilmelidirler.

Ekonomik ve ticari sebepler ile, bilgisayar ağlarının bütün olası durumlar altında başarılı çağrı bağlantısını garanti edecek bir donanıma sahip olması yerine, birçok kullanım durumunda kabul edilebilir bir performans sunacak en düşük seviyede bir donanıma sahip olması tercih edilmektedir. Ağ şartlarında önemli bir değişim olması durumunda, kapasite sınırlamaları ve bağlantı kurulamayan çağrılar nedeni ile sistem / ağ hatalı çalışma durumuna gidebilir. Ağ içerisinde bulunan iki nokta arasındaki iletişimde takip edilecek birden fazla alternatif yol bulunabilir ve iletişim / iletilen mesaj bir takım ara anahtarlama istasyonları veya düğümler üzerinden yönlendirilebilir [3]. Bu yapı nedeni ile yedek kapasiteye sahip ağ kısımları kullanılarak mesajların yönlendirilmesi ve mevcut veya potansiyel tıkanıklıkların hafifletilmesi mümkündür. Bu işlem 'yük dengeleme' olarak adlandırılır ve sistemdeki değişken yükü düğümlere / birimlere eşit olarak dağıtan ve kayıp çağrıların sayısını en aza indirgeyen çağrı yönlendirme sistemlerinin kurulması olarak tanımlanır. Ağdaki bir düğümden diğer bütün düğümlere en kısa yolu belirlemek ve bu şekilde düğümlerin ortalama kullanımını en aza indirgemek ağdaki trafiğin nasıl dağıtılması gerektiği ile yakından ilgiliyken, düğüm tıkanıklıklarından sakınmak için ideal bir yöntem değildir.

Bölüm 4'te detaylandırılan, yönlendirme algoritmalarının temel görevi; ağ performansını en üst seviyeye getirmek ve maliyeti en aza indirgeyerek ağın sağlıklı hizmet vermesine yardımcı olmaktır. Bir ağa yönlendirme algoritmalarının uygulanmaması durumunda tıkanıklıklar meydana gelir ve bu tıkanıklıklar büyük miktarda verinin kaybolmasına neden olur. Ağ teknolojileri ile birlikte gerçekleşen yönlendirme algoritmalarının geliştirilmesi ve araştırılması ağ alanında yapılan çalışmalarda önemli bir yere sahiptir. İlk olarak telefon ağlarında bir tür dinamik çağrı yönlendirme şemaları şeklinde uygulanan yönlendirme metotları / algoritmaları geçirdikleri hızlı değişim sonucunda günümüzde birçok ağ sisteminde ve internet içinde yaygın kullanıma sahip olmuştur.

Ağ kontrol veya yönlendirme mekanizmaları / yöntemleri merkezi (centralized) ve merkezi olmayan / dağıtık (decentralized) şeklinde ikiye ayrılabilir [3]. Merkezi bir

kontrol sistemi aracılığıyla dağıtık sistemleri kontrol etmenin çeşitli sakıncaları bulunmaktadır. Merkezi sistemlerde kontrolör sistemin her parçasından kontrol birimine iletişim bağlantılarını gerektirir ve bütün sistem hakkında güncel bir bilgiye gereksinim duyar. Çalışma ve iletişim maliyetlerinin sistem boyutuyla birlikte hızla artması nedeni ile, merkezi kontrol mekanizmaları büyük ağ boyutlarında ölçeklenemez ve oluşan kontrolör hatası çoğu kez bütün sistemin çökmesiyle sonuçlanır. Bilgisayar ağları gibi dağıtık sistemlerin özellikleri; dinamik, karmaşık, davranışı önceden kestirilemez, davranışı tek bir kontrol faktörüne indirgenemez, vb. şeklinde özetlenebilir [4].

Merkezli olmayan bir ağ yönetim sistemi, yukarıda bahsedilen problemleri ortadan kaldırır [5]. Bu yöntemde, birden fazla benzeri varlık veya gezgin görevli (mobile agent) birbirleri ile işbirliği / etkileşim yaparak yönlendirme işlevini gerçekleştirir. Merkezli olmayan yaklaşım, yönlendirme sisteminin yönetimini daha fazla karmaşıklaştırsa da, hata toleransını arttırmak, ağdaki değişimlere hızlı uyum sağlamak, kolay ölçeklenebilirlik gibi sayısız avantajlara sahiptir. İşlevin birden fazla birim üzerine yayılması yönlendirme sisteminin hata toleransını artırması yanında, ağdaki trafik değişimlerine hızlı cevap verilmesini sağlar. İşlevi birden fazla varlık üzerine yaymak, tek bir varlıkta gerekli yönlendirme sistemi kaynaklarının büyüklüğünü düşürür ve yönlendirme sisteminin ağ boyutuyla artarak büyümesini sağlar (ölçeklenebilirlik). Bu yöntem, ağ içindeki bütün düğümlerde bulunan dağıtık işlem yapma kapasitesini ve kaynaklarını kullanır.

Başlangıçta 5 üniversite arasında test ağı olarak tasarlanan ilk ağ yapısının (ARPANET), yakın gelecekte ulaşılacak bir milyar düğümlü İnternet yapısına erişeceğini tahmin etmek imkansızdı [6]. Ağların bu şekilde global bir yapı ile hızla büyümesi, ağlarda ölçeklenebilirlik konusunun yeni anlamlar kazanması yanında, yeni kavramların / konularında teknolojiye eklenmesine neden oldu. Ağ konusunun gelişmesi, Bilgisayar Bilimleri, Kuyruklama Teorisi, Dağıtık Hesaplama, Ağ Trafik Mühendisliği, vb. konuların araştırma alanları olarak ortaya çıkması ve Bilgisayar Ağları dalının bu disiplinler ile ilişkisini ortaya çıkardı [7].

İnternet'in mevcut boyutuna ve karmaşıklığına erişmeden önce, küçük homojen ağlarda yönlendirme algoritmalarını tasarlamak, test etmek ve modelleme /

simülasyon yoluyla prototip ağları incelemek kısmen mümkündü. Bununla beraber, bu algoritmalar, günümüzün son derece çeşitlilik arz eden İnternet ortamında hala kullanılan temel algoritmaları oluşturmaktadır. Hatalara karşı hassasiyet derecesini tespit etmek, daha sağlam yönlendirme algoritmalarını tasarlamak ve test etmek amacıyla İnternet gibi büyük ölçekli ağlar üzerinde deney yapmak günümüzde mümkün olmamakta, bütün ağ sisteminin davranışını çözmek veya hatalar karşısında ağın çökmesini önlemek gibi problemleri çözme konusunda mevcut simülasyon araçları yetersiz kalmaktadır. Bununla birlikte, İnternet ortamının davranışının modellerini oluşturmak amacıyla yeterince veri ve analiz yöntemi bulunsa idi, kritik hataların ve yapısal zayıflıkların tespit edilebilmesi yanında olası hatalara karşı önlemler alınabilirdi.

Dağıtık sistemleri ve iletişim ağlarını doğrudan modellemek amacıyla kullanılabilecek yazılımlar (COMNET, OPNET, NS2, GLOMOSIM, JNS, vb.) yanında donanım bileşenlerini modellemek için kullanılabilecek araçlar (VHDL araçları gibi) bulunmaktadır [8] [9]. En yaygın kullanılan ağ simülatörleri olan NS2, OPNET, COMNET, vb. simülatörler küçük boyutlu ağları çalışmak için ideal platformlardır. Bu ürünler / yazılımlar kullanılarak büyük ölçekli sistemleri modellemek ve simüle etmek zor olmasının yanında, bu yazılımlar değişik teknolojilerin içine katıldığı sistemleri modelleme yeteneğine sahip değildirler [2]. Ağ sistemlerinin boyutlarının üstel olarak artışı karşısında statik topoloji üreten bu simülatörler yetersiz kaldılar ve gelişen sistemlerin performansını doğru bir şekilde test edemez / ölçemez duruma düştüler. Bu eksikliklerin ortaya çıkmasının sebepleri; basit simülatörlerin yapısal sınırlamaları ve büyük ölçekli karmaşık yapılı ağların yetersiz bir şekilde analiz edilmesi olarak sıralanabilir. Belirtilen kısıtlamaları / yetersizlikleri bir ölçüde karşılayabilen ve binlerce düğümü modelleyebilen GLOMOSIM, PDNS, vb. simülatörler bulunsa da, bunların hiçbiri dinamik, gelişebilir, yeniden boyutlanabilir ve değişik trafik şartlarına uyarlanabilir (adaptif) bir ağı modelleyememektedir [2][10][11]. Ayrıca, bu tür simülatörlerde bileşenler modüler ve hiyerarşik bir yapıda olmadığından bileşenlerin yeniden kullanımı, değişik uygulamalara uyarlanabilirliği ve hiyerarşik tasarımı zordur. Klasik simülatörlerin çalıştırıldıkları bilgisayarlarda oldukça yüksek kaynak kullanım gereksinimleri, gelişmiş ve büyük uygulamaların meydana getirilmesini

zorlaştırmaktadır. Bu simülatörler hakkında dikkate değer bir çalışma Mittal ve Zeigler tarafından yapılmıştır [2].

Sonuç olarak, ağ sistemlerinin karmaşıklık, ölçeklenebilirlik, vb. problemlerinin çözülmesi amacıyla gelişmiş yönlendirme sistemlerini tasarlamak / test etmek için bileşenlerin hangi seviyede soyutlanması gerektiğini, hangi elemanlar arasında ne tür bir ilişki olduğunu belirlemek amacıyla kullanılacak gerçek dünya verisi, modelleme / simülasyon desteği ve büyük ölçekli sistemleri tasarlama yaklaşımı gerekmektedir. Bu çalışma, yukarıdaki gereksinimleri karşılamak üzere biyolojik mekanizmalar ile modelleme ve simülasyon araçlarının kullanılmasını ve uygulanmasını içermektedir.

1.1 Biyolojik Sistemler ve Yeni Ağ Uygulamaları

Geleceğin ağ uygulamalarında karşılaşılacak problemlerin, büyük ölçekli biyolojik sistemlerde halledilmiş durumda olduğu ve geleceğin ağ uygulamalarının bu önemli prensipleri ve mekanizmaları benimseyeceği geniş bir araştırıcı grubu tarafından kabul edilmektedir [4] [5] [12] [13] [14] [15] [16] [17] [18] [19] [20]. Büyük ölçekli biyolojik sistemler, ölçeklenme, uyarlanma ve hayatta kalma konularında gelişmiş mekanizmalara sahiptirler [21]. Örneğin, bir arı kolonisinde kovan içerisinde halledilen işlerin çoğu herhangi bir merkezi kontrol otoritesine bağlı olmadan gerçekleştirilebilir ve koloni çok sayıda arıya ölçeklenebilir. Arılar kendi başlarına (otonom) hareket ederler ve sadece yerel şartlar ile diğer arılarla yaptıkları yerel etkileşimlerden etkilenirler. Kovanı inşa ederken sadece tamamlanmış altıgen hücrelerin yapısını takip eden ve herhangi bir merkezi otoriteden emir almayan arılar, dinamik şartlara kendilerini uyarlayabilirler ve enerji tüketimine bağlı olarak besin kazancını optimize edebilirler. Kovandaki bal miktarının çok düşük olması durumunda; büyük miktarda bal toplayıcı arı kovanı terk ederek nektar aramaya giderken, kovanın bal bakımından dolu olması durumunda; arıların çoğu kovanda kalarak istirahat ederler. Arı kolonisi kraliçe arı dahil tek bir arıya bağlı olmadığından, kovandaki bazı arılar ölse bile koloni yaşamını sürdürür. Bir arı kolonisinin istenen karakteristikleri olan ölçeklenebilirlik, uyarlanabilirlik ve hayatta

kalabilirlik tek bir arıda bulunmamasına rağmen, bu özellikler koloni içindeki bütün arıların kolektif hareketlerinden ve etkileşimlerinden ortaya çıkar [21].

Nispeten basit birimlerin davranışından ve bunların kendi aralarındaki etkileşimlerinden ortaya çıkan *'karmaşık kolektif davranış' (complex collective behavior)* düşüncesi, sanal ortamlarda '*yapay yaşamlar' (artificial life)* meydana getirme alanı için temel teşkil etmektedir [4]. Bu sistemlerin anlaşılmasının gelişmesi / kolaylaşması, ortaya çıkan ortak davranış tarafından kontrol edilen yapay sistemler oluşturma şansını artırmaktadır. Karmaşık kolektif davranış kavramının kullanımının iletişim ağlarındaki yük dengeleme ve yönlendirme algoritmaları gibi dağıtık sistemlerin yönetiminde tamamen yeni yöntem ve yaklaşımlara neden olabileceği düşünülmektedir. Karıncalar, arılar, vb. karmaşık kolektif davranış sergileyen sosyal canlıların merkezi olmayan ve dağıtık doğası, son derece dağıtık ve dinamik bir davranış gösteren ağ yönetim sistemleriyle benzerlikler taşımaktadır [21].

1.2 Biyolojik Sistemlerdeki Mekanizmaların Bilgisayar Ağlarına Uygulanması

Dağıtık sistemlerin davranışsal karmaşıklığı, bileşenlerin dinamiklerinden ve bileşenler arası yapısal ilişkilerden kaynaklanır. Dağıtık bir ağ sisteminin tasarımı sırasında verilen kararlar, bileşenlerin tek tek dinamiklerini, bu bileşenlerin birbiriyle olan bağlantısını, yapısını, gelişim / değişim altındaki sistemin performansını ve davranışını önemli oranda etkilemektedir. Örneğin, ağ sistemlerindeki hız ve bellek seçimlerinin paket çıkışını doğrudan etkilemesi yanında, ağ teknolojilerinin seçiminde yapılan tercihler iletişim protokollerinin / standartlarının seçimini kısıtlamaktadır. İletişim hata giderme düzenleri / mekanizmaları, ağın sıkışıklık durumlarını ve yük altındaki davranışlarını kontrol etmektedir. Bu kontrol işleminde kullanılan ve bütün düğümlere dağıtılan kontrol varlıkları ağın trafik yükünü nispeten artırmaktadır. Sistemlerin geliştirilmesinde büyük öneme sahip tasarım kararlarının verilebilmesi ve alternatiflerin araştırılması yapılan çalışmada ilgi odağındadır. Bu çalışmanın amacı, tasarım kararlarının dinamik davranış sonuçlarını araştırmayı olanaklı kılan bir modelleme ve

simülasyon çerçevesi meydana getirmektir. Geliştirilen çerçeve sistem tasarımcılarına yazılım nesnelerinin, donanım bileşenlerinin, ağ protokollerinin (özellikle biyolojik kökenli) ve büyük ölçekli ağlar gibi ileri ağ uygulamalarının çalışılmasını sağlamaktadır.

Bu çalışmada gerçekleştirilen ağ uygulamalarının tasarlanması ve uyarlanmasında, biyolojik prensipler ve mekanizmalar temel referans olarak alınmıştır. Geliştirilen ağ modelinin biyolojik sistemler gibi ölçeklenebilir, uyarlanabilir ve dayanıklı ağ uygulamalarının tasarımı için bir örnek çerçeve oluşturması amaçlanmaktadır. Dağıtık ağ sistemlerinin geliştirilmesinde bir yöntem / çözüm elde etmek ve tasarım alternatiflerini araştırmak amacıyla modelleme ve simülasyon araçlarının kullanıldığı bir uygulama gerçekleştirilmiştir. Bir başka değişle, donanım mimarilerinden bağımsız çalışan yazılım sistemlerini modelleme araçlarının geliştirilmesi, donanım altyapısının ağ hesaplama sistemleri olarak modellenmesi ve daha sonra dinamik bir sistem meydana getirmek üzere bu sistemleri bir araya getirilmesi işlemleri gerçekleştirildi.

Ağ ortamını modellemek için, ayrık olaylı sistemlerin sistem teorisi ve modelleme kavramlarını daha farklı ve özgün bir biçimde ifade eden *DEVS (Discrete Event System Specification)* modelleme ve simülasyon yöntemi, modellenen bileşenleri simüle etmek için *DEVSJAVA* modelleme ve simülasyon ortamı kullanıldı. Kullanılan DEVS modelleme yöntemi, sistem teorisi temellerini kullanması nedeniyle nesneye yönelik uygulamalar için elverişlidir [22]. Kullanılan DEVS yaklaşımının modüler ve hiyerarşik modeller kurmaya elverişli esnek yapısının sağladığı avantajlar, bilgisayar ağları gibi ayrık olaylı sistemlerin modellenmesinde sistem teorisi tabanlı bir metodoloji sağlamakta ve uygun arabirimli hiyerarşik modüllere sahip sistemleri tasarlamayı olanaklı kılmaktadır [23]. DEVS modelleme yaklaşımının matematiksel formatı, durum değişkenlerindeki değişimlere ve parçalı-sabit biçiminde olan grafiklerin üretilmesine odaklanır. Yöntem, durum değerlerinin üretilmesini ve yeni değerlerin etkin olduğu zaman aralıklarını tanımlar. Yaklaşımın önemli bir özelliği de, olayların olduğu zamanlar arasındaki zaman aralıkların değişken olmasıdır. Bu özellik DEVS yaklaşımını diğer klasik sistemlerden farklı kılmasına rağmen, DEVS sadece ayrık olaylı modeller için değil, aynı zamanda, ayrık zamanlı ve diferansiyel denklemli sistemlerle ifade edilen davranışları da

modelleyebilmektedir [24]. İleriki bölümlerde açıklayacağımız gibi, DEVS yaklaşımının paralel çalışan sistemler için uyarlanmasıyla ortaya çıkan *Paralel DEVS*, modern hesaplama teorisinde çok önemli bir yere sahip olan paralelliğin meydana getirilmesini sağlar ve bilgisayarların sıralı çalışan mimarileri nedeniyle ortaya çıkan sınırlandırmaları ortadan kaldırır.

1.3 Tezin Amacı

Bu çalışma, dağıtık bir ağ sistemini oluşturan bileşenlerin DEVS yöntemi kullanılarak tanımlanmasını, tanımlanan bileşenlerin davranışlarının detaylarının belirlenmesini, ortaya çıkan modelle bir takım örnek çalışmaların ve deneylerin yapılmasını kapsamaktadır. Bir ağ sisteminin modellenmesi işlemi; ağ bileşenlerinin tanımlanmasını, bu bileşenlerde çalışacak yazılım nesnelerinin ve nesneler arasındaki etkileşimlerin tanımlanmasını, yazılım nesnelerinin işlem yapan düğümlere dağıtılmalarını ve ağ topolojileri ile iletişim protokollerinin tanımlanmasını içerir. Ağın kontrolü ve yönetiminde aktif rol oynayan bir takım küçük, ağ içinde hareket edebilen varlıklara sahip olan düğümler ve bu düğümleri birbirine bağlayan bağlantılardan oluşan dağıtık ağ sistemlerinin modellenmesinde DEVS modelleme ve simülasyon yöntemi kullanıldı. Yönlendirme algoritmalarının test edilmesi, büyük ölçekli ağların analizi, modellenmesi, vb. ileri ağ uygulamalarını gerçekleştirmek amacıyla modellenen sistemi oluşturan parçalar ve bileşenler Java dilinde kodlandı. Daha sonra, DEVS *'birleşik model' (coupled model)* kavramı kullanılarak sistem bileşenleri birbirine bağlandı ve simülasyon deneyleri ile model davranışı üzerinde gözlem yapmak amacıyla *deneysel çerçeve* aracı kullanıldı. Deneysel çerçeve kavramı, modellerin analizi ve test edilmesinde gözlemlenecek simülasyon şartlarını tanımlayan bir araçtır [22]. Java dilinde yazılan DEVS metodolojisi kullanılarak DEVSJAVA modelleme ve simülasyon ortamında, özellikle ekoloji kökenli yönlendirme algoritmalarını çalışmak üzere, geliştirilen ortam *'SwarmNet'* olarak adlandırıldı.

Modellenen sistemi simüle etmek için DEVSJAVA modelleme ve simülasyon ortamı kullanıldı. Kolay tasarım ortamı ve anlaşılır arabirimiyle model tasarım ve eğitiminde etkili bir araç olan DEVSJAVA ortamı; nesneye-yönelik modellemeyi,

eş zamanlı paralel çalışan simülasyonları, etkileşen simülasyon nesneleri arasında uyumluluğu ve web tabanlı simülasyonları olanaklı kılmaktadır [22]. DEVS modelleme ve simülasyon yaklaşımının ve Java programlama dilinin sağladığı esneklik, değişen ortama adapte olabilen zeki bileşenlerin tasarımını kolaylaştırmaktadır. DEVSJAVA'nın nesneye yönelik yapısı, bir ağı oluşturan düğümlerin, linklerin, yazılım varlıklarının ve deneysel çerçevelerin modüler bir yapıda tasarımını, yeniden kullanımını ve sistemler sistemini oluşturmayı sağlamaktadır [25].

1.4 Tezin Kapsamı

Tez içerisinde yapılan çalışmalar dört grup altında özetlenebilir:

i- Farklı yönlendirme algoritmalarının incelenebilmesine olanak tanıyan bir ortam oluşturulması amacıyla örnek bir ağ modelinin DEVS (discrete event system specification) kullanılarak modellenmesi ve simülasyonu işlemleri gerçekleştirildi.

ii- Büyük ölçekli biyolojik sistemlerde (karıncalar, balarıları, termitler, vb.) kullanılan optimizasyon düzeneklerinden esinlenerek ağlarda kullanılacak kural-tabanlı yeni bir yönlendirme algoritması geliştirildi.

iii- Oluşturulan ağ modeline hali hazırda kullanılmakta olan yönlendirme algoritmaları ile çalışma sırasında geliştirilen biyolojik-tabanlı yönlendirme algoritması uygulanarak, özellikle biyolojik-tabanlı yönlendirme algoritmalarının klasik yönlendirme algoritmalarıyla karşılaştırılması yapıldı.

iv- Geliştirilen algoritmanın büyük ölçekli ağlarda kullanılabilirliğini göstermek amacıyla, çeşitli boyutlarda ağ modelleri oluşturularak kural-tabanlı algoritmanın performansı incelendi.

Yapılan çalışmada, '*SwarmNet'* modelleme ve simülasyon ortamının çalışmasını ve yönlendirme algoritmalarının bu ortamda gerçeklenmesini göstermek amacıyla bir takım örnek uygulamalar gerçekleştirilmiştir. İlk uygulamada, en yaygın kullanılan

klasik yönlendirme algoritmalarından biri olan *'yönlendirme bilgi protokolü' (Routing Information Protocol - RIP)* yönlendirme algoritmasının *SwarmNet* içinde modellenmesi ve simülasyonu işlemleri gerçekleştirildi. Uzaklık vektörü sınıfı yönlendirme algoritmalarının en yaygın kullanılanlardan biri olan RIP algoritması uygulanırken, *SwarmNet* ortamının klasik algoritmaları modelleme yeteneğinde olduğu gösterilmeye çalışıldı.

İkinci uygulamada; yüksek seviyeli bir dinamizme sahip olan bal arılarının nektar sahalarını ararken göstermiş oldukları davranışı esin kaynağı olarak alan bir oğul zekası yönlendirme yönteminin geliştirilmesi, modellenmesi ve simülasyonu işlemleri gerçekleştirildi. Daha öncede ifade edildiği gibi, *SwarmNet* ortamının temel geliştirilme hedeflerinden biri, yeni nesil yönlendirme algoritmalarının araştırılması ve geliştirilmesidir. Balarılarının davranışlarından kural-tabanlı, dağıtık (çok merkezli) ve hatalara karşı sağlam bir yönlendirme algoritması türetilerek, biyolojik kökenli yaklaşımların dağıtık sistemlerdeki üstünlükleri gösterildi. Üçüncü uygulamada, gerçekleştirilen ilk iki uygulamadan elde edilen sonuçlar, merkezi olmayan ve dağıtık yaklaşımların üstünlüklerini göstermek amacıyla karşılaştırıldı.

Gerçekleştirilen son uygulamada; DEVSJAVA ortamının yüksek performansı kullanılarak, *SwarmNet* ortamının kapasitesini belirlemek ve DEVS yaklaşımının paralel / dağıtık uygulamalardaki gücünü göstermek amacıyla büyük ölçekli ağlar incelendi ve topoloji analizi yapıldı. Büyük ölçekli ağları incelemede başlıca hedeflerimizden birisi; İnternet türü ağların analizi ve yönetimidir. Büyük ağların modellenmesinde düğüm başına artan yönlendirme veritabanını azaltmak için arılardan esinlenerek özel bir '*kümeleme' (clustering)* yöntemi kullanıldı.

Yapılan çalışmaların ve gerçekleştirilen uygulamaların katkısı, dağıtık sistemlerin karmaşık dinamiklerinin ve etkileşimlerinin modellenmesi ve simülasyonu için pratik bir yöntemin sunulması şeklinde özetlenebilir. Bu çalışmanın, geliştirilen araçlarla birlikte modelleme ve simülasyon eğitimine katkı yapması yanında, ağ bileşen davranışlarının tasviri ve simülasyonunda DEVS yaklaşımının elverişliliğini göstereceği düşünülmektedir. DEVS hiyerarşik ve modüler tasarım yaklaşımıyla, karmaşık sistemlerin davranışlarını modellemeyi, yönetilebilir model ailelerinin meydana getirilmesini ve modellerin yeniden kullanımını mümkün kılmaktadır.

DEVS metodolojisi üzerine kurulan ve Java dilinde geliştirilen ağ simülatörü, temel ağ yönetimi sistemlerinin, yük dengeleme yaklaşımlarının ve tıkanıklık kontrol yapılarının çalışılmasını olanaklı kılmaktadır. Ayrıca simülatör aracılığıyla farklı topolojileri test etmek mümkündür.

Yapılan çalışmadan elde edilen sonuçlar, kolektif zekaya sahip biyolojik sistemlerdeki optimizasyon mekanizmalarının geleceğin ağlarının problemlerini çözmede önemli bir role sahip olacağını göstermektedir. Büyük ölçekli biyolojik sistemlerin en önemli özelliklerinden bir ölçeklenebilirliktir. Karıncalar ve balarıları gibi biyolojik sistemler çok büyük koloni boyutuna ulaşabilirler ve boyutları artarken performansları düşmemektedir. Büyük ölçekli ağların meydana getirilmesinde ve yönetiminde biyolojik kavramların kullanıldığı çalışmalarda gerçekleştirilen deneylerden, geliştirilen ağ simülatörünün birkaç bin mertebesinde bileşene sahip ağların davranışlarının çalışılabilmesi için son derece elverişli bir yapıda olduğunu göstermektedir.

1.5 Tez Planı

Giriş bölümünde yapılan çalışmaların ilişkisi olduğu konular kısaca tanımlandıktan ve yapılan çalışmalar genel hatlarıyla özetlendikten sonra çalışmaların sunulduğu tezin içeriği ile ilgili bilgiler verilmektedir.

2. bölümde, modelleme ve simülasyon teorisi ve metodolojisinin altında yatan bazı temel kavramların açıklanması ve konu hakkında temel bir anlayış oluşturulması hedeflemektedir. Modelleme ve simülasyon aktivitelerinin daha planlı ve sistematik bir şekilde gerçekleştirilmesi amacıyla bir sistemi modellerken yapılması gereken süreç tanımlanmakta ve bu süreçteki aşamalar ayrı ayrı incelenmektedir Bu süreçte önemli aşamalardan biri modellerin doğrulanması ve geçerlenmesidir. Modelin sistem karşılığına kabul edilir bir uygunlukta olup olmadığının denetlenmesi işlemidir. Daha sonraki aşamada, sistemlerin davranışını matematiksel bir biçimde tanımlamaya yarayan modelleme yöntemlerinin ve soyutlama seviyelerinin belirlenmesi işlemleri gerçekleştirilmektedir. Modelleme yöntemleri, nesneye yönelik yaklaşımlar, ayrık sistemlerde zaman kavramı, vb. konular modelleme ve

simülasyon alanının önemli ve hassas alanlarıdır. Son yıllarda oldukça ilgi gören sistemleri birden fazla modelleme ve simülasyon yaklaşımı kullanarak modelleme düşüncesi, temel çıkış noktalarıyla birlikte bu bölümde detaylandırılmaktadır.

3. bölümde, 2. bölümde temelleri atılan ayrık olaylı sistemler için Ayrık Olaylı Sistem Tanımı (Discrete Event System Specification - DEVS) yaklaşımı tanımlanmaktadır. Bu bölümde ilk olarak ayrık olaylı sistemleri tanımlayan kavramları ele alınmakta, daha sonra ayrık olaylı sistemleri farklı değerlendiren dünya görüşleri açıklanmaktadır. Zeigler tarafından ortaya atılan DEVS yaklaşımı tanımlandıktan sonra, atomik ve birleşik DEVS kavramları tanımlanmaktadır. DEVS'in sadece ayrık olaylı modeller için değil, aynı zamanda, ayrık zamanlı ve diferansiyel denklemli sistemlerle ifade edilen davranışları uyarlamak için bir hesaplama temeli oluşturması veya DEVS'in evrenselliği detaylandırılmaktadır.

4. bölümde, tezde sunulan çalışmanın uygulandığı problem alanını temsil eden dağıtık sistemler ile yönlendirme algoritmaları verilmektedir. Paralel ve dağıtık sistemlerin genel anlayışı ve bu sistemler arasındaki temel farklılıklar kısaca ele alınmakta, ağ kontrol ve izleme işlevlerinin merkezi olması yerine çok merkezli / dağıtık olmasının gerekliliği / üstünlükleri açıklanmaktadır. Ağ yönetimi ve kontrolünde yeni bir araştırma alanı olan büyük ölçekli biyolojik sistemlerin (arılar ve karıncalar gibi) ve bu sistemlerdeki önemli optimizasyon yapılarının ağlara uygulanması ile bu alanda daha önce yapılmış çalışmalar bu bölümde detaylandırılmaktadır.

5. bölümde, dağıtık bir ağ sistemini oluşturan bileşenlerin DEVS kullanılarak tanımlanması ve tanımlanan bileşenlerin davranışları açıklanmaktadır. Bu açıklamalar yapılırken, SwarmNet ortamının modellenmesi ve simülasyonunun safhalarını oluşturan modelleme süreci tanımlanmakta, daha sonra SwarmNet ortamını meydana getiren bileşenler ve bileşenleri oluşturan bir takım nesneler / kavramlar detaylandırılmaktadır. Yönlendirme algoritmalarının test edilmesi, büyük ölçekli ağların analizi ve test edilmesi gibi çeşitli ileri ağ uygulamalarını gerçekleştirmek amacıyla sistemi oluşturan parçalar ve bileşenler bir ağ simülatörü meydana getirmek üzere JAVA dilinde kodlanmaktadır. Sunulan teorik bilgi

DEVJAVA ortamından alınan ekran çıktılarıyla desteklenmektedir. Bu bölümde tanımlanan bileşenlerin Java kodları ekte verilen CD'de bulunabilir.

6. bölümde, *SwarmNet* modelleme ve simülasyon ortamının çalışmasını ve yönlendirme algoritmalarının bu ortamda gerçeklenmesinin göstermek amacıyla çeşitli örnek çalışmalar sunulmaktadır. Bunlardan birincisi; en yaygın klasik yönlendirme algoritmalarından olan RIP yönlendirme algoritmasının modellenmesi, ikincisi ise; yüksek seviyeli bir dinamizme sahip bal arılarının nektar sahalarını ararken göstermiş oldukları davranışı esin kaynağı olarak alan bir oğul zekası yönteminin uygulanması ve modellenmesidir. Yapılan çalışmada, DEVS modelleme ve simülasyon yaklaşımı kullanılarak dağıtık ağ sistemleri ile bu sistemlerde çalışan yönlendirme algoritmalarının modellenmesinde geliştirdiğimiz ortamın gerçek bir ağ simülatörü gibi çalışıp çalışmadığı test edilmektedir. Daha öncede ifade edildiği gibi, böyle bir ağ modelleme ve simülasyon ortamının geliştirilmesinde amaç, ekolojideki sosyal canlıların (karıncalar, bal arıları, vb.) davranışlarından türetilen yönlendirme algoritmalarının çalışılmasına olanak tanıyan bir modelleme ve simülasyon ortamının DEVS modelleme ve simülasyon yöntemi kullanılarak geliştirilmesiydi. Bu bölümde, bu konular ile ilişkili olarak gerçekleştirilen örnek çalışmalar sunulmaktadır. Sunulan çalışmada, bal arıların 'keşfet-yönlendir' sisteminden uygun bir soyutlamayla kural tabanlı bir yönlendirme algoritmasının geliştirilmesi ve modellenen ağa uygulanması detaylandırılmaktadır. Geliştirilen deneysel çerçeve yoluyla üretilen çıkışlar grafikler halinde sunulmakta ve her iki uygulamadan elde edilen çıkışlar karşılaştırılmaktadır. Bu uygulamaların yanında, DEVS yaklaşımının paralel ve dağıtık uygulamalardaki gücü ile '*SwarmNet'* modelleme ve simülasyon ortamının yüksek performansı kullanılarak topoloji analizi ve büyük ölçekli ağlar incelenmesi ile ilgili sonuçlar verilmektedir.

Sonuç ile tartışma ve öneriler kısmında, elde edilen sonuçlar özetlenmekte ve ileriye yönelik olarak bu konuda yapılabilecek çalışmalara ışık tutacak öneriler verilmektedir.

BÖLÜM 2. MODELLEME ve SİMÜLASYON TEORİSİNE GİRİŞ

2.1 Giriş

Birçok mühendislik alanında gerçekleştirilen analiz ve tasarım işlemlerinde bilerek veya bilmeyerek modelleme ve simülasyon yöntemleri kullanılmaktadır. Genel olarak, modelleme ve simülasyon çoğu kez Sistem Teorisi, Kontrol Teorisi, Sayısal Analiz, Bilgisayar Bilimleri, Yapay Zeka ve Yöneylem Araştırmasının bir alt kümesi olarak görülmektedir [22]. Modelleme ve simülasyon gittikçe artan bir şekilde yukarıda ifade edilen tüm bilim dallarını içine almaktadır. Son zamanlarda, modelleme ve simülasyon geleceğin bir hesaplama yaklaşımı olmaya doğru yönelmiştir. Özellikle modelleme ve simülasyon teorisiyle uğraşan bilim adamları ve araştırmacılar, modelleme ve simülasyonun bütün mühendislik sistemlerine temel teşkil eden matematik bilimi gibi genel ve bu alanda kullanılan kavramların / yöntemlerin herkes tarafından kabul görmüş bir şekilde ortak olmasına gayret göstermektedirler [24].

Bir hesaplama yöntemi olarak modelleme ve simülasyon, bir problem çözüm yöntemi veya problemleri tanımlama ve problemler hakkında fikir yürütme yolu / yöntemi olarak açıklanabilir. Problemler, en basit sistemlerden en karmaşık sistemlerin tasarımını ve analizini içerecek şekilde geniş bir alana yayılabilir. Problemlerin analizinde, soyut modeller (kavramsal model) bir gerçek sistemin gözlemlenmesi yoluyla oluşturulur / tasarlanır. Modellerin oluşturulmasında, modellenen sistemle ilgili temel bir ön bilgiden mantıksal sorgulama yoluyla türetilen bilgilerden faydalanılarak belirli bir tasarım hedefine yönelik bir sistemin meydana getirilmesine çalışılır. Gerçek problemleri çözerken çoğu durumda analiz ve tasarımın birleştirilmesine gereksinim duyulur.

Modelleme ve simülasyonun odağında dinamik (zamanda değişen) sistemlerin davranışı olsa da, sınırlı oranda statik sistemlerde (Birleşik Modelleme Dilinde -

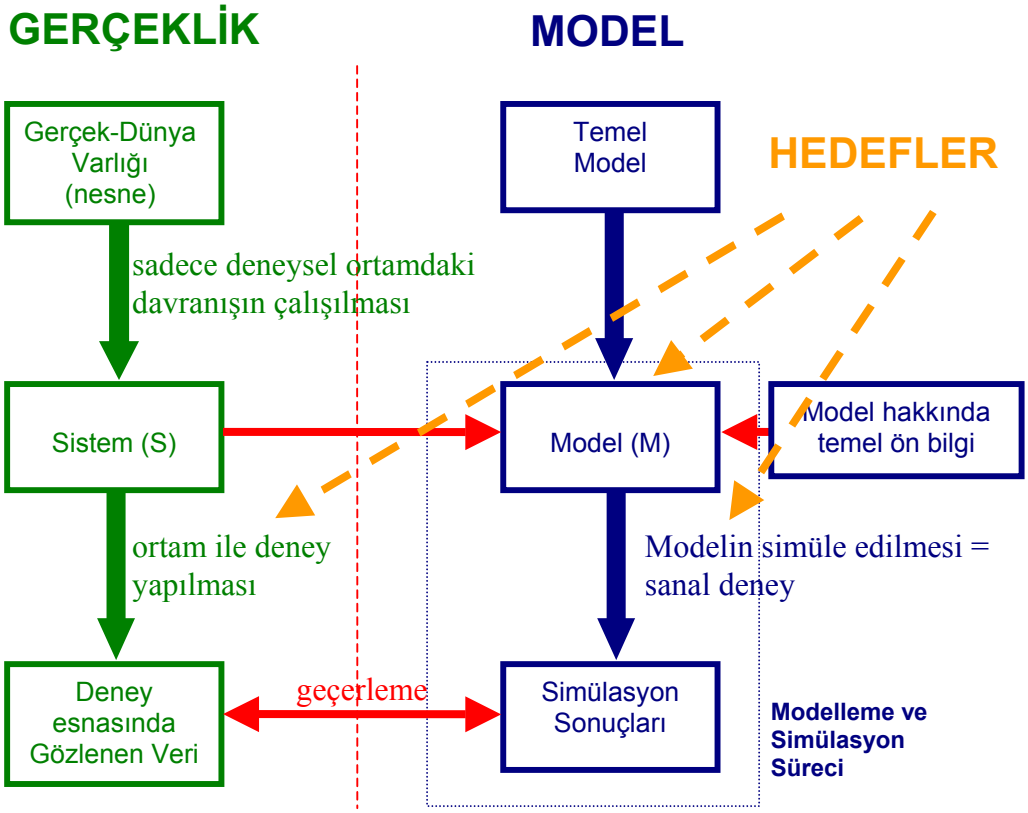
UML- kavramsal olarak tanımlı varlık ilişkili modeller gibi) kullanılmaktadır [59]. Hem fiziksel (kuralları muhafaza etme ve onlara uymaya zorlamaya bağlı olarak) hem de fiziksel olmayan sistemler (yazılım gibi bilgisel) ile birlikte sistemlerin etkileşimleri, modelleme ve simülasyon aracılığıyla öğrenilebilmektedir.

Modelleme / model geliştirme ve simülasyon, yazılım (nesneye-yönelik programlama) ve donanımdaki (daha hızlı işlemciler) teknolojik ilerlemeler sonucunda daha kolay ve hızlı bir şekilde gerçekleştirilebilmektedir. Ancak, model güvenilirliği (geçerleme, doğrulama ve model aile tutarlılığı), karşılıklı çalışabilirlik (interoperability) ve model kütüphaneleri yoluyla model parçalarının yeniden kullanılması (reusability), vb. temel konular araştırmacılar tarafından çok az ilgi çekmiştir. Ancak bu problemler yüksek seviyeli yapı (High Level Architecture – HLA) konusu altında ele alınarak çözüme kavuşturulma yönünde ilerlemektedir [26].

Bu bölümde temel kavramlarla birlikte, modelleme ve simülasyon sürecinin aşamaları tek tek ele alınmaktadır. Daha sonra, modelleme ve simülasyon yaklaşımları (formalisms - biçimsel sistem tanımlamaları) konusu işlenmekte ve çoklu-formalizm kavramı kısaca açıklanmaktadır. Son olarak, bütün modelleme yaklaşımlarının DEVS yaklaşımına dönüştürülebileceği şekillerle izah edilmektedir.

2.2 Temel Kavramlar

Modelleme ve simülasyon teorisi ile ilgili temel kavramlar aşağıda sunulmaktadır. Şekil 2.1'de, Zeigler tarafından ortaya atılan modelleme ve simülasyon kavramları ve bu kavramların birbirleriyle ilişkileri tanımlanmaktadır [22][23]. Burada, gerçek dünya ve modellerimizi tasarladığımız sanal dünya kesikli bir çizgiyle ayrılmıştır. Bu gösterim modelleme ve simülasyon sürecinin sistemlerin tasarımındaki konumunu göstermesi açısından önemlidir.



Şekil 2.1 Modelleme ve simülasyon kavramları ve birbirleriyle ilişkileri [24].

Nesne, gerçek dünyadaki bir varlıktır ve çalışıldığı ortama bağlı olarak son derece değişken bir davranış sergiler.

Temel Model, özellikle tüm olası durumlar için geçerli olan davranış gibi bir takım nesne özelliklerinin soyut bir açıklamasıdır ve nesnenin dış görünüşünün tamamını tanımlar. Pratikte bir nesnenin "bütün" bir modeli kurulamaz ve tanımlanamaz ise temel model farazidir, yani teorik olur. Temel modelin var olup olmayacağı sorusu felsefi bir sorudur (fizikteki gizli değişken problemi gibi) [24].

Sistem, sadece yapı ve davranış açısından dikkate alınan belirli şartlar altında gerçek dünyada iyi tanımlanmış bir nesnedir.

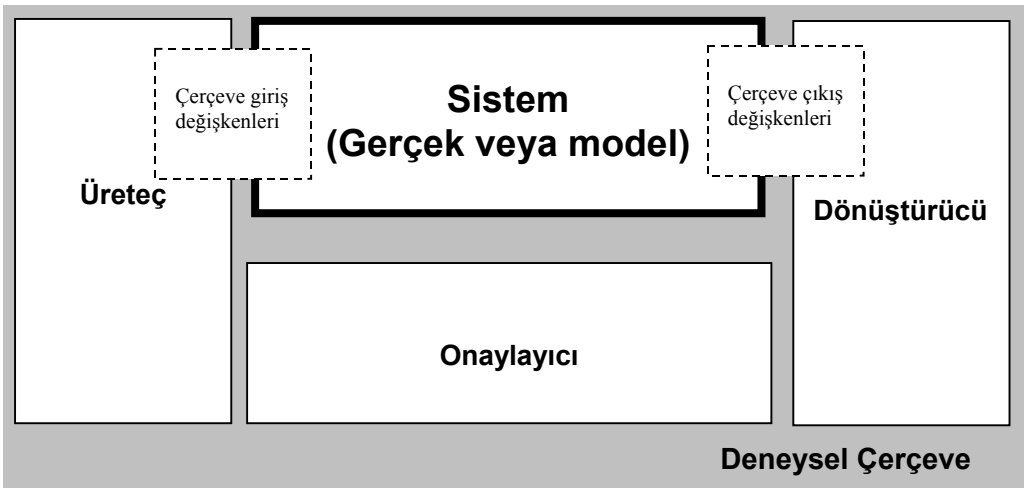
Deneysel Çerçeve, herhangi bir sistemle gerçek dünyada çalıştığımızda, deneysel çerçeve (experimental frame - EF) sistem ve sisteme karşılık düşen modellerin çalıştırılacağı deneysel şartları (ortamları tarif eder. Diğer bir değişle deneysel

çerçeve, gerçek bir sistem veya bir simülasyon aracılığıyla model üzerinde deneyler yapan kişinin / modelleyicinin hedeflerini yansıtır.

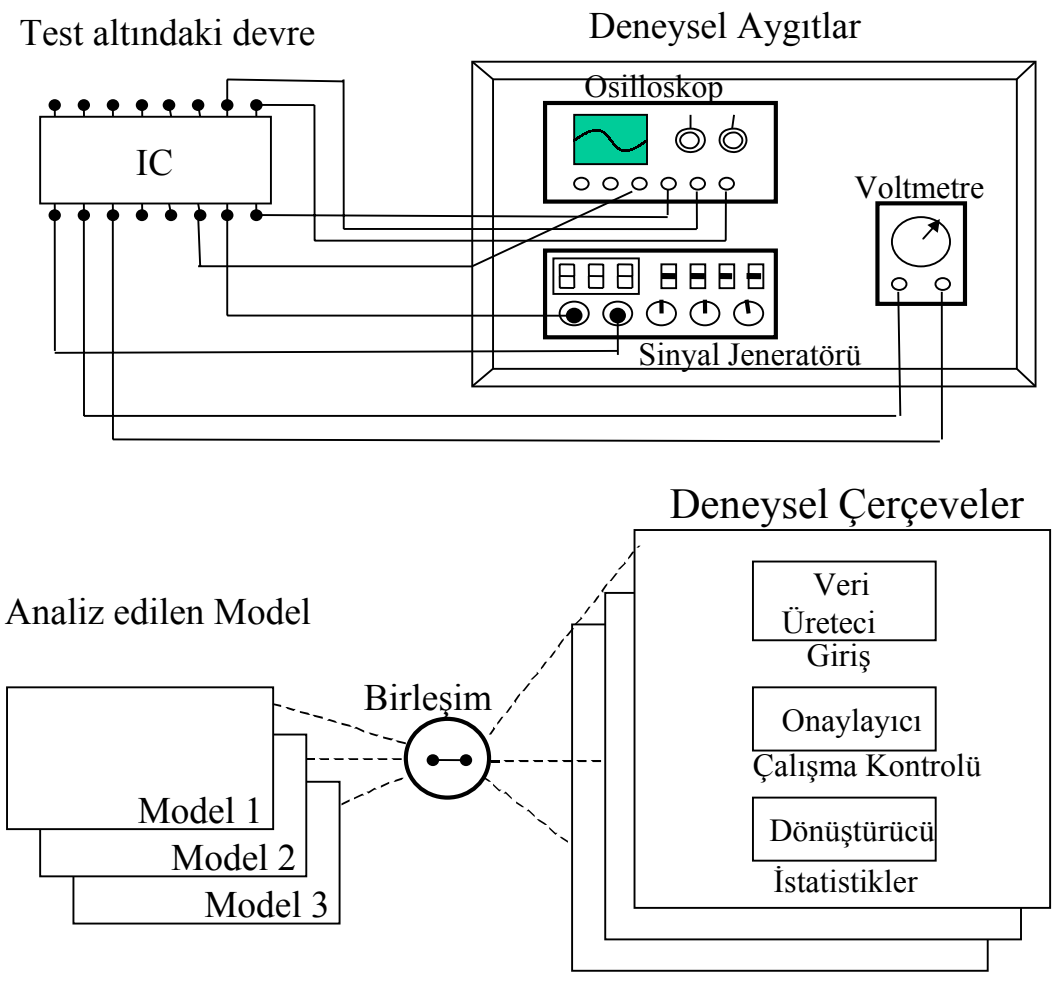
En genel şekliyle bir deneysel çerçeve iki adet değişkenden oluşur: Sistemin veya modelin giriş ve çıkış uçlarıyla bağlanan Çerçeve Giriş Değişkenleri ve Çerçeve Çıkış Değişkenleri (Şekil 2.2). Giriş değişken kısmında bir *üreteç,* giriş veya deney esnasında sisteme / modele uygulanacak olan bir uyarıcı vazifesi görür. Üreteç, bir birim basamak girişi üretebilir. Çıkış değişken tarafında bağlı bir *dönüştürücü* (transducer), sistemden gelen sonuçlar hakkında mantıklı bir yorum yapmak için sistemi (deney) veya model (simülasyon) çıkış değerlerine uygulanacak olan dönüşümleri tanımlar. Dönüştürücü, çıkış değişkenlerden bir kısmının değerlerinin hesaplanması için kullanılabilir. Çıkış kavramı ile hem fiziksel sistem, hem de gözlemci tarafından ölçülen model içi durumlar biçimindeki yapay değerler kastedilmektedir. Çıkışlar, bir modelde durum değişkenleri ve parametreler şeklindeki bilgiyi içerebilir. Üreteç ve dönüştürücü, giriş / çıkış değişkenleri ile birlikte deneysel çerçeve içindeki üreteç girişleri ile dönüştürücü çıkışlarını karşılaştıran bir *onaylayıcıdan* (acceptor) oluşur. Onaylayıcı, sistemin (gerçek veya model) deneysel çerçevesinin deney yapanın hedefleriyle uygun olup-olmadığını belirler. Bu üç bileşen işlevleri bakımından bir elektronik devre deneyindeki bir sinyal üreteci, spektrum analizörü ve bir osiloskop benzer. Şekil 2.3 bu benzerliği göstermektedir.

Model, belirli bir deneysel çerçeve ortamı içerisindeki sistemin belirli bir soyutlama seviyesindeki tarifidir / tanımıdır. Bir sistemin yapısının ve / veya davranışının belirgin özellikleri belirli bir doğruluk dahilinde model tarafından yansıtılmalıdır.

Deney, bir sistemi fiziksel olarak test etme işlemidir ve sistemin çalışmasını etkileyebilir (giriş ve parametrelerini etkileyerek). Deney ortamı ayrı bir sistem olarak görülebilir ve dolayısıyla birleşik bir model yoluyla ayrıca modellenebilir. Deney gözlem işlemini içerirken, gözlem ölçümler sonucunda değerlere ulaştırır.



Şekil 2.2 Sistem ve deneysel çerçevenin yapısı.



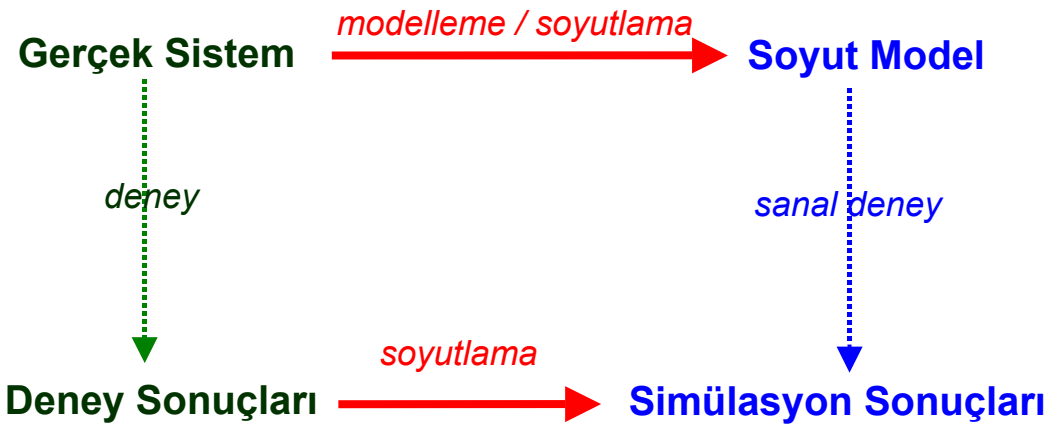
Şekil 2.3 Elektronik devreler deneyi ile simülasyon deneyi arasındaki benzerlik

Simülasyon, belirli bir modelleme yaklaşımında (DEVS, Petri Net, Diferansiyel Eşitlikler, Bond Grafikleri, vb) tanımlanan bir modelin, simülasyon sonuçlarını (dinamik giriş / çıkış davranışı) üretir. Bir gerçek dünya deneyini taklit eden simülasyon, bir sistemin davranışı ile ilgili sorulara cevap vererek sanal bir deney yapma işlemi olarakta görülebilir. Simülasyon hem sembolik hem de nümerik teknikleri kullanabilir. Modellemenin hedefi bilgiyi sunan sistemi anlaşılır ve yeniden kullanılabilir bir şekilde mantıksal olarak tarif etmek iken, kullanılan tekniğin önemli olmadığı simülasyonun amacı; olabildiğince hızlı olmak ve modelin işlevlerini doğru bir şekilde yansıtmaktır. Sembolik teknikler tek bir çözümden daha çok çözümler sınıfının oluşmasını sağlamaları nedeniyle, çoğu kez sayısal tekniklere göre daha çok tercih edilmektedir. Örneğin, bir harmonik denklem için çözüm olarak *sin(x)* fonksiyonu, yaklaşık bir eğriye göre daha çok tercih edilir. Ayrıca sembolik optimizasyonlar doğaları sayesinde nümerik çözümlerden daha büyük bir etkiye sahiptirler. Model ve sistem arasında *eşbiçimli (homomorphic)* bir bağıntının olması Sistem – Deney / Model – Sanal Deney düzeneği için çok önemlidir: gerçek bir sistemin modelini geliştirmek ve geliştirilen modelin davranışını simüle etmek, deneysel sonuçları gözlemlemeyi ve sistematik bir şekilde düzenlemeyi takiben yapılan gerçek bir deneyle aynı sonucu verir (Şekil 2.4). Bir simülasyon modeli herhangi bir hedefe (tasarım, analiz, kontrol, optimizasyon vs.) ulaşmak için bir araçtır. Bu nedenle temel bir ön şart, modelleme ve simülasyon (araçları) aracılığıyla yapılan çıkarımların/elde edilen sonuçların rahatlıkla kabul edilebilmesinin güvence altına alınmasıdır. Bu güvenin tesis edilmesi için iki ayrı çalışma gerçekleştirilir: *doğrulama (verification)* ve *geçerleme (validation).*

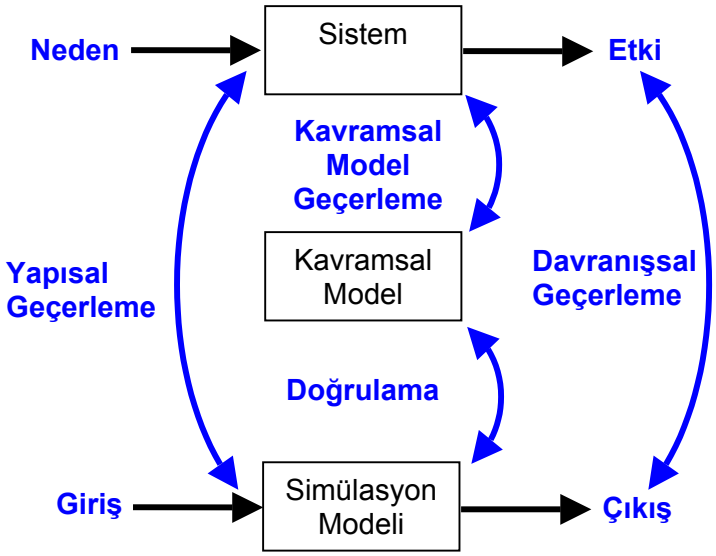
Doğrulama, bir simülasyon programının tutarlılığını türetildiği modele göre kontrol etme işlemidir. Diğer bir değişle doğrulama; yazılan program kodunun, kavramsal modelin tanımında dolaylı olarak bulunan davranışı tam olarak yansıtmasını garanti altına alarak, soyut tasvirden / tanımlamadan (kavramsal model) program koduna (simülasyon modeli) geçişin doğruluğuyla ilgilenir [27].

Geçerleme, belirli bir deneysel çerçeve ortamında elde edilen deneysel sonuçların simülasyon sonuçlarıyla karşılaştırılması işlemidir [27]. Karşılaştırma sonucunda farklılıklar ortaya çıkarsa, biçimsel olarak geliştirilen model gerçek sisteme karşılık düşmeyebilir.

Çeşitli geçerleme türleri bulunmaktadır: *kavramsal model geçerleme*, *yapısal geçerleme*, *davranışsal geçerleme, vb.*. *Kavramsal geçerleme*; kavramsal modelin sisteme göre değerlendirilmesidir. Burada amaç, çalışma hedeflerine göre kavramsal modelin gerçekliğinin değerlendirilmesinin yapılmasıdır. *Yapısal geçerleme*; algılanan sistem yapısına göre bir sistemin model mimarisinin değerlendirilmesidir. *Davranışsal geçerleme;* simülasyon modelinin davranışının değerlendirilmesidir. Doğrulama ve geçerleme aktivitelerinin bir özeti Şekil 2.5'te görülmektedir. Burada, bir sistem ve o sistemin modeli arasında üretilen davranış açısından uygunluk sadece sınırlı bir deneysel çerçeve içinde değerlendirilecektir. Diğer taraftan, ölçümlerin ve simülasyon sonuçlarının büyük oranda örtüşmesi simülasyona olan güveni artırmasına rağmen modelin geçerliliğini ispatlamaz. Bu sebepten dolayı, tahrif (falsification) kavramı ortaya atılmıştır [28]. Tahrif, tam olarak bir modeli çürütmek ve bozmak işlemidir. Sonuç olarak, bilgiyi karşılıklı olarak değiştiren modeller kullanıldığında, modelin deneysel çerçeveye uyumlu olmalı ve bir model kendi deneysel çerçevesiyle eş zamanlı olarak tasarlanmalıdır. Bu gereksinim bir model tasarım dilinin geliştirilme gereksinimini ortaya çıkarmıştır.



Şekil 2.4 Modelleme – simülasyon dönüşümü.



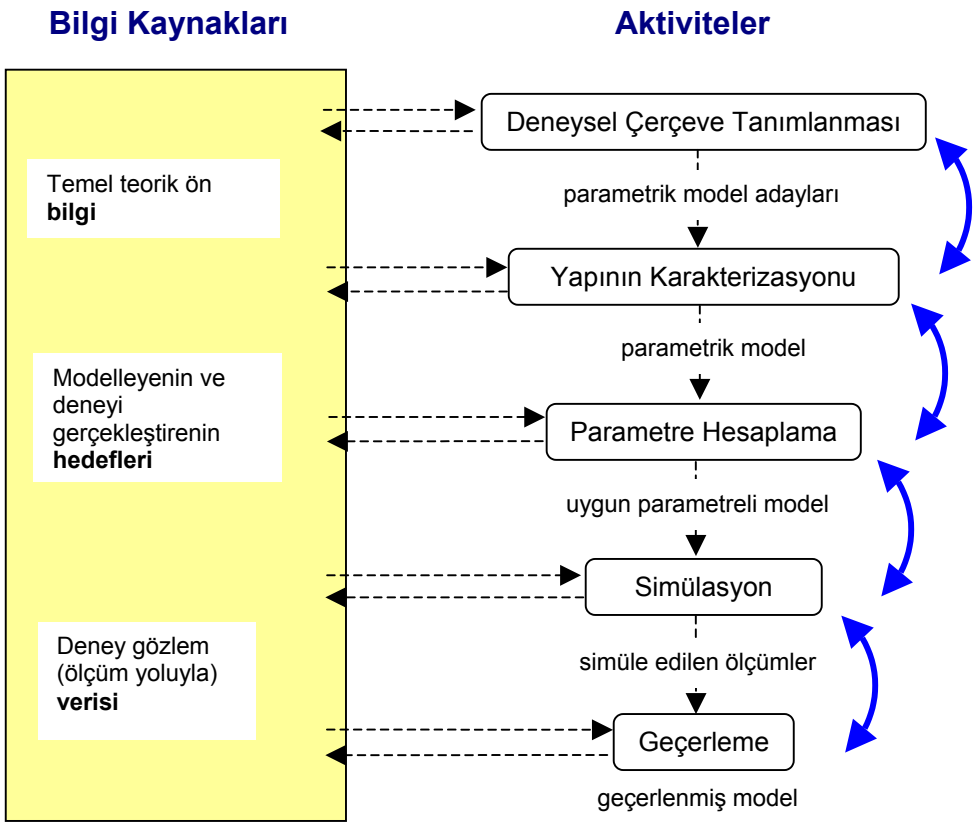
Şekil 2.5 Doğrulama ve geçerleme aktiviteleri [17].

2.3 Modelleme ve Simülasyon Süreci

Modelleme ve simülasyon aktivitesini anlamak için gerçekleştirilen aktivitelerin, çalışacak varlıkların, nedensel ilişkilerinin (aktivite sırasını ve zamanda uyumluluğun belirlenmesi), vb. işlemlerin / faktörlerin analiz edilmesi gerekir. Yapılacak işlemin veya sürecin tanımlanması, sistem mantığının kavranmasına yardımcı olması ile birlikte, önerilecek çalışma taslağı ile yazılım aracının otomasyonuna / uyarlanmasına temel teşkil edebilir. Alınacak kararların büyük çoğunluğunun kullanıcıya bırakılması nedeni ile, önerilen işlem '*deterministik*' yani belirleyici değildir. Simülasyon işlemi, büyük ölçekli model-tabanlı sistemlerin analizinde bir aşamadır ve bu aşamada gerçekleştirilen aktivitelerin temel işleyiş modeli Şekil 2.6'da görülmektedir.

Modelleme ve simülasyon işlemleri sırasında veri / model / bilgi şeklinde açık bir halde veya tasarımcının kafasında kapalı halde bulunan *bilgi kaynakları* kullanılır (Şekil 2.6). Kullanılan bilgi kaynakları aşağıdaki şekilde özetlenebilir;

***i- Temel ön bilgi*:** Tümdengelim (deductive) yöntemi ile gerçekleştirilen modellemede, genel prensiplerden (kütle, enerji, momentum koruma kanunları, sınırlamalar, vb.) faydalanılabilir ve bu yolla mantıksal belirli bir bilgi ortaya çıkarılır. Ortaya çıkarılan bilgi genellikle sistemin tasarlanmasında kullanılır.



Şekil 2.6 Model-tabanlı sistemlerin analizi [14].

***ii- Hedefler ve niyetler*:** Modelleme hedefleri, modelin sistemlerin tasarımındaki, yönetimindeki ve kontrolündeki rolüyle ilişkilidir. Hedefler ifadesi, model geliştirme işlemini belirli meseleler üzerine odaklama vazifesi görür [22]. Bu nedenle, hedeflerin model geliştirme işleminde mümkün olduğunca erken ortaya konması çok önemlidir. Hedefler üzerinde sağlam bir birliktelik, proje liderlerine takımın işleyişi üzerinde kontrol yapabilmelerine olanak tanır. Hedeflerin önceden belirlenmesi ve net olarak ifade edilmesi, hedeflere uygun deneysel çerçeveler geliştirilebilmesini ve ortaya çıkan soruların cevaplandırılmasında yüksek seviyede bir kararlılığın oluşmasını kolaylaştırır. Kullanılan soyutlama seviyesi, modelleme ve simülasyon yaklaşımları ve yöntemleri, cevaplanmasını istediğimiz sorular yoluyla belirlenir. Daha çok sayıda sorunun talep edilmesi, bu soruları cevaplandırmak için ihtiyaç duyulan daha büyük bir kararlılığın belirtisidir. Böylece, uygun soyutlama düzeylerinin seçimi, aynı zamanda, hedeflere ve kendi deneysel çerçeve karşılıklarına bağlıdır .

***iii- Ölçülen veri*:** Tümevarım yöntemi ile gerçekleştirilen modelleme işlemine veriyle başlanır ve böylece yapı ortaya çıkarılmaya çalışılır. Bu yapı / model, sonuçta tümden gelen bir tarzda kullanılabilir. Bu yöntem sistemlerin analizinde yaygın olarak kullanılır.

Şekil 2.6'da gösterilen model tabanlı sistemlerin analizindeki aktiviteler / işlemler, deneysel çerçevenin tanımlanmasıyla başlar. Daha öncede ifade edildiği üzere, deneysel çerçeve modelleyicinin sistemi araştırmak istediği deneysel şartları gösterir ve modelleyicinin hedefleri ile cevap almak üzere sisteme yönelttiği soruları yansıtır. Deneysel çerçeve en genel biçimi ile; sisteme muhtemel girişleri tanımlayan bir '*üreteç'*, çıkış işlemini tarif eden bir '*dönüştürücü'* (çıkışla bütünleştirilen performans ölçümlerinin hesaplanması gibi) ve sistemin (onu gerçekleyen veya modelleyen) eşleştiği durumları (mantıksal ifadeler) tanımlayan '*onaylayıcıdan'* oluşur (Şekil 2.3). Bu çerçeve yoluyla, ortaya çıkan sonuçlara uygun modeller sınıfı tanımlanabilir.

Deneysel çerçeve tanımlandıktan sonra, tanımlanan deneysel çerçeveye uygun bir model yapısı ve model parametreleri tanımlanır. Model karakterizasyonu yoluyla, uygun model yapısı temel bilgiye ve ölçülen veriye dayanılarak seçilir. Bu işlem sonucunda ortaya değişken değerlere sahip yani parametrik bir model çıkar.

Model parametrelerinin kalibrasyonu esnasında, parametre hesabı gibi bir takım işlemler ölçülen veriyi üretmek için uygun parametre değerlerini tayin eder. Bu parametreler yoluyla modelden bir simülatör üretilir. Modellemenin (anlaşılır ve yeniden kullanılır bir model gösterimi) ve simülasyonun (doğruluk ve hız gibi) hedeflerinin farklılık teşkil etmesi modelden simülasyona geçişte bir problem oluşturabilir ve bu sorunu minimize etmek için çok sayıda adım atılması gerekebilir. Tanımlı model ve parametreleri kullanan simülasyon, sistemin davranışını (sanal deney) taklit etmeyi olanaklı kılar. Bu nedenle sonuçta ortaya çıkan simülasyon / simülatör, optimizer gibi eğitici veya öğretici bir araçla birleştirilebilir.

Bu süreçte sorun, simüle edilen ölçümlerin istenen aralıkta olup olmadığı veya diğer bir ifadeyle modelin *geçerli* olup olmadığıdır. Geçerlemeyi ispat etmek için "model,

model seçiminde ve parametre hesabında kullanılan veriyi tekrar üretebilme ve model tanımında olmayan yeni bir davranışı önceden kestirebilme yeteneğinde midir?" sorusunu sormak gereklidir. Simülatörün her kullanımında bu geçerleme sorusu sorulmalıdır. Kullanıcı, geçerlemenin süreçte hangi aşamada olacağını ve süreçte içerilip içerilmeyeceğini belirler. Örneğin, bir uçuş simülatöründe, modelin başlangıçta geçerlenmiş olmasının beklenmesinin yanında bir ders esnasında (kullanıcı yoluyla geçerleme), eğitim işleminin bir parçası olarak geçerlenmesi yapılabilir.

Şekil 2.6'da detaylandırılan modelleme sürecinde her bir adımda, ortaya çıkan hataların geriye dönülerek giderilmesi işlemi, geribesleme oklarıyla gösterilmiştir. İşlem bozulduğunda veya hata oluştuğunda, şekilde oklarla gösterildiği gibi, geriye dönülerek düzeltme işlemi gerçekleştirilebilmelidir. Geçerleme işleminin önemli bir özelliği, modelleme hatalarının konumu ile ilgili olarak ipuçları sağlamasıdır. Bununla beraber, çok az yöntem böyle bir bilgiyi sistematik olarak sağlamak için tasarlanmıştır. Pratikte, süreç mükemmelleştirerek daha genel süreçlere dahil edilebilir (örneğin; eğitim, öğretim, optimal deneysel tasarım ve kontrol gibi). Model doğrulama ve geçerleme takip eden bölümlerde daha detaylı ele alınacaktır.

2.4 Doğrulama ve Geçerleme

Doğrulama, bir simülatör ve model arasında simülatör doğruluk ilişkisinin tesis edilmesi gayretidir [27]. Simülatör bir model ailesini / grubunu çalıştırma kapasitesinde olmalıdır. Bu durumda, modellerin herbirinin doğru bir şekilde çalışacağını garanti altına almak gerekir. Doğruluğu sağlamada veya tesis etmede, temel bir kavram '*eşbiçimliliktir'*. Bunun anlamı, bir modelin *a, b, c, d* gibi bir durum dizisi izlemesi durumunda, simülatöründe benzer bir A, B, C, D durum dizisi izlemesi gerektiğidir.

İki farklı doğrulama yöntemi bulunmaktadır;

- Doğruluğun biçimsel olarak ispatlanması.
- Çok sayıda test yapılması.

Doğruluğun biçimsel olarak ispatlanmasında, eşbiçimlilik şartını sağlamak için matematik ve mantıksal formatlar kullanılır. Bununla beraber, bu yöntemi kullanmak büyük ölçekli ve karmaşık sistemlerde zordur.

İspatlamanın yapılamadığı durumlarda, simülatörün çalışması sırasında ortaya çıkması muhtemel tüm durumları belirlemek için çok sayıda test yapılmalıdır. Zaman ve diğer kaynaklardaki sınırlamalar, yapılacak test miktarını / sayısını düşürür [29].

Model *geçerleme*, bir model, sistem ve deneysel çerçeve arasındaki bağıntıdır. Geçerleme, çoğu kez bir modelin tanımladığı sistemi temsil etme derecesi olarak düşünülebilir. Bununla beraber, bir modelin, sadece simülasyon çalışmasının hedefleri tarafından talep edilen sistem davranışını yansıtması gerektiği pratik yaklaşımı daha fazla kabul görmüştür. Geçerleme kavramı, ilgilenilen deneysel çerçeve içinde model ve sistemi birbirinden ayırt etmenin mümkün olup olmadığı sorusunu cevaplar. Pratikte, bir modelin uygunluğunu yansıtmada modelleme ve simülasyon topluluğu tarafından farklı terimlerde kullanılmaktadır: '*Doğruluk'* terimi, genellikle geçerlemenin yerine kullanılırken, '*uygunluk (fidelity)'* terimi, hem geçerliliğin ve hem de diğer ayrıntıların bir birleşimi olarak kullanılmaktadır. Bu nedenle, yüksek-uygunluğa sahip bir model; hem yüksek ayrıntıya hem de yüksek geçerliliğe sahip bir model olarak adlandırılır (ilgili deneysel çerçeve kapsamında). Bir başka terim olarak simülatör doğruluğu, *simülatör* ve *model* arasındaki bir ilişkidir. Eğer bir *simülatör,* modelin başlangıç durumunun ve girişin fonksiyonu olan çıkışı tam olarak üretirse, o simülatör doğru bir şekilde modeli simüle eder. Simülatörler, sadece bir tane modeli çalıştırmak için değil, muhtemel bir model grubunu çalıştırmak için tasarlanırlar. Bu esneklik, eğer simülatör bir uygulamalar silsilesiyle uyumlu çalışabiliyorsa gereklidir. Bu gibi durumlarda, belirli bir model sınıfını sağlıklı bir şekilde çalıştırabilecek bir simülatör tasarlamak en mantıklı yoldur. Hem simülatörün hem de modelin yapısı iyi bilindiği için bir *eşbiçimlilik* ilişkisine sahip olduğunu göstererek doğruluğu kanıtlamak mümkün olabilir. Daha öncede ifade edildiği gibi, eşbiçimlilik, simülatör ve model durumları arasındaki bir benzerliktir ve durum geçişleri / çıkışları karşısında korunur.

Model geçerleme türleri üçe ayrılır: Kopyalama geçerliliği (replicative validity), tahmini geçerlilik (predictive validity) ve yapısal geçerlilik (structural validity) [22].

Kopyalama geçerliliği; oluşturulan modelin gerçek sistemin giriş/çıkış verisinin bir kopyasını oluşturabilmesiyle ilişkilidir. Bir başka değişle kopyalama geçerliliği, modelin ve sistemin davranışının deneysel çerçeve içindeki bütün muhtemel durumlar için kabul edilir bir sınır içinde olup olmadığını ölçer.

Tahmini geçerlilik, gerçek sistemin herhangi bir girişe olan cevabını kestirme kabiliyeti olan bir modelin durumunu tanımlama yeteneğiyle ilgilenir. Geçerliliğin daha güçlü biçimi olan *tahmini geçerlilikte*, bir modelin yalnız kopyalama geçerliliğine değil, aynı zamanda henüz görülmeyen sistem davranışını önceden tahmin edebilme yeteneğine de sahip olması gerekir.

Önemli bir sorun bir modelin tahmini olarak geçerli olup olmadığıdır: Sadece parametreleri seçmek için kullanılan veriyi tekrar üretme değil, model aynı zamanda yeni bir davranışı tahmin edebilme yeteneğine sahip midir? Bir modelin tahmini geçerliliği genellikle deneysel verilerin simülasyon tarafından üretilenlerle karşılaştırılmasıyla doğrulanır ve bu aktivite model geçerleme olarak bilinir. Model tasarımcısı ve kullanıcısı arasındaki bağlantıda önemli bir yere sahip olmasından dolayı, model geçerleme konusunda son yıllarda dikkate değer bir ilgi bulunmaktadır [27]. Genel geçerleme metodolojilerinden, somut test teknolojilerine kadar geniş bir alana yayılan problemler detaylı bir şekilde araştırmacılar tarafından ele alınmıştır. Deney sonuçları ile simülasyon verisinin karşılaştırılması işlemi; grafik karşılaştırması gibi sübjektif olarak veya değişken gerileme analizinin çok değişkenli analizi, spektral analizi, özbağlanım analizi, öz ilinti fonksiyon testi, hata analizi ve parametrik olmayan metotlar gibi istatistiksel olarak yapılır. Modelleme ile ilgili çeşitli konuların detaylı bir sunumu ile doğrulama, geçerleme ve test tekniklerinin sınıflandırılması konuları Balcı tarafından özetlenmiştir [27].

Yapısal geçerlilik; gerçek sistem ile model arasındaki yapısal ilişkiyle ilgilenir. Yani yapısal geçerlilik, modelin sadece sistemden gözlemlenen verinin kopyasını üretebilme yeteneğine sahip olmasına değil, aynı zamanda kendi durum geçişlerini yapan sistemin adım adım, parça parça taklidini yapmasını gerektirir.

Bu noktada deneysel çerçeve ve modelin farklı özellikleri arasındaki ilişkiyi açıklamak konunun daha iyi anlaşılmasına yardımcı olacaktır. Çok uzun bir süreçte elde edilen tecrübelerle bir araya getirilen modeller ve deneysel çerçevelerden oluşan bir bilgi birikimine sahip olunması durumunda, mevcut hedeflerimizi karşılayan bir deneysel çerçevenin ve bu çerçeve içinde çalışabilen modellerin olup olmadığını (sadece bu modeller bizim sorularımıza geçerli cevaplar sağlama şansına sahiptir) sorgulama olanağına sahip olmak çok önemli bir kavram olarak karşımıza çıkar. Mantıksal olarak bir çerçevenin bir modele uygulanabileceğini veya uygulanamayacağını belirleyen ilişki '*uygulanabilirlik' (applicability)* ve bunun tersi '*uyum' (accommodation)* olarak adlandırılır (Tablo 1). Belirli bir deneysel çerçevedeki bir modelin geçerliliği, bir ön koşul olarak modelin o çerçeveyle uyumlu olmasını gerektirir. Bir deneysel çerçevenin, bir diğerinden aynı koşullarda daha sınırlı olma derecesi, '*türetilebilirlik' (derivability)* ilişkisi şeklinde formülize edilir. Eğer deneysel çerçeveyi sistem dinamiklerinin geçerli bir tasviri olan modelin içinde olduğu ortamın resmi bir gösterimi olarak düşünürsek, daha sınırlı bir deneysel çerçeve daha özel bir davranış anlamına gelir. Böyle bir sınırlı deneysel çerçevenin daha genel bir deneysel çerçeveden daha fazla modelle eşleşeceği açıktır. Bu nedenle, modeller arasındaki *eşbiçimlilik* bir model değiştirildiğinde veya dönüştürüldüğünde (model lineer olmayan terimler ekleyerek) aynı deneysel çerçeve altındaki simülasyon sonuçları aynı kalmalıdır.

Tablo 2.1 Bir model kütüphanesinden bahsedildiğinde önem kazanan M&S ilişkileri.

İlişki	Tanım
Deneysel Çerçeve bir modele uygulanır (veya uygulanabilir).	Çerçeve tarafından gereksinim duyulan deney şartları model içine tatbik edilebilir
Model deneysel çerçeveye uygundur.	Çerçeve modele uygulanabilir
1 nolu deneysel çerçeve 2 nolu deneysel çerçeveden türetilebilir.	2 nolu deneysel çerçeveye uyumlu bir model aynı zamanda 1 nolu deneysel çerçeveye de uyumludur.

2.5 Soyutlama Seviyeleri ve Modelleme Yaklaşımları (Formalisms-Biçimsel Sistem Tanımlamaları)

Sistemlerin soyut modellerinin kullanılmasının birçok nedeni bulunmaktadır:

i- Bir sistemin soyut model tanımının o sistem hakkındaki bilgiyi yakalamasıdır. Bu bilgi saklanabilir, paylaşılabilir ve yeniden kullanılabilir.

ii- Soyut bir model, bir sistemin yapısı ve davranışı hakkındaki soruları analiz edip cevaplandırmayı sağlar. Model çoğu kez gerçek sistemde gözlemlenemeyen niceliklerin değerlerini elde etmek için kullanılır. Gerçek bir deneyi (simülasyon veya sanal deneye karşılık olarak) yapmak maddi açıdan uygun olmayabilir. Yapıyla ilişkili sorulara cevap bulma genellikle modelin sembolik analizi aracılığıyla olur. Örneğin bir elektrik devresinin bir döngü içerip içermediği merak edilebilir. Sistemin dinamik davranışını incelemek ve davranışı hakkındaki sorulara cevap bulmak simülasyon yoluyla gerçekleştirilebilir. Simülasyon sembolik veya nümerik olabilir. Modellemenin amacı bilginin anlaşılmasını ilerletmek ve yeniden kullanımı mümkün kılmakken, simülasyonun amacı doğruluğu ve hızı arttırmaktır.

Sistemlerin modellerini geliştirmede olası bir yol sistemin yapısını aynen kopyalamak olmasına rağmen bu temel bir gereksinim / şart değildir. Örneğin; bir atık su arıtma tesisindeki havalandırma tankının davranışını simüle eden bir sinir ağı, bir tank modeli olarak değerlendirilebilir. Her ne kadar tankın fiziksel yapısı ve içeriği bu modelde gözükmese de, model tankın davranışını doğru ve kabul edilebilir bir şekilde yansıtabilir. Kontrol amacına uygun olarak, belirli durumlar altında bir gerçek sistemin davranışını doğru bir şekilde tahmin edebilen ancak gerçek sistemle benzerliği olmayan bir model çoğu durumda yeterli olabilir.

Sistem davranışının soyut modelleri detaylı olarak farklı soyutlama seviyelerinde veya farklı modelleme yaklaşımları kullanılarak tanımlanabilir. Kullanılan belirli bir modelleme yaklaşımı ve soyutlama seviyesi, modellenen sistem kadar modelleyen kişinin altyapısına / bilgi birikimine ve hedeflerine de bağlıdır.

2.5.1 Sistemlerin tanımlaması

Sistem teorisi, '*sistemin yapısı'* (bir sistemin dahili oluşumu) ile '*sistemin davranışını'* (dış göstergesi) birbirinden ayırır. Bir sistemin giriş / çıkış davranışı, gerçek bir sistem veya modelden toplanan veri kayıtlarından oluşur. Sistemin iç

yapısı, hem durum çıkışlarını hem de durum / durum geçiş mekanizmalarını içerir. Sistem yapısının bilinmesi, sistemin davranışını (analiz, simüle etme) ortaya çıkarmaya olanak tanır. Genellikle, davranıştan yapıyı çıkarma işlemi kolay olmamasına karşılık, gözlenen davranışın geçerli bir temsilini bulmak, modelleme ve simülasyon aktivitesinin önemli aşamalarından birisidir.

Mevcut sistemler incelendiğinde, sadece yapı ve davranış üzerinde yapılan gözlemler elle tutulur araçlardır [30]. Modelleyen kişi, gözlemlere dayanarak bir sistemin daha karmaşık modellerini tasarlar. Karmaşık modelleri tasarlarken önemli bir kavram hiyerarşidir. Hiyerarşi modellerin sınıflandırılmasını ve yönetimini kolaylaştırır. Hiyerarşi içinde her bir yapı sistem hakkında daha detaylı bilgi sağlayarak bir öncekini ayrıntılandırır. Sistemin yapısı ve davranışı ile ilgili bazı soruların hiyerarşideki en alt seviyelerde daha iyi cevap bulduğu için, daha üst seviye bir modelden daha az detaylı bir modele doğru ters yönde hareket etmekte mümkündür. Özellikle en alt seviyede tanımlı eğriler biçimindeki açık davranış çoğu durumda gereklidir.

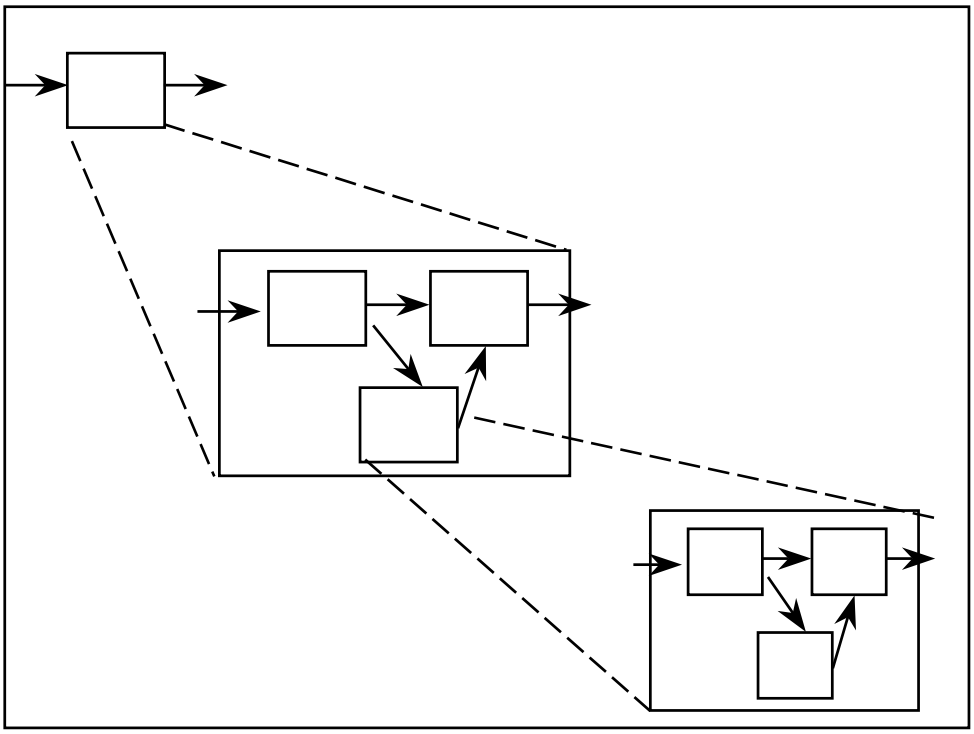
Önemli bir yapı kavramı, sistemin sistem parçalarına ayrılması olarak adlandırılan '*ayrışmadır'* (*decomposition*) (Şekil 2.7). İkinci bir kavram; sistemlerinin bir araya gelip birleşerek daha büyük bir sistemi meydana getirmesi olarak ifade edilebilecek *'bileşimdir' (composition)*. Sistemlerin bileşiminin yapısı ve davranışı temel sistem teorisi terimleriyle ifade edilebildiği için, sistemler teorisi *bileşim altında kapalıdır*. Daha önceden tasarlanan bileşenlerden daha büyük sistemleri oluşturma ve oluşturmaya devam etme işlemi '*hiyerarşik yapıyı'* ortaya çıkarır. Bileşim altında kapalılık, böyle bir birleşimin iyi tanımlı yapı ve davranışa sahip bir sistem (bileşke – 'resultant' olarak adlandırılır) ile sonuçlanmasını garanti eder. *Modüler* sistemler, çevre ile tüm etkileşimlerin yapıldığı giriş ve çıktı portlarına sahiptir. Modüler sistemler, giriş portları çıkış portlarına bağlanarak '*birleşik' (coupled)* hale getirilebilirler ve daha büyük bir sistemi oluşturmak için bir araya getirilen sistemler hiyerarşik bir yapıya sahip olabilirler.

Matematiksel kümeler ve işlemler soyut sistem gösteriminde veya modellemesinde başlangıç noktasıdır. Soyut sistem gösteriminde ve modelleme işlemlerinde sonlu

sayılar kümesi {1,2,...,9} ve karakterler {a,b,...,z} ile sonsuz kümeler (N, N^+, R ve R^+) kullanılır. Genellikle, kümelere ve küme elemanlarına özel bir anlam verilir.

2.5.2 Nesneye yönelik ilişki

Bir sistem teorisi kalıbıyla geliştirilen modeller ile nesneye-yönelik programlama kavramları bazı benzerlikler taşır. Nesneler ve sistem modelleri, dahili durum kavramını kullanır. Matematiksel sistemler bir zaman ekseni üzerinde çalışan biçimsel yapılarken, nesneye yönelik yaklaşımdaki nesneler genellikle zamansal kavramlara sahip değildirler. Tipik nesneye-yönelik kalıptaki nesneler, yukarıda tanımı yapılan manada hiyerarşik veya modüler değildirler.



Şekil 2.7 Hiyerarşik sistem ayrışımı.

Nesneye yönelik yaklaşımda nesnelerin birbirine bağlanması veya birleşmesi giriş ve çıkış arasında yollar kursa da, sistemleri modelleyen kişi verinin bu kanallar boyunca nasıl aktığını belirlemekte serbesttir. Bilgi akışı, temsil edilebilen pek çok etkileşimden biridir. Diğer etkileşimler, örneğin, fiziksel kuvvetleri ve alanları, materyal akışını, parasal akışları ve sosyal aktiviteleri içerebilir. Sistemler kavramı, bunların herhangi birinin bir temsilini kapsayacak kadar geniştir ve birden fazla

modelin bir büyük-ölçekli model içerisinde çalışabildiği M&S ortamlarının gelişmesini destekler.

Her ne kadar sistem modelleri klasik nesneler tarafından paylaşılmayan resmi bir zaman anlayışına bir birleşim özelliklere sahip olsa da, nesneye-yönelik yöntem, sistemleri modellemeye uygun destekleyici bir hesaplama mekanizması sağlar. Aslında, hiyerarşik ve modüler modelleme sistemlerinin birçok nesneye yönelik uyarlaması vardır. Bu uygulamalar, özellikle dağıtık hesaplama (distributed computing) için nesneye-yönelik kalıpların, modüler sistemlerin uyarlanmasında kuvvetli bir altyapı olarak hizmet edebileceğini ispatlar.

2.5.3 Zaman ekseni

Hayatımızda gerçek zamanda (real time) meydana gelen olayları incelerken, bir saat tarafından gösterilen zaman değişkenini kullanırız. Her simülasyon modeli, soyutlama seviyesinde durum geçişlerine olanak tanıyan bir '*sıralama karakteristiğine'* sahip olmalıdır. Zaman en genel bir sıralama karakteristiğidir. Zaman, değiştirilemez ve etki yapılamaz bir şekilde ilerlediği için özeldir: Bir sistemin mevcut durumu ve davranışı geçmişte değil ancak gelecek bir zamanda değişebilir. Bir değişim oluşmadan veya bir sonuç elde edilmeden önce bir '*neden'* olması gerektiğinden bu kavram, genellikle '*nedensellik'* olarak adlandırılır. Simülasyon ortamında oluşan sıralama karakteristiği, '*sistem zamanı'* olarak adlandırılır.

Zaman değiştirilemez ve etki yapılamaz bir olay / kavram olmasına rağmen izafiyet teorisine göre, farklı yerlerdeki gözlemciler tarafından farklı olarak algılanabilir. Bu ayrıntıya dayanarak, zaman '*yerel'* veya '*evrensel'* anlam içerebilir. Zamanın yerel olması, sadece bir sistemin bir bileşeninin içerisinde geçerli iken; zamanın evrensel olması, bütün sistem içinde geçerli olması anlamına gelir.

Bir model, bir ağa bağlı bilgisayarlar arasında dağıtık olabilen ve aynı zamanda gerçek dünya ile etkileşim yapabilen bir simülatörde çalıştırıldığında, zaman kavramı daha karmaşık bir hal alabilir. Farklı zaman yönetimlerine / kavramlarına

sahip modeller arasındaki senkronizasyon, bu gibi durumlarda dağıtık bir simülasyon protokolü yoluyla halledilir [26][31][32].

2.5.4 Temel sistemleri tanımlama yaklaşımlarının gelişimi

Bir fikrin gelişimini resmetmek veya tanımlamak, bu fikirler geliştirilirken ortaya çıkan karmaşıklıkların anlaşılmasında yardımcı olabilir. Sürekli durumlara ve sürekli bir zamana sahip olan klasik diferansiyel denklemli sistemler, *'Diferansiyel Denklemli Sistem Tanımı' (Differential Equation System Specifications - DESS)* ile ifade edilirler. Aynı zamanda ayrık bir zaman ekseninde çalışan sistemler, *'Ayrık Zamanlı Sistem Tanımı' (Discrete Time System Specifications - DTSS)* yoluyla tanımlanırlar. Bütün bu sistemlerin matematiksel temsil ve ifade biçimleri, bilgisayarlar keşfedilmeden önce olmuştur (örneğin; Newton-Leibnitz'den bu yana 300 yıl geçti). Dolayısıyla, bu sistemler bilgisayarlarda gerçeklenirken bir takım sıkıntılar ortaya çıkmaktadır.

Bilgisayarların keşfinden ve simülasyonların bilgisayarlarda kullanımından sonra ortaya çıkan ve sistemleri modellemede kullanılan yeni tanımlama yaklaşımı, *'Ayrık Olaylı Sistem Tanımlamaları' (Discrete Event System Specifications - DEVS)* olarak adlandırılmıştır [22]. Ayrık olaylı modeller başlangıçta, kendi simülasyon dili uyarlamalarına veya algoritmik kod ifadelerine oldukça bağlı kaldılar. Bölüm 3'te açıklanacak olan ayrık olaylı *"dünya görüşlerini",* resmi kalıplarla bağlantısı olmayan simülasyon yöntemlerinin ortaya çıkardığı iddia edilmiştir. Kontrol ve tasarımdaki soyutlamaların faydalarının açığa çıkması ile, ayrık olaylı sistemlerin resmi olmayan sistemler olduğu düşüncesi değişmeye başlamış ve DEVS genel bir modelleme yaklaşımı olarak modelleme ve simülasyon alanında yerini almıştır.

DEVS bütün yöntemleri ifade etmek için kullanılabilir. Ancak, ayrık zamanlı bir sistemin aynı zamanda ayrık olaylı bir sistem olduğu doğru değildir (mesela, zaman temelleri farklıdır). Bu nedenle, bir sistemin ne zaman başka bir sistemin temel bir işini yapabileceğini söylemeyi olanaklı kılan bir genel '*simülasyon'* kavramına ihtiyaç duyulur. Eğer herhangi bir modelleme yaklaşımında tanımlı bir sistem, başka bir yönteme sahip bir sistem tarafından simüle edilebilirse, o zaman ikinci yaklaşım birinciye katılabilir. Gerçekten, sistemlerin eşdeğerliğinin istenildiği çeşitli yapı ve

davranış düzeyleri olduğu için böyle bir veya birden çok ilişki veya '*dönüşüm-morfizm'* yararlı olabilir. Örnek olarak, herhangi bir DTSS'in zamanda ilerleyişi sabit yapılarak DEVS tarafından simüle edilebildiği saptanmıştır [33].

2.5.5 Birden fazla modelleme yaklaşımı kullanarak modelleme yapılması

Modelleme ve simülasyon alanındaki başlıca ilerlemelerden biri, ayrık olaylı ve diferansiyel denklemli modelleme yaklaşımlarının DEVS&DESS olarak bir araya getirilmesi olmuştur. Bu modelleme yaklaşımı, hem DESS hem de DEVS'i (dolayısıyla aynı zamanda DTSS'i) kapsamakta ve bu yüzden parçaları temel yaklaşımların herhangi biri ile ifade edilen birleşik sistemlerin gelişimi / geliştirilmesi fikrini desteklemektedir. Gerçek dünyanın genellikle sadece bir tek soyutlama biçimini kullanmaya elverişli olmaması veya farklı çalışma biçimlerine sahip değişik sistemlerden oluşması nedeniyle, *'çoklu formalizm' (multiformalism)* modelleme veya birden fazla yöntem kullanılarak modelleme yaklaşımı önemlidir.

Yukarıda yaptığımız açıklamalarda dolaylı olarak, bir birleşik modeli oluşturan bileşenlerin aynı yöntemde tanımlandığını kabul etmemize ve bir birleşik modele davranışsal semantiklerin mantıklı bir şekilde iliştirilmesi için birleşim altında kapalı olduklarını varsaymamıza rağmen, karmaşık sistemlerde bu kabuller / varsayımlar doğru değildir. Bu sistemler sadece çok sayıda bileşenlerle değil, aynı zamanda bileşenlerin farklılığı ile nitelendirilirler. Bu gibi karmaşık sistemlerin analizi ve tasarımında, farklı bileşenleri ayrı ayrı incelemek yerine bütün sistemin özellikleri / davranışı hakkında sorulara cevap bulmak daha önemlidir. Bu sistemler hakkında karar verme işinin, özel bileşenleri incelemek yerine bütün sistemin davranışını anlamayı gerektirdiği açıktır.

Sistemlerin ve sistemlerin modelinin karmaşıklığının nedenleri aşağıdaki şekilde sıralanabilir:

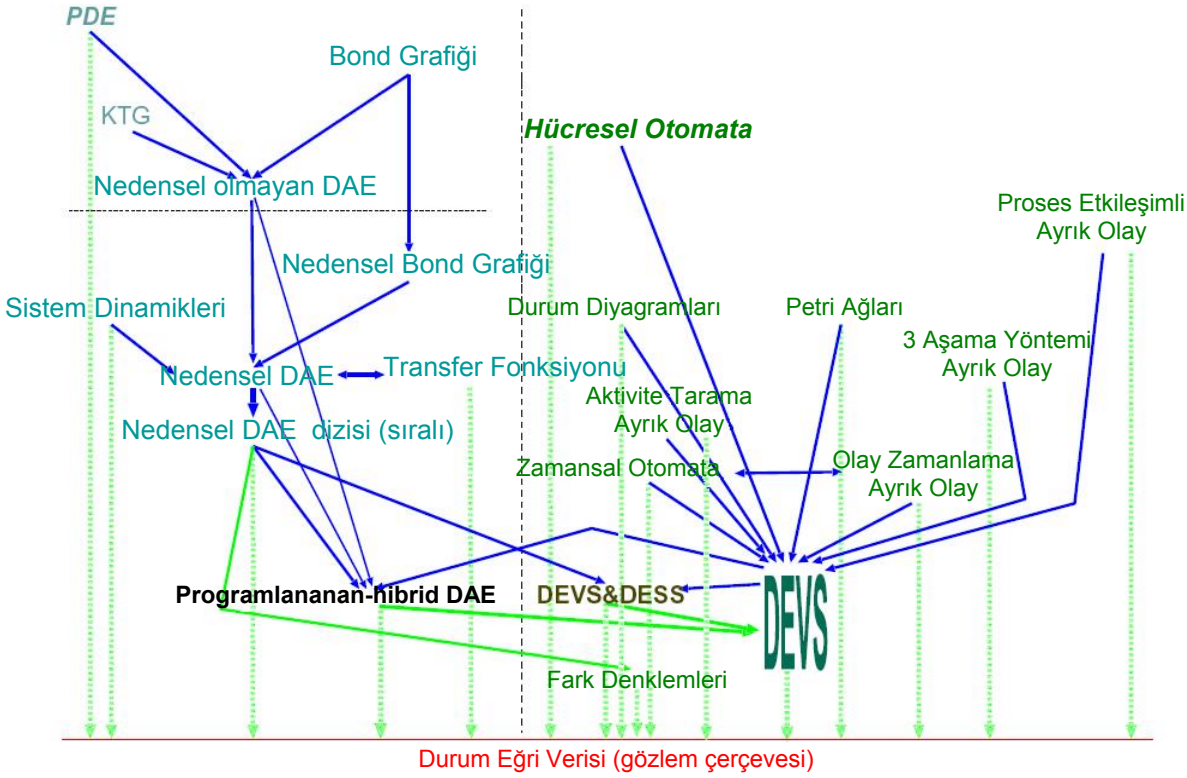
- Birbirleri ile etkileşen, birleşik ve eş zamanlı çalışan sistem parçalarının bulunması. Karmaşık davranış genellikle çok sayıda geri besleme döngüsü nedeniyle ortaya çıkar.

- Yazılım/donanım, sürekli/ayrık gibi sistem bileşenlerinin çeşitliliği.

- Bakış ve farklı soyutlama yaklaşımlarının çeşitliliği.

Bir sistemin modeli belirli bir soyutlama seviyesinde geçerlidir (belirli bir deneysel ortamında). Bileşenlerin her biri için farklı olabilen bu soyutlama seviyesi, mevcut bilgi ile sistemin davranışı hakkında sorular ve alınan cevapların doğruluğu tarafından belirlenir. Model soyutlama seviyesinin seçiminin yanında modelin tanımlandığı uygun yöntemin seçimi de önemlidir. Modelleme yönteminin seçimi soyutlama seviyesine, modeli ayarlamak için elde edilebilen veri miktarına, cevaplanması gereken soru türlerine ve kullanılacak modelleme yaklaşımına uygun uzmanın olup olmamasına bağlıdır. Aynı anda çalışan sistemleri modellenmesi gibi karmaşık çalışmaların tüm yönleri için tek bir kavramsal model veya tercih edilmiş bir yaklaşım olabilir. Aslında, birden fazla ifade seviyesine, farklı teorilere ve dillere ihtiyaç duyulur. Rasgele seçilmiş karmaşıklıkta sistemler araştırıldığında bu yaklaşım önem kazanır.

Bu konuda genel bir fikir vereceğini düşündüğümüz '*formalizm dönüşüm grafiği' (FDG)* Şekil 2.8'de görülmektedir [24]. Grafikteki düğümler modelleme yaklaşımlarını, alttaki yatay çizgi eğrilerin tanımlanabildiği I/O gözlem çerçevesini ve dikey noktalı oklar ilgili yaklaşımdaki deki bir modelden davranışı üreten bir simülatörün bir modelleme yaklaşımı için mevcut olduğunu gösterir. Diğer oklar davranışı koruyarak modelleme yaklaşımları arası dönüşümlerini göstermek için kullanılırken, dikey kesikli çizgili hat ayrık modelleme yaklaşımlarını (sağdakiler) sürekli model yaklaşımlarından ayırmak için kullanılır. FDG, genel olarak yaklaşımlar üzerinde birden fazla sınıflandırma yapmanın mümkün olduğunu göstermektedir.



Şekil 2.8 Modelleme Yaklaşımı (formalizm) Dönüşüm Grafiği (FDG) [24].

BÖLÜM 3. AYRIK OLAYLI SİSTEMLER VE DEVS MODELLEME VE SİMÜLASYON TEORİSİ

3.1 Giriş

Ayrık olaylı sistemler için sistem teorisi ve modelleme kavramlarını daha özel bir biçimde ifade eden ayrık olaylı sistem tanımlama (Discrete Event System Specification - DEVS) yaklaşımı olan DEVS, sadece ayrık olaylı modeller için değil, aynı zamanda ayrık zamanlı ve diferansiyel denklemli sistemlerle ifade edilen

davranışları uyarlamak için de bir hesaplama temeli oluşturması nedeni ile önemlidir.

Paralel DEVS, modern hesaplama teorisinde çok önemli bir yere sahip olan paralelliğin faydalı bir şekilde kullanılmasını engelleyen ve ilk zamanlardaki bilgisayarların sıralı operasyonları nedeniyle ortaya çıkan sınırlandırmaları ortadan kaldırmaktadır. Bu bölümde daha sonraki kısımlarda tartışılacak DEVS'in nesneye-yönelik çerçeveye uygulanması işlemine hazırlık amacı ile DEVS modelleme yaklaşımı konusu tanıtılmaya çalışılacaktır.

3.2 Ayrık Olaylı Modelleme Yaklaşımı

Zaman ekseninin sürekli olduğu, ancak sadece sınırlı bir zaman periyodunda sonlu sayıda olayların meydana geldiği sistemler, *'ayrık olay'* olarak adlandırılan bir soyutlama seviyesinde ele alınmıştır [22][23][25]. Bu olaylar sistemin durumunda bir değişikliğe neden olabilmelerine karşılık, olaylar arasında sistemin durumu değişmez. Bu sebeple *'ayrık olaylı modeller'*, sistemin durumunun zaman üzerinde sürekli değiştiği sürekli modellerden ayrılırlar.

Ayrık olaylı modelleme yaklaşımları yüksek bir soyutlama düzeyinde ele alınır. Bu soyutlama, sistem davranışının gerçeğe uygun bir şekilde tanımlanması için elverişlidir. Diğer taraftan, olaylar arasında sistemin durumunun değişmemesi nedeni ile, ayrık olaylı simülatör olay olmayan zamanlarda sistemin durumunu temsil etmeye gereksinim duymaz. Bu, durum bilgisinin sürekli bir zaman ekseni üzerinde her noktada gösterilmesi gerektiği sürekli simülasyon ile karşılaştırıldığında, son derece verimli bir simülasyona neden olur.

Yüksek bir soyutlama düzeyi, gerçek dünya davranışıyla ilgisi olmayan simülasyon araçlarını getirebilir. Özellikle birden fazla olayın aynı zaman diliminde meydana geldiği olay çakışmaları, ayrık olaylı modelin yeteri kadar detaylandırılmamasından kaynaklanabilir. DEVS modelleme yaklaşımı ve onun türevleri, oluşan olay çakışmalarını başarılı bir şekilde yönetir ve bu durumların ortadan kaldırılması konusunda güvenilir bir çözüm yolu sunar.

3.2.1 Ayrık olaylı sistemlerde kullanılan kavramlar / terimler

Aşağıda, ayrık olaylı simülasyonu ve farklı türlerini doğru bir şekilde anlamamıza yardımcı olabilecek kavramlar ile kavramların tanımları sunulmaktadır [24].

- ***An;*** bir nesnenin en az bir karakteristiğine atanabilen bir zaman değeridir.
- ***Aralık*;** birbirini izleyen iki an arasındaki süredir.
- ***Zaman periyodu (time-span);*** bir veya daha fazla aralığın ardı ardına gelmesidir.
- Nesnenin ***durumu***; belirli bir anda bir nesnenin tüm karakteristik değerlerinin listesidir.

Bir simülasyon modelinin '*statik'* ve '*dinamik'* olmak üzere iki farklı yapısı bulunmaktadır. Statik yapı; modelin olası durumlarını tanımlarken, dinamik yapı; durumun zaman üzerindeki değişimini tanımlar. Statik yapı genellikle nesneler ve nesne karakteristikleriyle ifade edilirken, bir modelin dinamik yapısının tanımlanmasında '*dünya görüşleri'* olarak bilinen farklı yöntemler kullanılır. Farklı dünya görüşleri konusunu ele almadan önce, aşağıda verilen kavramları tanımlamak konunun anlaşılmasına yardımcı olacaktır:

- ***Aktivite;*** bir aralık boyunca bir nesnenin durumudur.

- ***Olay;*** bir nesnenin durumunda bir anda olan bir değişimdir ve o ana kadar olması engellenen bir aktiviteyi / işlemi başlatır. Olayın oluş şartı yalnızca sistem zamanına bağlıysa, bu olay '*belirlenmiş'* bir olaydır ve bu özelliğe sahip modeller, simülasyon modellemede, '*zaman olayı'* olarak isimlendirilir. Zamana bağımlı olmadan sistemin şartlarına bağlı olarak gelişen / oluşan olaylar *tesadüfidir (contingent)* ve bu tür olaylar modellemede, '*durum olayı'* olarak isimlendirilir.

- ***Nesne aktivitesi;*** durum değişimlerine sebep olan birbirini izleyen iki olay arasındaki nesnenin durumudur. Diğer olaylar, diğer nesnelerdeki durum değişimlerine göre meydana gelebilir.

- ***İşlem (process);*** bir zaman periyodu boyunca bir nesnenin birbirini takip eden durumlarıdır. Bir veya daha fazla nesne aktivitesinin ardı ardına gelmesi, '*işlem'* olarak adlandırılabilir.

Belirli bir problem için, olayların ne olduğunu belirlemek / anlayabilmek için aşağıdaki adımlar takip edilebilir:

i) Nesnelerin ve nesne karakteristiklerinin tespit edilmesi.

ii) Sistemin karakteristiklerinin belirlenmesi.

iii) Bir olay olarak, herhangi bir karakteristik değerinde değişimin nedenlerinin tanımlanması.

Çoğu zaman sayıcılar, minimum değerler, maksimum değerler, ortalamalar, değişkenlerin frekans dağılımları, vb. performans ölçütlerinin hesaplanmasına yardımcı olacak ilave durum değişkenleri modele eklenir. Ayrık olaylı simülasyonda, ortalama kuyruk uzunluğu ve kaynakların kullanımı gibi performans ölçütleri sık sık kullanılır. Performans ölçütleri bir simülasyon çalışmasının sonucunda elde edilen çıkışlardır.

Aşağıda, ayrık olaylı simülasyonun kaynağını oluşturan çeşitli dünya görüşleri sunulmaktadır.

3.2.2 Ayrık olaylı simülasyon stratejileri

Ayrık olaylı simülasyon dillerinde yaygın olarak kullanılan üç farklı simülasyon stratejisi bulunmaktadır: *Olay zamanlama (event scheduling), aktivite tarama (activity scanning)* ve *süreç etkileşimi (process interaction)* [22]. "Dünya görüşleri"

olarak da adlandırılan bu stratejilerin her biri diğerlerinden daha doğal bir şekilde ifade edilebilir bir model tanımlama biçimi ortaya koyar. Bir başka deyişle, her bir simülasyon stratejisi dünyanın çalışma şeklini ele alırken orijinal yaklaşımlara sahiptirler. Yukarıda isimleri verilen stratejilerden sonuncusunun ilk ikisinin bir kombinasyonu olması yanında günümüzde geliştirilen simülasyon ortamlarının çoğu bu stratejilerin bir kombinasyonunu kullanmaktadır.

i- Olay zamanlama simülasyon stratejisi

Olaya yönelik modelleme ve simülasyonda bütün olaylar *önceden planlanır*. Bir olayın planlanması, olayın gerçekleşmesi için gerekli bütün şartlar önceden bilinebildiği durumlarda mantıklı olur. Olay zamanlama simülasyon stratejisinde model; her bir olay için olayın durum üzerindeki ve sistemin gelecekteki davranışı üzerindeki etkisini tanımlar. Bu, yeni olayların gelecekteki bir zamana programlanmasıyla yapılır.

Şekil 3.1'de görüldüğü gibi bir olay zamanlama simülasyon çekirdeği iki (global) veri yapısı kullanır. Veri yapılarından bir tanesi modelde bildirilen durum değişkenlerini, diğeri ise artan zamanla ve azalan öncelikle sıralı bir olay listesinde zamanlanmış olay bildirimlerini içerir. Bir olay programlandığı zaman listede en sona eklenir. Öncelikler aynı zamanda gerçekleşen (çakışan) olaylar arasında seçim yapmak için kullanılır. Durum değişkenleri, durum değişkenlerinin ve kombinasyonlarının minimumunu, maksimumunu, ortalamasını ve standart sapmasının hesaplanması için ilave performans değişkenleriyle arttırılabilir. Bir olay zamanlama çekirdeği, olayları *'olay listesi'* şeklinde düzenleyerek (artan zaman sırasına göre), listenin en üstündekini kaldırıp işleyerek ve bunu liste tamamen boşalana kadar tekrarlayarak çalışır. Olay bildirimindeki olay zamanı, simülasyon zamanını ilerletmek için kullanılır. Olay bildirimindeki olay türüne bağlı olarak, uygun olay bildirimi yapılabilir. Bu rutin, olay bildirimlerini olay listesine koyarak sistemin durumunu değiştirebilir ve gelecekte yeni olaylar zamanlayabilir. Olay zamanlama yaklaşımında:

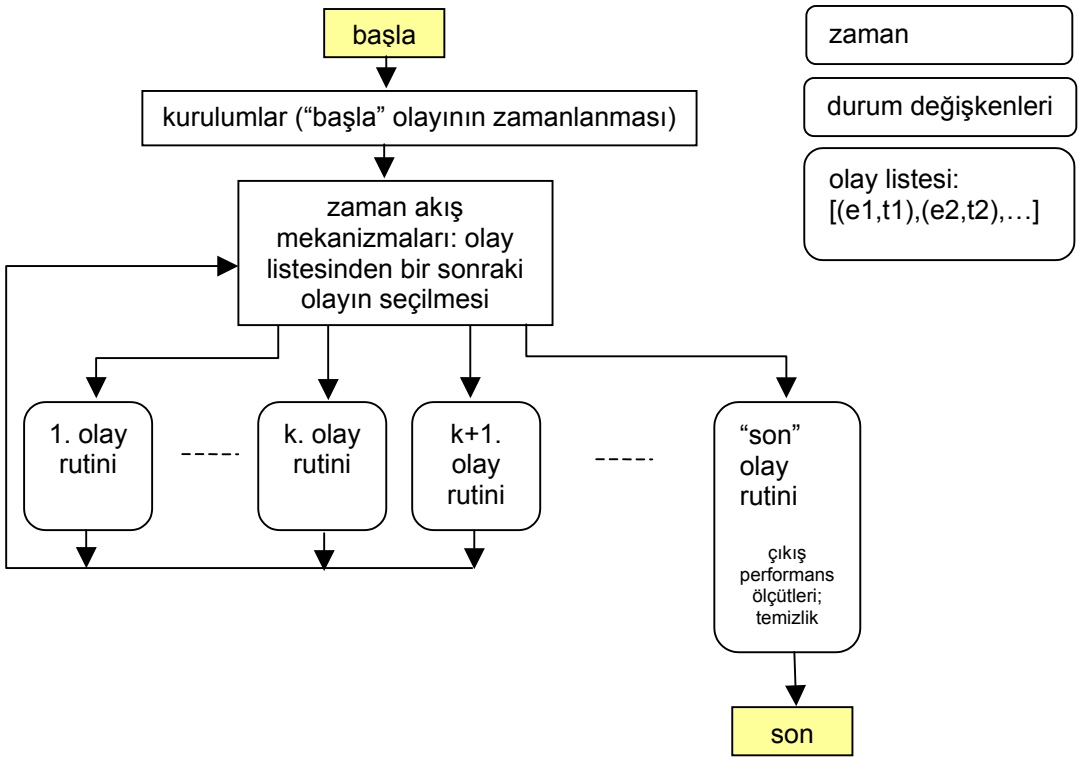
- Sistemin kurulumu ve ön zamanlamalı olaylar bir *'başlangıç'* olayına konulabilir. Bu olay otomatik olarak olay listesine yerleştirilir ve daha sonra normal bir olay gibi işlenir.

- Belirli bir simülasyon zamanında simülasyonun sonlandırılması, simülasyon prosedürü tarafından son olay olarak işlenecek olan özel bir 'son' olayının zamanlanmasıyla halledilir. Bu olay performans ölçümleri çıkışını ve elde edilen diğer istatistikleri üreten işlem yönergelerini içerir.

Bir olay zamanlama modeli, simülasyon zamanını ilerleten, olay listesini ve sistem durumunu güncelleyen yinelemeli bir simülasyon prosedürüyle simüle edilir.

ii- Aktivite tarama simülasyon stratejisi

Aktivite tarama yönteminde, bir takım aktiviteleri etkinleştiren şartlar tanımlanır. Bu yaklaşım, Prolog gibi bildirimli yapay zeka (Artificial Intelligence - AI) dillerinde kullanılan yaklaşımla benzerlikler taşır [34]. Bir aktivite tarama simülasyonu, zamanı ilerletmek için ayrık zaman adımını kullanır. Aktivite tarama safhasında, çözücü (solver) aktivitenin şartı sağlayıp sağlamadığını kontrol eder ve eğer şart doğruysa işler. Bu tarama işlemi en azından bir aktivite şartı doğru olarak değerlendirilene kadar devam eder. Eğer aktivitelerin hiçbiri şartı sağlamazsa, zaman akış safhası zamanı ilerleterek tekrar çalıştırılır. Petri Ağları aktivite tarama tabanlı bir modelleme yaklaşımına örnektir [60].

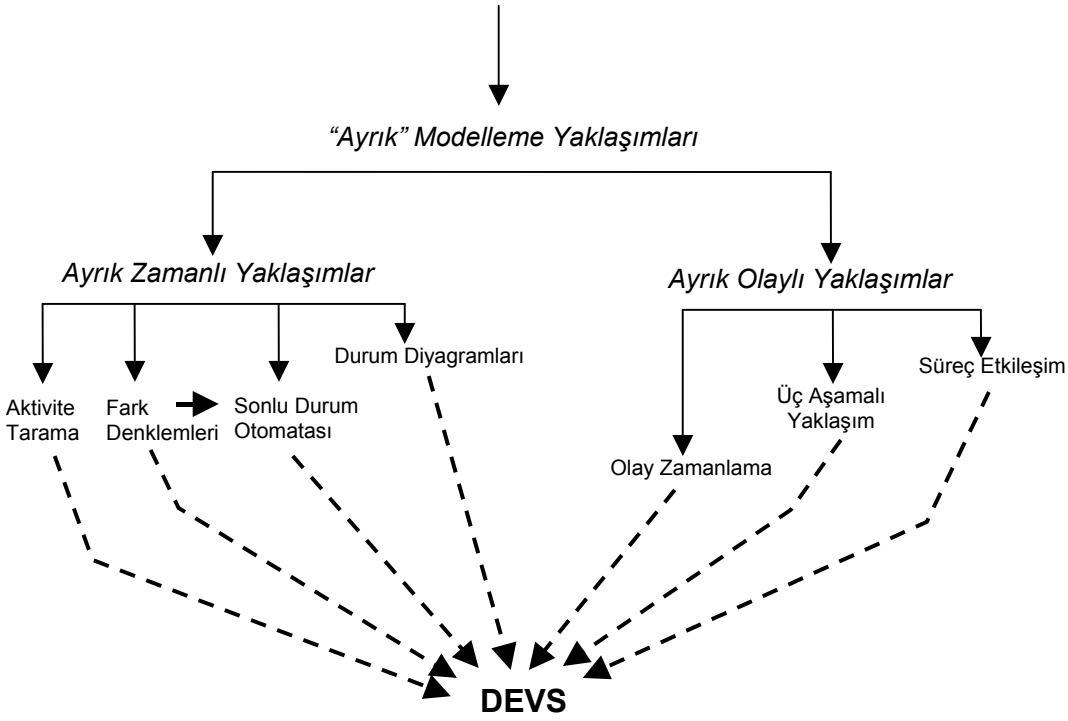


Şekil 3.1 Olay zamanlama simülasyon çekirdeği

iii- Süreç etkileşim simülasyon stratejisi

Aktivite tarama sabit bir zaman adımı kullanır ve bu pek çok durumda verimli değildir. Davranışı doğru bir şekilde modellemek için iki olay arasındaki zaman aralığı olabildiğince küçük seçilmelidir. Diğer taraftan, bazı olaylar zaman bakımından çok seyrek olabilir. Bu gibi uzun zaman aralıklarında, aktivite tarama simülatörü, şartlar değişmemesine rağmen zamandaki her noktada tüm şartları gereksiz bir şekilde kontrol eder.

Süreç etkileşim simülasyon stratejisinde, aktivite tarama yaklaşımı olay zamanlama yaklaşımıyla birleştirilir. Aktiviteler, olay zamanlama simülasyon stratejisindeki gibi gelecekteki bir zamana programlanır. Buna ek olarak, olay zamanlarında, bütün aktivite şartları aktivite tarama simülasyon stratejisindeki gibi kontrol edilir.



Şekil 3.2 Simülasyon stratejilerinin sınıflandırılması

3.2.3 Ayrık olaylı simülasyon stratejileri arasındaki ilişkiler

Şekil 3.2, farklı ayrık yaklaşımlar arasındaki ilişkilerin bir özetini göstermektedir. Şekilde sol tarafta zaman akış mekanizmasının sabit bir zaman ilerlemesi şeklinde olduğu yaklaşımlar sağ tarafta ise, *'ayrık olaylı'* zaman akışına (saat, sadece olayların olduğu zamanlarda ilerler) sahip yaklaşımlar görülmektedir.

Ayrık olaylı modelleme yaklaşımları modüler olmayan bir formdadırlar. Yani olay işleyicisi (event handler), aktiviteler, işlem blokları, vb. model bileşenleri bir takım arabirimler aracılığıyla kendi ortamıyla etkileşen varlıklar değildir. Bundan çok, direk olarak global durum değişkenleri ile birlikte diğer bileşenleri etkilerler.

Şekil 3.2'deki kesikli çizgiler, bir kaynak modelleme yaklaşımında tanımlı bir modelin hedef yaklaşımda tanımlı aynı modele dönüşümünü göstermektedir. Orijinal ve dönüşmüş modeller, aynı başlangıç şartlarında simüle edildiğinde aynı durum eğrilerini üretirlerse "denk" olarak adlandırılırlar. Şekilde tüm dönüşümler

bir sonraki bölümde tanımlanacak olan DEVS yaklaşımına yönelmiştir. Tüm bu dönüşümlerin özü, modüler olmayan tanımlamalardan modüler bir *'DEVS Birleşik modelinin'* kurulmasıdır. DEVS birleşik modeli, modüler bileşenler arasındaki bağlantılar aracılığıyla bağımlılıklar açık bir şekilde gösterilerek elde edilir.

3.3 Ayrık Olaylı Sistem Tanımlama (DEVS) Yaklaşımı

DEVS modelleme yaklaşımı, ayrık olaylı modelleme ve simülasyonda temel bir yaklaşım olarak Zeigler tarafından ortaya atılmıştır [23] [25]. DEVS yaklaşımının sadece ayrık olaylı modeller için değil, ayrık zamanlı ve diferansiyel denklemli sistemlerle ifade edilen davranışları uyarlamak için bir hesaplama temeli meydana getirmesi nedeni ile, modelleme ve simülasyon aktivitelerinde DEVS teorisini anlamak önemlidir. Ayrık olaylı yaklaşımlar sınıfı içinde modeller, zamanın sürekli olduğu ancak sınırlı bir zaman periyodunda sonlu sayıda olayın meydana geldiği bir soyutlama seviyesinde tanımlanır. Bu olaylar sistemin durum değiştirmesine neden olabilir. Olaylar arasında sistemin durumu kesinlikle değişmez. Sistemin durumunun zaman içerisinde sürekli değiştiği sürekli modellerden bu noktada farklıdır.

DEVS modelleme ve simülasyon yaklaşımı deterministik ve nedensel sistemleri klasik sistem teorisi kalıbına uydurur. DEVS sistem davranışını iki farklı seviyede tanımlar: Atomik DEVS ve birleşik DEVS. Atomik DEVS; en düşük seviyede, sıralı durumlar arasındaki geçişler gibi ayrık olaylı sistemin otonom davranışını, harici bir girişe (olaylar) nasıl tepki verdiğini ve çıkışı (olaylar) nasıl hesapladığını tanımlar. Birleşik DEVS; daha yüksek bir düzeyde, bir sistemi bileşenler ağı olarak tanımlar. Bileşenler, atomik DEVS modelleri ve birleşik DEVS modelleri olabilirler. Bağlantılar, bileşenlerin birbirini nasıl etkilediğini gösterir. Özellikle, bir bileşenin çıkış olayları ağ bağlantısı aracılığıyla bir diğer bileşenin giriş olayları olabilir. Her bir birleşik DEVS için bir atomik DEVS tasarlanabileceği gibi, atomik veya birleşik olan bir DEVS modeli bir atomik DEVS ile gösterilebilir. Birleşik DEVS, başka birleşik DEVS bileşenlerine sahip olabildiği için hiyerarşik modelleme yapısı desteklenir.

Aşağıda DEVS modelleme yaklaşımının farklı yönleri daha detaylandırılmaktadır.

3.3.1 Atomik DEVS modelleme yaklaşımı

Atomik DEVS modelleme yaklaşımı, sistemin ayrık olaylı davranışının değişik yönlerini tanımlayan bir yapıdadır. DEVS atomik model tanımı, değişken zaman periyotlarına sahip parçalı sabit eğriler şeklinde durumları (değişken değerler) ve ilişkili zaman eksenini tanımlar. Atomik model tanımı yeni durum değerlerinin nasıl üretileceğini ve ne zaman bu yeni değerlerin etkin olacağını da açıklar. Paralel bir atomik DEVS modeli aşağıdaki yapıdadır:

$$M=<X, S, Y, \delta_{int}, \delta_{ext}, \delta_{conf}, \lambda, ta>$$

Burada;

X, giriş değerleri kümesi,

S, durumlar kümesi,

Y, çıkış değerleri kümesi,

δ_{int} : $S \rightarrow S$ *dahili geçiş* fonksiyonu,

δ_{ext} : $Q \times X \rightarrow S$ *harici geçiş* fonksiyonu,

burada; $Q = \{(s,e) | s \in S, 0 \leq e \leq ta(s)$ *toplam durum* kümesi,

e, en son olan geçişten bu yana *geçen* süredir.

δ_{ext} : $Q \times X \rightarrow S$ *çakışma (confluent) geçiş* fonksiyonu,

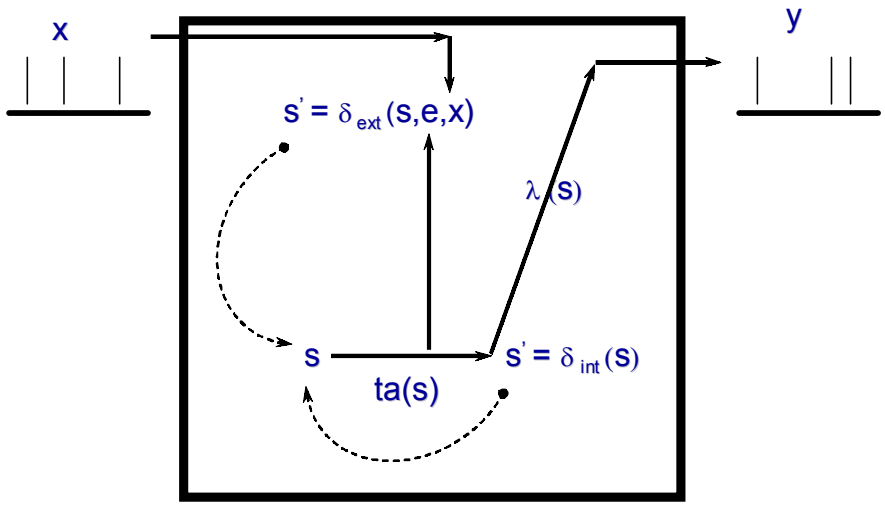
λ: $S \rightarrow Y$ çıkış fonksiyonu,

ta: $S \rightarrow R^{+}_{0,\infty}$ zaman ilerleme (time advance) fonksiyonu, 0 ve ∞ arasındaki pozitif reel sayılar kümesidir.

Bu elemanların yorumu Şekil 3.3'de görülmektedir. Herhangi bir anda sistem bir '*s'* durumundadır ve hiçbir harici olay meydana gelmemesi durumunda, sistem *ta(s)* zamanı süresince '*s'* durumunda kalır. Yukarıda yapılan tanımdan anlaşılacağı üzere '*ta(s)'* bir reel sayı olmasına rağmen '0' ve '∞' değerlerini de alabilmektedir. zaman ilerleme fonksiyonu *ta(s) = 0* olduğunda, '*s'* durumunun süresi araya başka olaylar giremeyecek kadar çok kısadır – bu durumda '*s'* durumunun bir *geçici* durum olduğunu belirtmek yanlış olmaz. İkinci durumda (*ta(s)* = ∞ olduğunda), harici bir olay bu durumu bozmadıkça sistem sonsuza kadar '*s'* durumunda kalır ve bu durumda *'s' pasif* bir durum olarak tanımlanır. Bir durumda kalma süresi dolduğu

zaman (geçen süre $e=ta(s)$ olduğu zaman), sistem $\lambda(s)$ değerini çıkış olarak verir ve $\delta_{int}(s)$ durumuna geçer. Bu anda, çıkışın dahili geçişlerden hemen önce üretildiğine dikkat edilmelidir.

Eğer bir $x \in X$ harici olayı bitiş zamanından önce meydana gelirse (sistem $e \leq ta(s)$ ile (s,e) durumundaysa), sistem $\delta_{ext}(s,e,x)$ durumuna geçer. Bu nedenle dahili geçiş fonksiyonu, en son geçişten itibaren hiçbir olay olmadığı zaman sistemin yeni bir duruma geçmesine neden olur. Bu durum, '*x'* girişi, '*s'* şimdiki durumu ve sistemin bu durumda ne kadar kaldığını gösteren '*e'* tarafından belirlenir. Her iki durumda da sistem, yeni bir $ta(s')$ sükunet zamanı ile yeni bir s' durumundadır ve aynı olay devam eder.

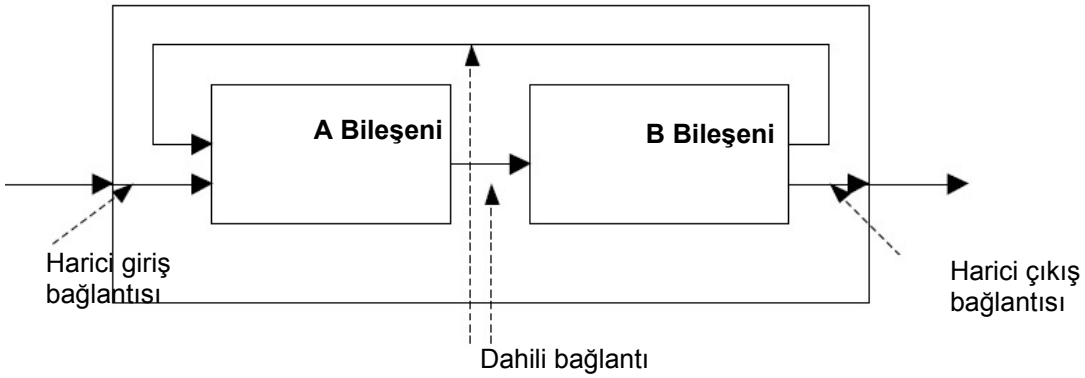


Şekil 3.3 DEVS işleyiş mekanizması.

3.3.2 Birleşik DEVS modelleme yaklaşımı

Birleşik bir model bir takım bileşen modellerden ve bunların birbirleriyle olan bağlantılarından oluşur. Daha öncede ifade edildiği üzere bileşenler atomik veya birleşik olabilir. Birleşik modelin davranışı bileşenlerin davranışıyla ve / veya bağlantısıyla tanımlanabilir. Bağlantı üç türe ayrılır: Harici giriş bağlantısı, harici çıkış bağlantısı ve dahili bağlantı (Şekil 3.4).

Harici giriş bağlantısı; birleşik modelin kendisi ve bileşenlerin bir veya birden fazlasının arasında olur. Harici çıkış bağlantısı ise; bileşen çıkışları ile birleşik modelin çıkışları arasındaki bağlantıdır. Dahili bağlantı; bileşen çıkışları ve bileşen girişleri arasında yapılır.



Şekil 3.4 Birleşik DEVS yaklaşımında bağlantılar.

Birleşik bir DEVS modelleme yaklaşımı aşağıdaki yapıdadır;

$$CM = < X, Y, D, \{M_i\}, EIC, EOC, IC, Select >$$

Burada;

X, Y : giriş ve çıkış kümeleridir

D : birleşik modelin bileşenler kümesidir

her $i \in D$ için,

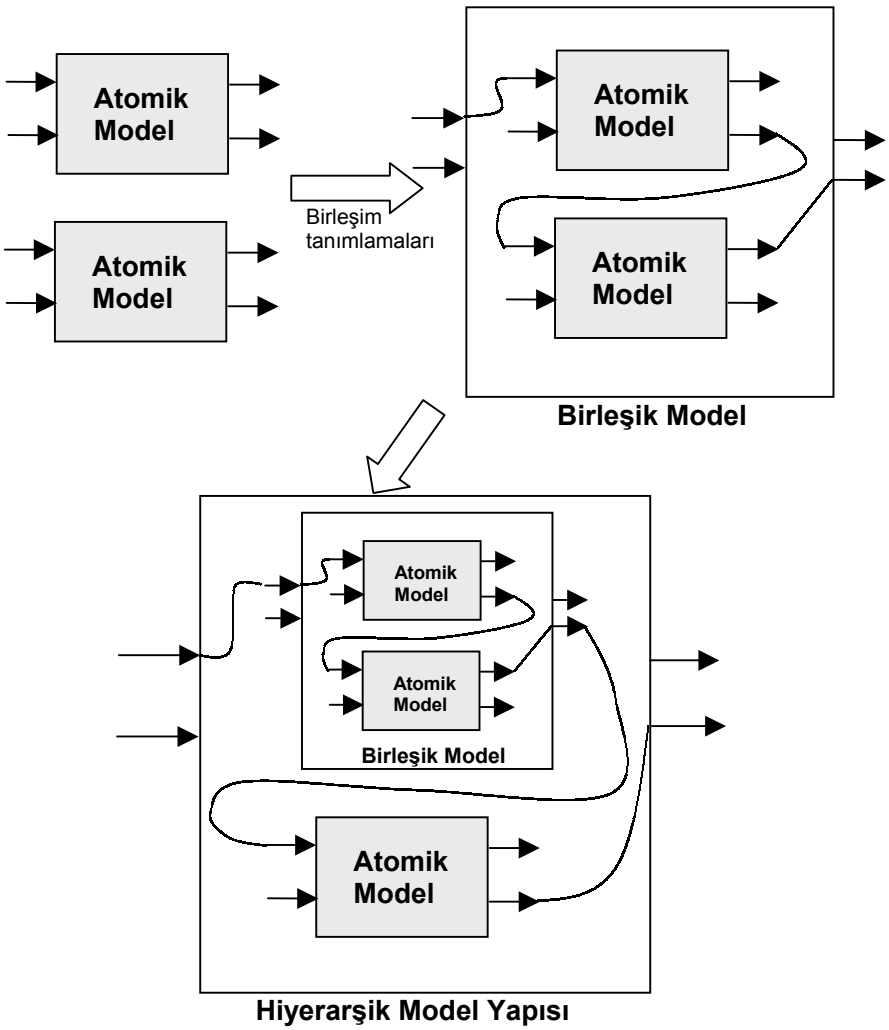
M_i :atomic veya birleşik olabilen bir bileşenin DEVS modelidir;

$EIC \subseteq X \times \cup_i X_i$, harici giriş bağlantı ilişkisi;

$EOC \subseteq \cup_i Y_i \times Y$, harici çıkış bağlantı ilişkisi;

$IC \subseteq \cup_i Y_i \times \cup_j X_j$, dahili bağlantı ilişkisi;

$Select: 2^{\{Mi\}} - \varnothing \rightarrow \{M_i\}$, eşitlik fonksiyonudur.



Şekil 3.5 DEVS hiyerarşik modüler birleşimi

3.3.3 Hiyerarşik model tasarımı: DEVS birleşim çerçevesi

Birleşik bir model DEVS modelleme yaklaşımında temel bir model olarak ifade edilebilir. Bu temel model daha büyük bir birleşik model içinde kullanılabilir. Bu, DEVS yaklaşımının hiyerarşik model yapısı için gerekli olan birleşim altında kapalı olduğunun göstergesidir. Bir birleşik modeli denk bir temel model olarak ifade etmek, tüm davranışı vermek üzere bileşenlerin etkileşmesi yoluyla yapılır. Daha önce ifade edildiği gibi, birleşik modellerin atomik modüller gibi daha büyük sistemlerde bileşenler olabilirler ve bu hiyerarşik ayrışmaya ve yapıya neden olur.

Şekil 3.5, bir bağlantı tanımını bir takım modellere uygulama yoluyla nasıl bir birleşik model ürettiğimizi göstermektedir. Bu modeli daha büyük bir sistemde yeni bileşenlerle birlikte bir bileşen olarak kullanarak ve bağlantı bilgisini ekleyerek, hiyerarşik birleşik model elde ederiz [22].

BÖLÜM 4. AĞLARIN VE DAĞITIK SİSTEMLERİN YÖNETİMİ

4.1 Giriş

Bilgisayar ağları, ekonomik ve ticari sebepler neden ile, bütün olası durumlar altında başarılı çağrı bağlantısını garanti edecek bir donanıma sahip olma yerine, birçok kullanım durumunda kabul edilebilir bir performans sunacak en düşük seviyede bir donanıma sahiptirler. Ağ şartlarında önemli bir değişim olması durumunda (bir anda toplam çağrı hacmi anormal bir şekilde yükselir veya herhangi bir düğüm aniden büyük hacimli çağrıların merkezi veya hedefi olursa), kapasite sınırlamaları ve bağlantı kurulamayan çağrılar nedeni ile sistem / ağ hatalı çalışma durumuna gidebilir. Günümüzde kullanılan ağların büyük bir kısmında iki nokta arasındaki iletişimde takip edilecek birden fazla alternatif yol bulunabilir ve iletişim / iletilen mesaj bir takım ara anahtarlama istasyonları veya düğümler yoluyla yönlendirilebilir. Bu yapı nedeni ile yedek kapasiteye sahip ağ kısımları kullanılarak mesajların yönlendirilmesi ve mevcut veya potansiyel tıkanıklıkların hafifletilmesi mümkündür. Bu işlem ‘yük dengeleme’ olarak adlandırılır ve sistemdeki değişen yükü eşit olarak dağıtan ve kayıp çağrıları en aza indirgeyen çağrı yönlendirme sistemlerinin kurulması olarak tanımlanır.

Yönlendirme algoritmalarının temel görevi; ağ performansını en üst seviyeye getirmek ve maliyeti en aza indirgeyerek ağın sağlıklı hizmet vermesini tesis etmektir. Bir ağa yönlendirme algoritmalarının uygulanmaması durumunda, ağda

tıkanıklıklar meydana gelebilir ve bu tıkanıklıklar büyük miktarda verinin kaybolmasına neden olabilir. Sonuç olarak, dağıtık bir sistemde ağ yönetimi ve trafik yönlendirmesi sağlıklı bilgi iletişimi için zorunludur.

Bu bölümde bir ağın veya dağıtık bir sistemin idare edilmesi için gerekli işlevleri tanımlamanın ve hizmetler bakımından yönetimin ne olduğunu kısaca açıklamanın yanında yönetimin stratejik önemi açıklanmaktadır. Ayrıca, bu tezde sunduğumuz çalışmanın, bir problemin çözülmesi hedefine odaklanan modelleme ve simülasyon aktivitesinin problem alanını temsil eden dağıtık sistemler ile yönlendirme algoritmaları verilmiştir. Paralel ve dağıtık sistemlerin genel anlayışı ve bu sistemler arasındaki temel nüans farklılıkları kısaca ele alınmıştır. Ayrıca ağ kontrol ve izleme işlevinin merkezi olmaktan ziyade, çok merkezli ve dağıtık olmasının gerekliliği ve avantajları ele alınmıştır. Son on yıldır ağ yönetiminde yaygın bir kullanıma sahip olan gezgin görevli (mobile agent) yaklaşımı ve ağ yönetimine getirdiği yenilikler açıklanmaktadır. Ağ yönetimi ve kontrolünde yeni bir araştırma alanı olan büyük ölçekli biyolojik sistemlerin (arılar ve karıncalar gibi) ve bu sistemlerdeki önemli optimizasyon yapılarının ağlara uygulanması ile bu alanda daha önce yapılmış çalışmalar yine bu bölümün kapsamındadır.

4.2 Paralel ve Dağıtık Sistemler

Karşılıklı birbirine bağlı bilgisayarlar '*ağ'* olarak adlandırılırken, ağ terimi çoğu kez *'iletişim sistemiyle'* eş anlamlı olarak kullanılmaktadır [1][3][7]. İletişim sistemi, bilgisayarların ve bilgisayar temelli sistemlerin karşılıklı bilgi alışverişine izin veren donanım ve yazılım parçalarından oluşmaktadır.

Son yirmi yıldır ağ ve dağıtık sistemler, ağ mimarileri ve protokolleri, iletişim metotları ve teknikleri, dağıtık programlama dilleri, eşanlı (concurrency) kontrol mekanizmaları, vb. teknolojiler hakkında çok sayıda dikkate değer araştırma / geliştirme çalışmaları yapılmış bulunmaktadır. Örneğin, daha hızlı iletişim sağlayan ağlar ve daha güçlü programlama dilleri konularında yapılması gereken çalışmalar bulunsa da, bu konular ile ilgili problemlerin çoğu çözümlenmiş durumdadır. Karmaşık yapıya sahip ağlar ve bunlar üzerinde çalışan dağıtık sistemleri

geliştirmek mümkün olsa da, bu sistemleri modellemek ve yönetmek çok zor bir problem olarak karşımıza çıkmaktadır [2].

Dağıtık çalışan bir sistem, birkaç bağımsız işlemciden ve bu işlemcileri destekleyen veri depoları veya veritabanlarından oluşur [7]. Bunlar bütün bir hedefi elde etmek üzere işbirliği yapmak için etkileşim halindedirler. İşlemler kendi aktivitelerini koordine ederler ve bir haberleşme sistemi aracılığıyla bilgiyi karşılıklı değişirler. İşbirliğini tesis etmek için sistemin bileşenleri arasında çok yakın bir senkronizasyon olması gerekir.

Dağıtık bir sistemde paylaşımlı bir belleğe sahip olmayan makineler topluluğu birbirinden bağımsız olarak çalışıp, farklı veya benzer işlemleri gerçekleştirirken, paralel bir sistemde işlemciler genellikle ortak bir hedefi gerçekleştirmek üzere birbirine bağlanırlar ve ortak bir belleği kullanırlar. Ağ sistemi ise genellikle, gönderilen ve alınan mesajlar arasında gecikmelerin oluştuğu, senkronizasyonun ve ortak bir hedefin olmadığı mesaj tabanlı bir sistem olarak tanımlanır.

4.3 Ağların Yönetimi ve İletişim Ağlarında Yönlendirme (Routing)

Bu kısımda, bir ağın veya dağıtık bir sistemin yönetilmesi için gerekli işlevler ve hizmetler bakımından yönetim konusunun ne olduğunu açıklanmaktadır. Yönlendirme, bütün düğümlerin doğrudan birbirine bağlı olmadığı bir ağda bir kaynaktan bir hedefe ara anahtarlama istasyonları veya düğümler aracılığıyla çağrıların iletilmesine olanak tanıyan bir mekanizma şeklinde tanımlanabilir. Maliyetin yüksek olması nedeni ile bir kaç taneden daha fazla düğüm için tam bağlantılı bir ağ mümkün değildir [5]. Yönlendirme kullanıcı trafiği ve ağ tarafından koyulan sınırlamalar ve hedefleri karşılayan yolları seçerek, hangi ağ kaynağının kullanıcı trafiği tarafından kat edileceğini ve kullanılacağını belirler. Bir haberleşme ağında yönlendirmenin amacı; kullanıcı trafiğini, ağın hizmet gereksinimlerine ve hizmet kısıtlamalarına uygun olarak bir kaynaktan bir hedefe doğru yönlendirmektir [3]. Hedefler, ağın ekipman ve araçlar gibi maliyetlerini en aza indirirken, ağın performansını (gecikme ve çıkış) en üst seviyeye çıkarmayı içerir. Kısıtlamalar, temeli oluşturan ağ anahtarlama teknolojisi, ağın ve kullanıcı trafiğinin dinamikleri, sağlanan ağ servisleri ve istenen kullanıcı hizmetleri tarafından ortaya konur.

Mümkün olan en kısa yolu bulmanın yöntemlerinden birisi, en kısa olması muhtemel yol ile birbirine bağlı olan iki düğümün belirtildiği uygun yönlendirme tablolarının tasarlanmasıdır. Bu gibi yönlendirme tablolarının tasarımı basit bir optimizasyon problemidir ve ağ topolojisinin tanımlanması aşamasında çözülmesi gerekir. Bununla beraber, trafik şartlarının sürekli olarak değişmesi sonucunda ağın kendisinin yapısı düzensiz bir şekilde değişebilir (anahtarlama istasyonları ve bağlantılar bozulabilir). Birden fazla hedefe / amaca ve birden fazla kısıtlamaya sahip bir optimizasyon problemi olan yönlendirme, araştırma yapmak için ideal bir alandır. Ağ teknolojilerinin hızlı gelişimi, sürekli olarak yönlendirme ile ilgili konularda yeni araştırma / geliştirme konuları ortaya çıkmaktadır. Her ne kadar, eski teknolojiler için tasarlanmış yönlendirme tekniklerinin belirli durumlarda yeni teknolojilere uygunluğu ispatlanmış olsa da, bu tezde de incelenecek olan yeni yönlendirme yöntemleri (gezgin görevli – mobile agent, oğul zekası – swarm intelligence tabanlı yönlendirme teknikleri, vb.) bir çok durumda gerekli olmaktadır.

4.3.1 Yönlendirmenin işlevleri

Farklı ağlar farklı yönlendirme algoritmaları kullansa da, bütün haberleşme ağları temel bir yönlendirme işlevini gerçekleştirirler. Bütün ağlar için genel olan çekirdek yönlendirme işlevlerinden ilki, *yolların* (*routes*) üretilmesinde ve farklı yollardan birisinin seçilmesinde kullanılan ağ / kullanıcı trafiği durum bilgisini toplamak ve iletmektir. Bu durum bilgisi, servis gereksinimleri ile mevcut kullanıcı konumlarını, sağlanan servisleri, ağda mevcut erişilebilir kaynakları, servislerin ve kaynakların kullanımıyla ilgili kısıtlamaları, vb. konuları kapsar. Ağla ilgili durum bilgisi, ağdan ve dış kaynaklardan gözlem yoluyla elde edilen, ölçülen veya tahmin edilen değerlerden oluşabilir.

İkinci çekirdek yönlendirme işlevi; kullanıcı ve ağ durum bilgisine dayanarak en uygun (optimal) yolları üretmek veya seçmektir. Uygun yollar, tüm kullanıcı isteklerini ve ağ kaynaklı servis kısıtlamalarını karşılayan yollardır. En uygun yollar ise, belirli bir performans kriterine göre "*en iyi*" değerlere sahip olan yollardır. Ağ performans kriterlerine / hedeflerine ve servis kısıtlamalarına bağlı olarak yol tespiti

/ seçimi çoğu kez yoğun bir işlem kapasitesi gerektirir. Mantıklı bir süre içinde kabul edilebilir sonuçlar üretmek için '*buluşsal* (*heuristic*) *yöntemler'* gerektirebilir.

Çekirdek yönlendirme işlevinden sonuncusu, seçilen yollar üzerinden kullanıcı trafiğini iletmektir.

4.3.2 Statik yönlendirme işlemi

Yönlendirme sistemlerinin tamamına yakını ağdaki ve kullanıcı trafiğindeki değişimlere benzer bir şekilde cevap verir. Bununla beraber, yönlendirme sistemleri, cevap verdikleri durum değişimlerinin türlerine ve cevap hızlarına göre değişim gösterir. Yönlendirme sistemlerinin ağ ve kullanıcı trafiğindeki değişimlere karşı cevap vermesi gerekmektedir. Verilen cevaba, cevap hızlarına, cevap verdikleri değişimlerin türüne, vb. etkilere göre değişen dinamik yönlendirme sistemleri bulunmaktadır.

Kullanıcıların ve ağın mevcut durumundan bağımsız olarak ve sabit bir yapıya sahip yönlendirme olan statik yönlendirme sistemleri, gerçek kullanıcı ve ağın davranışından daha çok beklenen davranışa dayanır. Birçok statik yönlendirme sisteminde, yönlendirme ağ tasarımının tümleşik bir parçasıdır ve bu nedenle tekrar yönlendirme nadiren yapılır. Bir ağ tasarımının hedefi, belirli bir ağ topolojisini, yönlendirme yoluyla bilinen ağ şartları altında kullanıcı trafiğini destekleyebilen minimal ekipman masrafıyla üretmektir. Ağ tasarımı, girişlerin kullanıcı trafik tahminleri, ağ performans gereksinimleri ve ağ maliyet sınırlamaları olduğu bir tümleşik optimizasyon problemidir. Çoğu kez, uzun vadeli kullanıcı trafiği (trafik tahminlerinde kullanmak için) ve ağ yükü (ağ tasarımının performansının değerlendirilmesinde kullanmak için) ölçümlerine dayanır. Ağ tasarımı, genellikle, bu ölçümleri işlemek ve kabul edilebilir tasarımlar üretmek için yoğun bir çevrimdışı (offline) hesaplama gerektirir. Bu nedenle kullanıcı hizmet taleplerinde uzun vadeli değişimlere cevap olarak nadiren yerine getirilir. Statik yönlendirme, trafik iletiminden farklı şekilde görsel olarak gerçek zaman aktiviteleri ile ilgilenmez ve bu nedenle ağın içinde herhangi bir hesaplama aracına gerek duymaz. Statik yönlendirme sistemleri kategorisinde olan yarı-statik yönlendirme sistemleri, aynı zamanda istisnai olaylara (link ve anahtar hataları gibi) cevap olarak veya

nispeten uzun zaman aralıklarında trafik iletimini değiştirir. Ağ trafik yönlendirmesinin elle güncellenmesi, çoğu kez yönlendirme işleminde değişimlerin seyrek olduğu durumlarda statik ve yarı-statik yönlendirme sistemlerinde yeterlidir.

Yönlendirme sistemlerinin bir kısmı 'yarı statik' olarak kabul edilebilir. Bu sistemlerde yönlendirme işlemi sadece istisnai olaylara (link ve anahtar hatası) ve / veya uzun bir zaman aralığındaki değişimlere karşı gerçekleştirilirken, diğer yönlendirme sistemleri son derece dinamik olarak kullanıcı ve ağ durumundaki algılanan değişimlere karşı gerçek zamanlı olarak trafiğin yönlendirilmesi işlemlerini gerçekleştirir.

Yönlendirmenin ağın veya kullanıcı trafiğinin mevcut durumundan bağımsız olarak yapılandırıldığı ve yönlendirmenin sabit kaldığı statik yönlendirme, yaygın bir kullanıma sahip değildir.

4.3.3 Dinamik yönlendirme işlemi

İletişim ağlarında oluşan çağrı hatalarının sayısının her şart altında en aza indirgenmesi gerekmektedir. Bir mesajın bir düğümden bir başka düğüme iletilmesinde genellikle birden çok muhtemel yol bulunması nedeni ve ağdaki tıkanıklıkların önlenebilmesi amacıyla dinamik / adaptif yönlendirme algoritmalarının geliştirilmesi gerekliliği ortaya çıkmaktadır. Adaptif yönlendirme algoritmaları kullanılarak çağrılar daha az tıkanığa veya boşta kapasiteye sahip düğümlere yönlendirilebilir. Ağda ani bir hareketlilik olması veya düğümlerden birisinin büyük bir boyutlu çağrıların hedefi / kaynağı olması durumunda, bu yönlendirme işlemi büyük bir önem kazanır.

Kullanıcı ve ağ durumundaki algılanan değişimlere gerçek zamanda adapte olarak trafik yönlendirilmesini güncelleyen dinamik yönlendirme sistemleri, sadece link ve anahtar arızasını değil aynı zamanda kullanıcı trafiği ve ağ kaynaklarının mevcudiyetindeki dalgalanmaları da kapsar. Dinamik yönlendirme sistemleri, kullanıcı trafiği ile ağ performansını ölçmede talep edilen yolları hesaplamak için mevcut kullanıcı trafiğini ve ağ durumunu dikkate alarak, küçük kontrol paketlerinin (gezgin görevli) ağ içinde aktif rol almasına dayanır [4]. Bu nedenle, dinamik

yönlendirme gerçek zamanlı bilgi edinme ve kontrol kararları için bellek ve ağ içine yerleştirilmiş hesap kaynakları gerektirir.

Dinamik ortamlarda gerçekleştirilen yönlendirme işleminde farklı durumlar ortaya çıkar. Hızlı durum değişimlerini yakalayabilen dinamik bir yönlendirme sistemi, gerekli ağ kaynaklarının niceliğinden dolayı her zaman pratik olmayabilir. Durum değişimleri küçük olduğunda, yönlendirme sistemi bu değişimlere cevap vermeye ihtiyaç duymayabilir. Ek olarak, mevcut durum değişikliklerinden daha çok istatistiksel olarak hesaplanan ve tahmin edilen durum değişimleri ile ilgili yönlendirme kararlarına dayanarak kabul edilebilir yolları koruyabilir. Durum değişimleri mevcut yolların kabul edilemez seviyede olmasına neden olduğunda, yönlendirme sistemi birçok durumda bu değişimlere adapte olmaya çalışmalıdır. Ancak, ağdaki durum değişimlerinin sıklığı, yönlendirme sisteminin minimum cevap gecikmesinden daha az olması durumunda, daha düşük dinamizme sahip bir yönlendirme sistemiyle daha iyi performans elde edilebilir. Ağ ortamları için yönlendirme sistemlerinin tasarlanmasında, karmaşık maliyet ve performans değişimlerinin göz önünde bulundurulması gereklidir.

Bu tezde tanımlanan ve gerçeklenen yönlendirme sistemlerinin çoğu dinamik yönlendirme kategorisindedir. Belirli bir yönlendirme sistemi için uygun bir dinamizm derecesi birkaç faktöre bağlıdır: İşlem yapma gücü, bellek, iletim işlevlerini desteklemek üzere mevcut taşıma kaynakları, ağ ve kullanıcı trafik durumundaki değişimlerin sıklığı, seçilen yollar, gerçek durum ile beklenen durum arasındaki uyuşmazlıktan doğan performans düşümü, montajda ortaya çıkan cevap gecikmesi veya sınırlamalar, yayılım ve durum bilgisiyle hareket etme, vb.

4.3.4 Yük dengeleme

Sistemin yükünün sistemde bulunan birimlere eşit bir şekilde dağıtan ve mesaj kayıplarını en aza indirgeyen çağrı yönlendirme algoritmalarının uygulanması sonucunda sağlanan *'yük dengeleme'* işlemi, dinamik yönlendirme ile sıkı bir ilişkiye sahiptir. Yük dengeleme, mesajların boş kapasiteye sahip ağ kısımları üzerine yönlendirilmesiyle gerçek veya potansiyel yerel tıkanıklığının önlenmesini sağlamaktadır. Dinamik yönlendirme ve yük dengeleme, statik ve yarı statik

yönlendirmeden daha fazla hesaplama kaynağına gereksinim duyar. Dinamik yönlendirme, kullanıcı trafiğini, ağ durumunu, ağ performansını ölçmek, düğümler arasında en kısa yolları hesaplamak, vb. işlemlerin yapılmasında ağ içerisinde bütün düğümlere dağıtılmış ve düğümler arasında hareket edebilen varlıkların / nesnelerin / görevlilerin sayesinde gerçekleştirilir.

4.3.5 Akış kontrolü

Yönlendirme, trafik akış kontrolü, kullanıcı trafiği ve ağın kendisi birbirini etkileyen ayrı dinamik sistemlerdir [3]. Özellikle yönlendirme ve akış kontrolü, ağ ve kullanıcı trafiği tarafından tanımlı dinamik bir ortamda çalışan ağ kontrol işlemleridir. Kullanıcı trafik biçimleri (patterns), hacimleri (volumes), servis gereksinimleri, kaynak uygunluğu, kullanım sınırlamaları, vb. parametreler bir ağ içerisinde yolların elverişliliğini ve uygunluğunu belirler. Yönlendirme kullanıcı trafiği ile ağ tarafından konulan hedefler ve kısıtlamaları karşılayan yolları seçerek, hangi ağ kaynaklarının hangi kullanıcı trafiği tarafından kullanılacağını tanımlarken, akış kontrolü kullanıcı trafiğinin ağ kaynaklarını nasıl kullanacağını belirler.

Akış kontrolünün yönlendirmeden farklı bir konu olması nedeniyle bu çalışmada detaylı olarak ele alınmayacaktır. Bununla beraber, yönlendirme ile akış kontrolünün birbirine bağlı konular olduğu unutulmamalıdır. Yönlendirme yolları performans karakteristiklerine göre seçerken, akış kontrolü performans karakteristiklerini kabul edilebilir bir aralıkta tutmaya çalışır. Diğer bir değişle, akış kontrolünün hedefi; ağ çıkışını (throughput) aşırı derecede kısıtlandırmaksızın ağ tıkanıklıklarını önlemek için kullanıcı trafiğinin kabul edilip edilmeyeceğini ve trafiğin ne zaman iletileceğini belirlemektir. Akış kontrolü ağ içinde, kullanıcı oturumlarını, yolları ve linkleri içeren çeşitli seviyelerde uygulanabilir. Akış kontrolü genelde yönlendirmeden daha küçük bir zaman diliminde çalışır.

4.3.6 Yönlendirme sistemini / işlevini yayma (decentralizing)

Yukarıda özetlenen temel yönlendirme işlevi; her biri merkezi veya merkezi olmayan (çok merkezli) bir şekilde uyarlanabilir. Çok merkezliliğin derecesi

arzulanan dinamizme, sağlamlığa ve yönlendirme sisteminin yönetilebilirliğine bağlıdır.

Merkezi uyarlamada, belirli bir yönlendirme işlevini tek bir birim yerine getirir. Yönlendirme algoritmaları, kontrol otoritesi tarafından belirli aralıklarla güncellenen anahtarlama istasyonlarındaki yönlendirme tablolarıyla birlikte merkezi bir birimde bulunur. Merkezi prosedürler, işlevsellik uygulama içerisindeki hataları düzeltme, yalıtma vb. yönetim işlevlerini basitleştiren tek bir varlıkta durduğu için, yönetimi kolaydır. Yönlendirme işlevini gerçekleştirmek için özel kaynaklar gerekmesi durumunda (örneğin, tümleşik optimizasyon problemlerini çözme yeteneğinde olan hesap mekanizmaları), gerekli kaynaklar maliyetleri düşürecek şekilde bir merkezde / bir noktada toplanabilir. Bununla beraber aşağıdaki özelliklerin dikkate alınması gereklidir:

i) Kontrolör genellikle sistemin her parçasından kontrol birimine iletişim bağlantılarını gerektirerek bütün sistem hakkında güncel bilgiye gereksinim duyar.

ii) Merkezi kontrol mekanizmaları, işleme ve iletişim maliyetlerinin sistem boyutuyla ilişkili olarak artması nedeni ile büyük ölçekli sistemler için uygun değildirler.

iii) Kontrolörde oluşan hatalar çoğu kez bütün sistemin çökmesiyle sonuçlanabilir.

iv) Merkezi olarak kontrol edilen sistemlerin tek bir otorite tarafından sahiplenilmiş olmasına gereksinim duyulabilir.

Merkezi yönlendirme uygulamalarının dezavantajı; yönlendirme işlevini yapan merkezi elemanın / birimin arızalanması veya ağdan ayrılması durumunda, yönlendirme işlevinin bu süre boyunca askıya alınmasıdır. Bir yönlendirme işlevinin durum değişimlerine cevap verebilirliği; yönlendirme işlevini sağlayan birim üzerindeki yüke, o işleve ihtiyaç duyan ağ bölümüne ve birimler arasındaki

mesafeye bağlıdır. Bu nedenle, yönlendirme işlevinin bir tek birimde toplanması cevap verebilirliği kısıtlar.

Dağıtık sistemlerin doğası gereği dinamik, karmaşık, ve stokastik olmaları ve bu sistemlerin davranışlarının bir merkezi kontrol sistemi ile kontrol edilebilecek formata dönüştürülmesi tahmin / ifade edilemeyen faktörleri ortaya çıkarır. Buna karşın, yönlendirme işlevinin merkezli olmayan bir yapıda uyarlanması durumunda, ağ içerisinde birden fazla varlık / görevli bağımsız olarak işlev görür ve bilgiyi karşılıklı değişir ve yönlendirme için gerekli hesaplama yükü birden fazla düğüme dağıtılır.

Çok merkezli yönlendirme uygulamasında, birden fazla gezgin görevli / birim yönlendirme işlevini gerçekleştirir. Yönlendirme işlevinin çoğaltılması durumunda, her bir eleman bağımsız olarak işlevi gerçekleştirir. Yönlendirme işlevinin dağıtılması durumunda, benzer görevliler / birimler bağımsız olarak işlevin parçalarını gerçekleştirmelerine karşılık, sonuçları karşılıklı değiştirerek eksiksiz bir işlevin sağlanmasına yönelik için işbirliği yapmalıdırlar. Benzer birimler arasındaki işbirliği eşzamanlı olabilir veya olmayabilir.

Çok merkezli yönlendirme, sistemin / ağın yönetimini karmaşıklaştırsa da, birçok avantaja sahiptir. İşlevin birden fazla birim üzerine yayılması yönlendirme sisteminin hata toleransını artırır. Yükü birden fazla birim arasında yayarak ve işlevselliği ona ihtiyaç duyan ağ bölümlerine yakın yerleştirmeyi sağlayarak cevap gecikmesini düşürür. İşlevi birden fazla birim üzerine yaymak, herhangi bir tek elemanda gerekli yönlendirme sistemi kaynaklarının büyüklüğünü düşürür ve yönlendirme sisteminin ağ boyutunda artarak büyümesini sağlar. Bu tezde tanımlı yönlendirme sistemlerinin çoğu dağıtık prosedürler olarak uyarlanmıştır.

4.3.7 Devre anahtarlama ve paket anahtarlama iletim yöntemleri

Haberleşme ağları, anahtarlama teknolojisi referans alınarak ‘devre anahtarlamalı’ ve ‘paket anahtarlamalı’ şeklinde iki sınıfa ayrılabilir. Bu sınıflandırma, ağ anahtarlama teknolojisi yanı sıra ağın geliştirilme hedefini de farklılaştırır.

i- Devre anahtarlamalı iletim: İletim süresi boyunca iletişim kanalının mesaja tahsis edildiği bir iletişim türüdür. Başlangıçta, ses trafiğini iletmek için tasarlanan devre anahtarlamalı ağlar, sıralamaya ve gecikme işlemlerine duyarlıdırlar. Devre anahtarları ağlar başlangıçta çoğullamaya (multiplexing) dayanan analog aygıtlar kullanmalarına rağmen, süreç içerisinde popüler bir sayısal anahtarlama yöntemi olarak zaman-bölmeli (time-division) çoğullama yöntemi kullanılmaya başlamıştır. Devre anahtarlama yönteminde, kaynaktan hedefe iletim yolu boyunca iletim ve anahtarlama kaynaklarının bir çağrıya ayrılması nedeniyle mesajın iletimi sırasında kuyruk gecikmeleri önlenir. İletilen mesajın tüm kaynakları kullanıp kullanmadığı dikkate alınmaksızın, bir mesaj iletimi boyunca tüm kaynaklar ayrılmış olarak kalır. Ayrılmasına karşılık boşta olan / kullanılmayan anahtarlama ve iletim kaynakları diğer mesajların iletimi için kullanılamaz.

Devre anahtarlamalı ağların çoğu telefon şirketleri tarafından ticari amaçlı olarak geliştirilmiş halka açık ağlardır. Telefon şirketleri, ağların tasarlanmasında ve kendi trafik yönlendirmesinin yönetilmesinde önemli miktarda kaynak harcayarak güvenilir servisler sağlamayı amaçlarlar. Bu amacı gerçekleştirirken, minimal tasarım ve yönetim maliyetleriyle, olumsuz durumlarda bile (dengesiz yük, birimlerin hizmet dışı kalması, vb.) iletişimin devamlılığını sağlamayı hedeflerler.

ii- Paket anahtarlamalı iletim: Mesajların gönderilmeden önce paketlere bölündüğü bir iletim sistemidir. Her paket daha sonra farklı yollar izleyerek hedefine ulaştırılır. Bütün paketler hedefe ulaştığı zaman tekrar derlenerek orijinal mesaj meydana getirilir. Başlangıçta, gecikmeye duyarsız veri trafiğini taşımak için tasarlanan paket anahtarlamalı ağlar, anahtarlama ve iletim kaynaklarını talep edilen trafiğe tahsis etmek üzere istatistiksel çoğullama kullanırlar. Paket anahtarlamalı ağlarda, kullanılmayan kaynakları kullanmak isteyen birimler tarafından kullanılmasına izin verileceği garanti edilir. Ağ trafiğinin mevcut kaynakları kullanma isteği, sunulan trafik yükü kaynağın kapasitesini aştığında paketlerin kuyruğa yerleştirilmesini, hatta atılmasını zorlayabilir.

Paket anahtarlama tekniği yakın zamana kadar çoğunlukla belirli amaçlar için geliştirilen özel ağlarda (mesela; havayolu rezervasyon sistemleri, bankalar, vb.) veya araştırma / altyapı projeleri olarak geliştirilen halka açık ağlarda (ARPANET, NSFNET, vb.) kullanılmaktaydı [6]. Özel ağlar, özel ortamlar veya kullanımlar için optimize edilmiş yönlendirme stratejilerinin geliştirilmesi gerekliliğini ortaya çıkardı. Diğer taraftan, araştırma ağlarının davranışı, önceden tahmin edilmesi güç deneysel adaptif yönlendirme stratejilerinin gelişimini motive etmiştir. Bu çalışma, paket anahtarlamalı ağlarda yönlendirme konusunun modellenmesi ve simülasyonuna odaklanmaktadır.

Devre ve paket anahtarlamalı iletim yöntemleri bazen b*ağlantıya yönelik (connection-oriented)* ve *bağlantısız (connectionless)* iletim yöntemleri olarak adlandırılır [7]. Bağlantıya yönelik yönlendirme; kullanıcı trafiğinin iletiminden önce, yol boyunca bütün anahtarlarda yönlendirme algoritmalarının / emirlerinin bulunmasını gerektirir. Bağlantısız yönlendirme ise, kullanıcı trafiğinin yol üzerindeki anahtarlar / ara bağlaşım cihazları tarafından bağımsız olarak yorumlanabilen yönlendirme bilgisinin taşındığı iletişim şeklidir.

4.3.8 Yönlendirme işleminin geliştirilmesi ve yeni yönlendirme algoritmaları

Birbirinden bağımsız ve paralel olarak gelişen devre anahtarlamalı ve paket anahtarlamalı sistemler, süreç içerisinde bir noktada toplanmaya başlamışlardır. Bu birleşmeye yönelik gelişim iki temel dürtünün sonucudur: kullanıcı hizmet talepleri ve yüksek hızlı iletim / anahtarlama teknolojileri.

Güçlü, kullanımı kolay, taşınabilir ve ucuz kişisel bilgisayarların sayısındaki artış, global olarak erişilebilir haberleşme ağlarıyla birlikte, karmaşık iletişim servislerine sahip büyük ağlar yanında artan bir kullanıcı topluluğunu ortaya çıkarmıştır. Ortaya çıkan kullanıcı hizmet gereksinimleri; düşük maliyet, yüksek kalite, kullanıcı konumundan bağımsız bir şekilde erişilebilir dağıtık çoklu ortam (multimedia) hizmetleri (ses, veri ve video) olarak özetlenebilir. Temel global iletişim ağları,

özellikle telefon ağları ve internet, ortaya çıkan gereksinimlere aşağıda özetlenen teknolojiler ile cevap vermektedir.

Telefon hizmet sağlayıcılarının kullanıcı taleplerini karşılamak amacıyla geliştirdikleri yeni teknolojilere örnek olarak iki yeni teknoloji verilebilir: konumdan bağımsız hizmetler sağlamak maksadıyla geliştirilen seyyar (mobile) hücresel radyo sistemleri ve çoklu ortam hizmetlerini destekleyen Genişband Tümleşik Hizmet Sayısal Ağları (Broadband Integrated Services Digital Networks – B-ISDN). Bu teknolojilerin geliştirilmesiyle birlikte, tarihsel olarak merkezi, yarı-durağan trafik yönlendirme stratejilerini kullanan telefon ağları mesaj taşıma kapasitesini ve ağ dayanıklılığını maliyet olarak en uygun bir şekilde arttırmak için dinamik yönlendirme stratejilerini benimsemeye başlamışlardır.

Global bir veri ağı sistemi olarak İnternetin ortaya çıkışı, İnternet ile ilgili / İnternetin kullanıldığı bir çok alanda yeni çalışmaların yapılması gerekliliğini ortaya çıkarmaktadır. Herhangi bir hizmet garantisi olmaksızın en iyi trafik yönetimi sağlayan dağıtık ve dinamik yönlendirme stratejilerini kullanan İnternete, bir noktadan birden fazla noktaya iletişim için hizmet garantisi sağlayan, seyyar kullanıcıları ve dinamik ağları destekleyen protokol standartları eklenmesi ile birlikte ileri ağ performansı ve düşük maliyet özellikleri eklenmesine yönelik çalışmalar yapılmaktadır. Diğer bir değişle, İnternetin dramatik ilerleyişi çok büyük ağları barındırabilen, güvenilir, yönetilebilir ve güvenli yönlendirme stratejilerine gereksinim yanında bu konuda yeni çalışmalar / araştırmalar yapılmasını ortaya çıkarmaktadır.

Ortaya çıkan yeni iletişim ihtiyaçlarının ve farklı hizmet taleplerinin karşılaması için en uygun ağ teknolojilerinin hem devre anahtarlama hem de paket anahtarlama kapasitelerine sahip ağlar olacağı açıktır. Hızlı paket anahtarlama ve fiber optik ağlar gibi geliştirilmekte olan yüksek hızlı ağ teknolojileri devre anahtarlama ve paket anahtarlama teknolojilerinin karışımı olan yeni iletişim teknolojilerini kullanıyorlar.

Sonuç olarak, küresel kullanıcı bağlantıları tek bir homojen ağ katmanı yerine birden fazla bağımsız birim / görevli ile idare edilen, farklı hizmetler sunan, farklı

anahtarlama teknolojileri kullanan, birbirine karşılıklı olarak hizmet sağlayan sistemler tarafından sunulacaktır. Böyle bir ortamda çalışan bir yönlendirme sistemi için temel problem; ağ boyunca yönlendirme trafiğinde kullanılan büyük hacimli bilgilerin verimli dağıtımı, idaresi ve sentezi olacaktır. Bu problemin yakın gelecekte ortaya çıkacak büyük, heterojen ve dinamik ağların yönetiminde karşılaşılacak önemli iletişim problemlerinden birisi olacağı beklenmektedir [3].

4.4 Gezgin Görevli (Mobile Agent) Tabanlı Yönlendirme ve Ağ Yönetimi

Günümüzün iletişim ortamlarında heterojen ağların kullanımına doğru artan eğilim, ağ operatörlerinin daha kapsamlı bir bilgiye ve daha iyi bir eğitime sahip olmalarını gerektirmektedir [3][4][5]. Farklı ağların yönetimi, büyük boyutta verinin ağdan toplanmasını ve yönetim aktivitesine başlanmadan önce verinin analiz edilmesi gereksinimini ortaya çıkarmaktadır. Diğer taraftan, günümüz ağ ve internet kullanıcılarının artan bir güvenilirlik ve hizmet kalitesi gereksinimine ve beklentisine sahip olmaları, gereksinimleri ve beklentileri karşılayacak olan '*yazılım görevlileri' (software agents)* konusunun gelişimine bir etki / katkı yapmaktadır [13].

Geniş ölçekli ağlarda ortaya çıkan gereksinimler ve hizmet beklentilerini karşılamak amacıyla geliştirilen 'yazılım görevlileri' ve yazılım görevlilerinin zeka yetenekleri hakkında yapılan araştırmanın çoğu *dağıtık yapay zekadan (Distributed Artificial Intelligence - DAI)* gelmektedir [36]. DAI, yapay zekanın (Artificial Intelligence - AI) *'çoklu görevli sistemlerine' (Multi-Agent Systems - MAS)* uygulanmasından türetilmiştir [37]. Çoklu görevli sistemlerde, merkezi ve sistemin tüm zekasını / işlevini gerçekleştiren çok büyük bir uygulamanın yerine bir takım küçük sistemler veya görevliler bir problemi çözmek için ortak bir çaba gösterirler. Bu yöntem, büyük sistemin sadece küçük parçalara bölündüğü anlamına gelmez. Bu yöntemde, her biri problemin belirli bir yönüne odaklanabilen birkaç uygulama bir iletişim sistemiyle birlikte bağlanarak, bakış açılarının karşılıklı değişilmesini sağlarlar. Problemin çözümüne yönelik olarak sonuçları toplamak, ilerleme stratejileriyle birlikte gündeme gelir. Bu problem çözme tekniği '*dağıtık problem çözümü'* olarak adlandırılır ve işbirliği yapan sistemlerin her biri '*görevli' (agent)* olarak adlandırılır

[38]. AI'da bir görevli, *doktrinleri, istekleri ve niyetleri (beliefs, desires, intentions - BDI)* ile karakterize edilir [39].

Görevli tabanlı sistemler herhangi bir istemci / sunucu teknolojisi ile uyarlanabilse de, istemci ve sunucu arasında açık bir fark olmaması nedeni ile klasik istemci / sunucu sistemlerinden farklıdır. Tüm görevliler, tasarımcı tarafından (bir insan veya yönetici bir birim) dinamik olarak atanmış bir role göre hesaplama işlemine katılır.

4.4.1 Yazılım görevlisinin tanımı

Herkes tarafından kabul edilen bir yazılım görevlisi tanımı yapmak zor olmasına karşılık, yazılım görevlisi; başkaları adına hareket eden, otonom, proaktif, reaktif, öğrenme, işbirliği yapma ve hareket etme yeteneğine sahip bir hesaplama varlığı olarak tanımlanabilir [40][41]. Bu görevli karakteristikleri, '*temel görevli modeli'* olarak adlandırılır.

Gezgin görevli, konumlar arası hareket edebilen bir yazılım görevlisidir. Bu tanımdan, bir gezgin görevlinin aynı zamanda temel görevli modeliyle de karakterize edilebileceği sonucu çıkartılabilir. Temel modele ek olarak, bir yazılım görevlisi *yaşam döngüsü modeli, hesaplama modeli, güvenlik modeli* ve *iletişim modelini* tanımlar [41]. Ayrıca bir gezgin görevli '*kılavuz' (navigation)* modeli ile de karakterize edilebilir.

Gezgin görevlileri kullanmak için, bir sistem hareket edebilme çerçevesi ile birleştirilmelidir. Çerçeve, kılavuz modelini içererek görevli modellerinin tümünü destekleyen imkanları sağlamalıdır. Yaşam döngüsü modeli için görevlileri oluşturmak, ortadan kaldırmak, başlatmak, askıya almak, durdurmak, vb. hizmetlere gereksinim duyulur.

- **Hesaplama modeli;** bir görevlinin veri işleme ve iş prosedürü (thread) kontrolünü içeren hesaplama yetenekleridir.

- **Güvenlik modeli;** görevlilerin hem ağ kaynaklarına erişebilme yöntemlerini hem de ağdaki görevlilere erişim yöntem ve yollarını tarif eder.

- **İletişim modeli;** görevliler arasındaki ve görevli diğer varlıklar (ağ gibi) arasındaki iletişimi tanımlar.

- Farklı konumlarda bulunan iki hesaplama varlığı arasında bir görevliyi (durumuyla birlikte veya durumu olmadan) taşıma işleminde gerçekleştirilen bütün işlevler **kılavuz modeli** ile halledilir.

Gezgin görevli sistemleri her bir aygıt üzerinde artan bellek gereksinimlerini, işleme ve erişim gecikmelerini kapsayan bir dizi maliyete neden olur. Buna rağmen, gezgin görevli sistemleri için altyapı oluşturan teknolojiler ve sistemler hızla gelişmektedir. Örneğin, birçok gezgin görevli uygulaması için altyapı sağlayan Java Sanal Mekanizması (Java Virtual Machine – JVM) tümleşik sistemler için gerektirdiği maliyetin çok az olması sebebiyle çok küçüktür [42].

Gezgin görevlilerin boyutu, gerçekleştirdikleri işlem ile doğrudan ilişkilidir. Bu çalışmada tasarlanan 'oğul zekası' uygulamalarında görevlilerin boyutları son derece küçüktür. Diğer taraftan yapılandırma ve tanılama (diagnostic) görevlileri, karmaşık algoritmalar ve probleme neden olan mekanizmaları / olayları çözmeye ihtiyaç duydukları için çok büyük boyutta olabilirler. Yerel ortama ve ihtiyaçlara bağlı olarak genişletilebilen minimum bir işlevselliği taşımaları yanında görevliler, çalışma esnasında herhangi bir yerde gerekli olan kodu ağ üzerinden indirerek kendi kapasitelerini genişletebilirler. Bu yetenek kod değişkenliği yoluyla sağlanır.

4.4.2 Gezgin görevlilerin üstünlükleri

Gezgin görevlilerin kullanımı, diğer görevlilerin kullanımlarına göre bir takım üstünlüklere sahiptir. Sanal olarak gezgin görevliler tarafından yapılabilen herhangi bir görev aynı zamanda durağan nesneler tarafından da yapılabildiği için diğer teknolojilerin (uzak nesneler gibi) kullanılamayacağı manasına gelmez. Buna karşılık, klasik çözümler düşük verimli, kullanımı zor ve yavaş olabilir.

Gezgin görevli teknolojisi esnek iletişim sistemleri yönetimi için bir çözüm sağlar [4][5][13][14][15][16][20]. Gezgin görevliler yerel olarak düğümler arasında göç

ederek her bir düğümdeki donanımları gözlemler ve kontrol ederler. Gezgin görevli-tabanlı ağ yönetimi istemci / hizmet birimi gibi klasik yöntemlerle karşılaştırıldığında belirli üstünlüklere sahiptir. Örneğin gezgin görevliler, ağ trafiğini azaltırlar ve bağlantısız işlemleri kolayca desteklerler. Bir ağ sistemi içindeki dinamik yerleştirme yanında yeni veya mevcut işlevlerin konfigürasyonu son derece önemli görevlerdir. Bu görevler özellikle potansiyel olarak zaman aşımına uğramış sistemlerin verimli bir şekilde güncellenmesine olanak tanır. Gezgin görevliye sahip sistemlerin tercih edilmesi, birçok ağ yönetim aktivitelerinin sürekli izlenmesi için yöneticiye olan ihtiyacı ortadan kaldırır (yazılım kurulması ve yükseltilmesi ve ağın periyodik olarak dinlenmesi). Bu teknolojiyi ağ yönetim işlemelerine uygulamak için gerçekleştirilen çalışmalar bulunmaktadır.

Aşağıda klasik istemci / sunucu modellerinin yerine gezgin görevlilerin kullanımından ortaya çıkan faydalar sıralanmaktadır [15]:

Verimlilik kazanımları: İşlemci (CPU) kullanımı oldukça düşüktür. Bir gezgin görevli bir anda sadece bir düğüm üzerinde çalışması nedeni ile, diğer düğümler gerekli olana kadar bir görevli çalıştırmazlar.

Alan kazanımları: Bir gezgin görevli herhangi bir anda sadece bir düğüm üzerinde olduğu için kaynak tüketimi sınırlıdır. Buna karşılık statik birden fazla sunucu her konumda işlevselliğin çoğaltılmasını gerektirir. Gezgin görevlilerin işlevselliği kendileriyle birlikte taşırlar, dolayısıyla çoğaltılmaya gerek yoktur. Java teknolojisinin sunduğu *'uzak nesneler' (remote objects)* benzer üstünlükler sağlasalar da, ihtiyaç duyulan özel yazılımın (middleware) maliyeti yüksek olabilir.

Ağ trafiğinde düşüş: kullanılan kodun çoğu kez işlediği veriden daha küçük olması nedeni ile veri kaynaklarına gezgin görevlilerin transferi, veri transferinden daha az bir trafik oluşturur.

Eşzamanlı olmayan özerk (autonomous) etkileşim: Gezgin görevliler, görevlendirilen varlık aktif kalmasa bile belirli görevler yerine getirmek için görevlendirilebilir.

Gerçek-zamanlı sistemlerle etkileşim: Bir gezgin görevliyi gerçek-zamanlı bir sisteme yakın kurmak ağ tıkanıklığı tarafından neden olunan gecikmeleri önleyebilir. Ağ yönetim sistemlerinde görevlilerin genellikle donanıma yakın konumlandırılmaları nedeni ile bu üstünlük diğerleri gibi açık olmayabilir.

Sağlamlık ve hata toleransı: Bir dağıtık sistemin arızalanması durumunda, gezgin görevliler ilgili alanlarda belirli servislerin erişilebilirliğini artırmak üzere kullanılabilir. Örneğin, hata bulma ve onarım görevlilerinin yoğunluğu artırılabilir. Bir tür görevli yönetimi sistemi, görevlilerin üstlendikleri görevlerini yerine getirip getirmediklerini test etmek için gereklidir.

Farklı ortamlar desteği: Gezgin görevliler ana bilgisayarlardan (hosts) hareket kabiliyetini sunan bir çerçeve ile ayrılmışlardır. Eğer çerçeve varsa, görevliler herhangi bir sisteme erişebilir ve görevlerini yerine getirebilir. Daha öncede ifade edildiği gibi gezgin görevli sistemlerine altyapı oluşturan Java Sanal Mekanizmasının (JVM) yetenekleri ve çalışma kapasitesi her geçen gün gelişmektedir.

Hizmetlerin çevrimiçi genişletilebilirliği: Gezgin görevliler hizmetleri gerçekleştiren / sağlayan uygulamaların kapasitesini genişletmek için kullanılabilirler. Bu özellik, sistemlerin son derece esnek tasarlanmalarını sağlar.

Kolay gelişim yapısı: Gezgin görevlilere dayanarak dağıtık sistemleri geliştirme nispeten kolaydır. Zor olan kısım hareket kabiliyeti çerçevesidir ve bu belirlendikten sonra gerçekleştirilecek uygulamaların oluşturulması kolaylaşır. Alan olgunlaştığında, görevliler için yüksek seviyeli, hızlı uygulama geliştirme ortamlarına ihtiyaç duyulur. Nesneye yönelik programlama için gelişmekte olan araçlar, hareket kabiliyetini kolaylaştıran bir takım işlevleri barındıracak olan görevliye yönelik gelişim ortamlarını geliştirmektedir.

Kolay yazılım yükseltmeleri: Gezgin bir görevli, istenildiğinde sanal olarak karşılıklı değiştirilebilir. Buna karşılık, uygun servis kalitesi (QoS) düzeyinin korunması istendiğinde sunucuların değiş tokuş işlevleri karmaşıklaşır.

4.5 Yönlendirme Algoritmaları

Yönlendirme algoritmaları, kullanıcı trafiği ile ağ tarafından belirlenen hedefler ve sınırlamaları karşılayan yolları seçer. Yönlendirme algoritmalarında istenen belirli özellikler bulunmaktadır: Doğruluk, basitlik, sağlamlık, kararlılık, açıklık ve optimallik.

Bir ağın yükünü dengelemek için gösterilen çabaya bağlı olarak yönlendirme algoritmaları yukarıda ifade edildiği şekilde statik - dinamik veya merkezi - dağıtık olarak sınıflandırılabilir. Daha önceki kısımlarda belirtildiği gibi, merkezi algoritmalar genellikle ölçeklenebilirlik problemlerine sahip olmaları yanında, merkezi kontrol istasyonunda oluşan herhangi bir hata durumunda ağın çökmesi gibi sakıncalara sahiptir. Adaptif yönlendirme algoritmaları ise, düğüm hatalarından ortaya çıkan tutarsızlıklara, dairesel yollara / döngülere ve kararsızlığa neden olan potansiyel osilasyonlara, vb. problemlere sahiptir [7].

Daha öncede ifade edildiği gibi, statik yönlendirme algoritmaları yönteminde, belirli yönlendirme tabloları ağ çalışmaya başlamadan önce oluşturulur. Bir iletişim ağında statik yönlendirme düğümler arasındaki en kısa yolu bulmaya karşılık gelir. Burada kullanılan ölçüt iki yönlendirici arasındaki hoplama sayısı, fiziksel uzaklık, iletim gecikmesi, vb. olabilir. Klasik *'Dijkstra'* algoritması en kısa yol problemini çözer ve yönlendirici tarafından gelen paketleri kendi hedeflerine doğru iletmek için kullanılan yönlendirme tablolarını oluşturmak için kullanılabilir [43]. Bu yöntem sadece ağ topolojisine önem verir.

Dinamik yönlendirme algoritmaları, belirli bir ağ yüküne dayanarak çalışma anında yönlendirme tablolarının oluşturulması ve güncellenmesi esasına göre çalışır. Bu tür algoritmalara örnek olarak; *'uzaklık vektörü' (distance vector)* ve *'link durumu' (link state)* yönlendirme algoritmaları verilebilir. Ayrıca, bu çalışmada ilgi odağı olan yeni bir yönlendirme algoritma grubu olarak *'oğul zekası'* yöntemini kullanan algoritmalar bulunmaktadır. Aşağıdaki kısımlarda algoritmalar açıklanmaktadır.

4.5.1 Uzaklık vektörü yönlendirme algoritması

Ağ kaynakları ve hedefleri arasında en az maliyetli (cost) yolları üreten en kısa yol algoritmaları, paket anahtarlamalı ağlar için en yaygın kullanılan yol üretim algoritmalarıdır [3]. En kısa yol algoritmaları arasında, dinamik programlama kavramlarına dayalı algoritmaları oluşturan uzaklık vektörü sınıfı en yaygındır. Uzaklık vektörü algoritmaları başlangıçta 'ARPANET' içinde kullanılmıştır ve günümüzde İnternette işlevini sürdürmektedir. Uzaklık vektörü algoritmaları günümüzde birbirine bağlı ağlarda kullanılan standart yönlendirme prosedürlerinin bir çoğuna temel teşkil ederek geniş bir alanda kullanıma sahip olmuştur. Bu algoritmaların tercih edilmesinin nedeni, dağıtık ve asenkron çalışmaya izin vererek basit uyarlamalardaki esneklikleri ve yolları üretmek / seçmek için sadece yerel olarak erişilebilir yönlendirme bilgisini kullanmalarıdır.

Uzaklık vektörü yönlendirme algoritmaları ARPANET ile başlayarak yirmi yıldan fazla paket anahtarlamalı ağlarda kullanılmaktadır. Basit, bir maliyet hesaplama işlemini kullanarak en az maliyetli yollar belirlenir ve bu işlem genellikle ağdaki anahtarlama elemanlarına dağıtılır. Bir kaynak anahtarı kendi komşu anahtarlarından elde edilen maliyet hesaplamalarına dayanarak bir hedef anahtarı için yolun maliyetini hesaplayabilir. Bu nedenle anahtarlar uzak hedeflerin en az maliyetli yollarını hesaplarken, sadece komşularıyla yerel maliyet hesaplarını karşılıklı değişirler.

Uzaklık vektörü algoritması belirli bir ölçüte göre en uygun (optimal) yolları üretir [3]. Yönlendirme ölçütlerine örnek olarak; gecikme, çıkış, hata oranı, parasal maliyet, hop sayısı veya birkaç ölçütün bir fonksiyonu verilebilir. Bu bölümden sonraki bölümlerde detayları verilecek olan uygulamalarda, genellikle yönlendirme ölçütü olarak hop sayısını kullanılmaktadır. Bu uygulamalarda optimal yol en az hop sayısına sahip yol manasına gelmektedir. Uzaklık vektör algoritmaları hop sayısını kullanmanın yanında maksimum kapasite gibi ölçütleri de kullanabilir ve bu durumda maksimum yol kapasitesi, link kapasitelerinin minimumu anlamına gelir.

Temel uzaklık vektör yönlendirme algoritması çoğu kez *'Ford-Fulkerson'* yöntemi olarak adlandırılır [1]. Aynı zamanda dinamik programlama perspektifinden

Bellman denklemleriyle ifade edilebildiğinden dolayı *'Bellman-Ford'* algoritması olarak ta bilinir.

Bir iletişim ağını düğümlerden ve linklerden oluşan bir sistem olarak kabul edersek; düğümler anahtarlama elemanlarını (yönlendiriciler ve ana bilgisayarlar), linkler ise anahtarlar arasındaki bağlantıları gösterir. D_{ij}, i kaynak düğümünden j hedef düğümüne en düşük maliyete sahip yolun maliyeti ve her i - j düğüm çifti için tanımlı olması yanında, '*i'* ve '*j'* birbirine komşu iseler, bu doğrudan bir linke sahip olmaları anlamına gelir ve d_{ij} bu linkin maliyeti olur (aksi halde, $d_{ij} = \infty$ olur). Link maliyetlerinin toplandığı varsayılarak, '*i'* ve '*j'* arasındaki minimum maliyetli yol aşağıda detayları verilen 'Bellman denklemleri' aracılığıyla çözülebilir;

$D_{ii} = 0$ tüm i'ler için,
$D_{ij} = min\ (d_{ik} + D_{kj}),\ i \neq k$ için.

'Bellman-Ford algoritması' kullanılarak, h hop sayısı kadar yinelenen bir algoritma tanımlanır:

$D_{ii}\ (h + 1) = 0$ tüm i'ler için.
$D_{ij}\ (h + 1) = min\ (d_{ik} + D_{kj}\ (h)),\ i \neq j$ için.

Başlangıç şartları aşağıdaki gibi tanımlanabilir;

$D_{ii}\ (0) = 0$ tüm i'ler için.
$D_{ij}\ (0) = \infty$ $i \neq j$ için.

Bu kısımda detaylandırılan uzaklık vektör algoritmasının bir türü olan *yönlendirme bilgi protokolü (Routing Information Protokol – RIP)* detayları, modelleme ve simülasyon uygulaması Bölüm 6'da detaylı olarak ele alınacaktır.

4.5.2 Link durumu yönlendirme algoritması

Link durum yönlendirme algoritması, statik yönlendirme ve uzaklık vektörü yönlendirme gibi klasik yöntemlere bir alternatif olarak son zamanlarda popüler bir

duruma gelmiştir. Günümüzde link durum yönlendirme protokol türlerinin çoğu İnternet gibi büyük ölçekli ağlarda kullanılmaktadır. Link durumu yönlendirme algoritmalarının çekirdeğini dağıtık ve çoğaltılmış bir veritabanı oluşturur ve bu veritabanı kullanılarak veri trafiği için en uygun yollar hesaplanabilir Bu veritabanı, ağın bileşenlerini ve mevcut bağlantılarını tanımlayan bir dinamik ağ haritasıdır.

İlk link durum protokolü Bolt Beranek ve Newman tarafından 1979 yılında ARPANET için geliştirilmiştir [44]. Link durum algoritmasının daha önceki Bellman-Ford tabanlı yönlendirme protokolü üzerinde verimlilik, güvenilirlik, döngü serbestliği ve adaptasyon hızı bakımından daha önemli üstünlükleri olduğu tespit edilmiştir. Daha sonra, link durum yönlendirme protokolleri geliştirilmiş ve yeni protokoller ortaya çıkmıştır. Bunlara örnek olarak TCP/IP için OSPF (Open Shortest Path First) protokolü [1] ve OSI için IS-IS [3] verilebilir.

Her bir link durum protokolünün çekirdeğini oluşturan dağıtık ve çoğaltılmış veritabanı '*link durum veritabanı'* olarak adlandırılır [3]. Ağdaki her bir anahtarlama elemanı bu veritabanının özdeş bir kopyasını saklar ve bu veritabanı yönlendirme alanının eksiksiz bir haritasını tanımlar. Link durum veritabanını oluşturan parçalar '*link durum ilanları'* (Link State Advertisements - LSAs) olarak adlandırılır. Her anahtar, kendisinin çalışan arabirimlerini gösteren bir LSA oluşturmak ve bunu diğer düğümlere yollamaktan sorumludur.

'Akış' (flooding) olarak adlandırılan işlem yönlendirme alanı boyunca LSA'ları dağıtmak için kullanılır. Akış işleminde, bir arabiriminden LSA'yı alan bir anahtar diğer arabirimleri üzerinden LSA'yı gönderir. Akış işlemi güvenilir ve çabuk olmalıdır. Yönlendirme alanında bir değişiklik olursa (örneğin; bir link koparsa ve bir yönlendirici çalışmaz duruma gelirse) değişimi tanımlayan yeni bir LSA oluşturularak tüm anahtarlara gönderilmeli ve akış işlemi başlamalıdır. Akış işlemi esnasında tüm anahtarlar aynı veritabanına sahip olmayabilir: bazıları yeni LSA'ya sahip olurken, bazıları olmayabilir. Ancak güvenilir bir akış algoritması veritabanlarının bir *'yakınsama' (convergence)* periyodu sonrasında senkronize olacağını garanti eder. İyi bir akış algoritması LSA'ların çabucak dağıtımını yapar ve yakınsama zamanını küçük tutar.

Bir anahtar senkronize bir link durum veritabanına sahip olduğunda, link durum veri tabanını giriş olarak kullanarak ağ trafiği için en iyi yolları hesaplayabilir. Bu işlem, '*yönlendirme hesabı'* olarak adlandırılır. '*Dijkstra algoritması'* veya '*en kısa ilk yol (shortest path first - SPF)'* en uygun yolları hesaplamada yaygın olarak kullanılan metotlardır [43]. Bu sebepten dolayı link durum protokolleri çoğu kez, 'SPF-tabanlı protokoller' olarak adlandırılır.

4.6 Oğul Zekası Tabanlı Yönlendirme Algoritmaları

Oğul zekası (swarm intelligence) grup halinde yaşayan böceklerin sosyal davranışlarından esinlenen algoritmaların yeni sınıfını oluşturur. Zeki ve ölçeklenebilir sistemlerin tasarlanmasında yeni örnekler / yaklaşımlar, grup halinde yaşayan biyolojik canlıların doğasındaki tasarım prensiplerinin anlaşılması ve genişletilmesiyle meydana getirilebilir. Oğul zekası, bireysel olarak zeki olmamasına rağmen kolektif olarak zeki davranış gösteren ve sınırlı bireysel yeteneklere sahip varlıkların bir özelliğidir [45]. Karmaşık sistemler olarak dağıtık ağların tasarımında ve geliştirilmesinde önemli bir kavram *'görünen (emergent) zeka'* olmaktadır. Görünen zeka; grup üyelerinin basit yerel etkileşimleri sonucu ortaya çıkan karmaşık ve zeki davranış olarak tanımlanabilir. İletişim ağlarının yönetimi, artan ağ boyutu, hızla değişen topoloji, karmaşıklık, çeşitlilik, vb. nedenlerden dolayı tasarımı ve modellenmesi zor bir konudur. Oğul zekası tabanlı yeni bir algoritma sınıfı, ağların çeşitli problemlerine çözüm getirebilmektedir. Bu algoritmalar aynı anda birbirleriyle etkileşim yapan birçok görevlinin / varlığın kolektif uyumuna dayanır. Bu kısımda bu algoritmaların ilk örneklerinden bahsedilecektir.

Doğada biyolojik grup halinde yaşayan canlılar tarafından meydana getirilen oğul zekası, iletişim ağları gibi bir çok mühendislik sistemlerinde istenen sayısız özellikler barındırır. Oğul zekası çözümleri dağıtık sistemlere ve ağ problemlerine uygulandıkları zaman bir takım üstünlükler sunarlar. Bu üstünlükler;

- *Ölçeklenebilirlik:* Birbirinden bağımsız varlıklar, çoğalma ve göç yoluyla sistemin boyutuna uyarlanabilen tam olarak merkezi olmayan bir çözümü

ortaya koydukları için merkezi uygulamalara göre herhangi bir ölçeklenebilirlik problemine maruz kalmamaktadır.

- *Hata Toleransı:* Sistemi oluşturan bireyler / varlıklar bağımsız olarak hareket ettiklerinden dolayı (diğer varlıklara bağlı olmadan çalışabildikleri için), sistem sağlam ve hatalara karşı toleranslıdır. Merkezi yaklaşımlarda merkezi birim arıza yaptığında bütün sistem çökerken, bir grup varlığın ortadan kalkması / ölmesi sistemin tamamen çökmesine neden olmamakta, sadece sistemin performansında bir miktar düşüşe neden olmaktadır.

- *Uyarlanabilirlik:* Sistemi oluşturan varlıkların yaşam döngüsü (doğum, ölüm, göç, vb.) ve yetenekleri, değişen sistem şartlarına uyarlanabilen bir sistem meydana getirir.

- *Hız:* Sistemdeki değişikliklere yerel etkileşimler yoluyla hızlı cevap verilir.

- *Modülerlik:* Bireysel ve sadece kendinden sorumlu varlıklara dayalı problem çözüm yöntemi, yüksek seviyeli, modüler ve açık yapılı sistemlerin ortaya çıkmasına neden olur ve böylece sürekliliği ve güncelliği arttırır.

- *Otonomi:* Bu sistemlerde yönetim son derece dağıtık ve dinamik olduğu için herhangi bir yönetici birim gerekli değildir.

- *Paralellik*: Varlıkların işlemleri ve etkileşimleri doğal olarak paraleldir.

Oğul zekası, bu nedenlerden dolayı özellikle geniş ve yüksek derecede dinamik sistemler için oldukça uygundur.

İletişim ağlarında kontrol için kullanılabilen gezgin yazılım görevlileri (mobile software agents), ilk kez 1994 yılında Appleby ve Steward tarafından telefon ağları için yapılmış ve gerçek uygulamayla ilgili olarak açık ancak belirsiz duran problemleri ortaya çıkaran bir makale halinde yayınlanmıştır [14]. Daha sonra, 1996 yılında Schoonderwoerd, basit görevlilerin ağdaki düğümlerin yönlendirme

tablolarını güncellediği Appleby ve Steward'ın çalışmasının ilginç bir versiyonunu sunmuştur [5]. 1997 yılına gelindiğinde, karıncaların optimizasyon sistemleri üzerinde araştırmalarda bulunan Di Caro ve Dorigo, bu alanda en çok ilgi gören çalışmaları olan '*AntNet'* sistemini geliştirmişlerdir [20]. *AntNet* adaptif görevli tabanlı bir algoritmadır ve ağ üzerinde keşfedilen yollar üzerinde ileri ve geri hareket edebilen karınca türü görevlilerin etkileşimlerine dayanır. Aynı yıl, Bonabeau, karınca tabanlı görevli sistemini biraz daha geliştirerek 'akıllı' yönlendirme algoritması geliştirdi [19]. 1998 yılında, Heusse et al. dinamik programlamaya dayalı bir başka ilginç yönlendirme tekniğini sunmuştur [13]. Yine aynı yıl, White et al. tarafından geliştirilen algoritma karınca tabanlı yönlendirmenin ilginç örneklerinden biridir [46]. Daha yakına gelindikçe, Lipperts ve Kreller, yük dengeleme problemine farklı bir gezgin görevli yaklaşımı sundular [16]. Roth ve Wicker tarafından sunulan Termit algoritması [47], Kassabalidis tarafından sunulan Adaptif-SDR [48] biraz daha gelişmiş özellikler taşıyan oğul zekası algoritmalarıdır.

BÖLÜM 5. SWARMNET MODELLEME VE SİMÜLASYON ÇERÇEVESİ

5.1 Giriş

Bu bölümde, dağıtık bir ağ sistemini oluşturan bileşenler ayrık olaylı sistem tanımlama yaklaşımı (Discrete Event System Specification - DEVS) kullanılarak tanımlanacak ve tanımlanan bileşenlerin davranışları detaylandırılacaktır. Ağın kontrolü ve yönetimi işlemlerinde aktif rol oynayan bir takım küçük, hareket edebilen varlıklarının bulunduğu düğümler ve bu düğümleri birbirine bağlayan linklerden oluşan dağıtık sistemlerinin modellenmesi ve simülasyonu, Bölüm 3'te bahsedilen DEVS yaklaşımı kullanılarak gerçekleştirildi. Daha önceki bölümlerde

ifade edildiği üzere DEVS, bileşen davranışlarının tanımlanmasına olanak tanıyan bir sistem teorisidir [22] [57] [58].

Davranışları belirlenen sistemin modellenmesinden sonra, geliştirilen sistemde yönlendirme algoritmalarının test edilmesi, büyük ölçekli ağların analizi ve modellenmesi, vb. bir takım ileri ağ uygulamalarını gerçekleştirmek amacıyla sistemi oluşturan parçalar ve bileşenler JAVA dili kullanılarak kodlandı. DEVS birleşik model kavramı kullanılarak birleştirilen parçalar ve bileşenler birbirine bağlandıktan sonra, gerçekleştirilecek simülasyon deneyleri ve model davranışı üzerinde gözlem yapmak işlemlerinde Bölüm 2'de tanımlanan *'deneysel çerçeve'* aracı kullanıldı. Deneysel çerçeve kavramı, modellerin analizi için gözlemlenecek simülasyon şartlarının tanımlanmasında resmi bir yapı sağlar [22].

Java dilinde DEVS metodolojisi kullanılarak özellikle ekoloji kökenli yönlendirme algoritmalarını çalışmak üzere geliştirilen ortam, '*SwarmNet'* olarak adlandırıldı. Simülasyon deneyleri, DEVS modellerinin çalıştırılmasına olanak tanıyan DEVSJAVA modelleme ve simülasyon ortamını kullanılarak gerçekleştirildi. DEVSJAVA ortamı; kullanıcı dostu arabirimiyle kolay tasarım, modelleme, simülasyon ve eğitim işlemlerinde kullanılan etkili bir araçtır [25]. Bir sistemi DEVSJAVA ortamında uyarlamak; nesneye-yönelik modellemeyi, eş zamanlı paralel çalışan simülasyonları, etkileşen simülasyon nesneleri arasında uyumluluğu ve web tabanlı simülasyonları olanaklı kılmaktadır.

Aşağıdaki kısımlarda, *SwarmNet* ortamının modellenmesi ve simülasyonunun aşamalarını oluşturan modelleme sürecinin tanımlanmasından sonra ağı oluşturan bileşenler, bileşenleri oluşturan bir takım nesneler ve kavramlar detaylandırılacaktır. Sunulan teorik bilgi, DEVJAVA ortamından alınan ekran çıktılarıyla desteklenmekte ve bu bölümde tanımlanan bileşenlerin Java kaynak kodları ekte verilen CD içinde sunulmaktadır.

5.2 Java Programlama Dili

Java, taşınabilir uygulamalar ve appletler gibi bir takım kolaylıklarla yazılımcılara heyecan vermiştir [42]. Gerçekte, Java üç farklı taşınabilirlik sunar: kaynak kodu taşınabilirliği, CPU mimarisi taşınabilirliği ve OS/GUI taşınabilirliği. Bu üç tür taşınabilirlik bazı teknoloji paketlerinin bir sonucudur. Bunlar; Java programlama dili, Java sanal mekanizması (JVM) ve dil ile ilişkili sınıf kütüphaneleridir. Java programlama dili en alışık olduğumuz taşınabilirliği sağlar: kaynak kodu taşınabilirliği. Belirli bir Java programı, temel CPU, işletim sistemi veya Java derleyicisi dikkate alınmaksızın aynı sonuçları üretmelidir. JVM CPU mimarisi taşınabilirliğini sağlar. Java derleyicisi, JVM için nesne kodunu (J-code) üretir ve belirli bir CPU üzerinde JVM ilgili CPU için J-kodunu derler. OS/GUI taşınabilirliğini tesis etmek için, Java sanal OS ve GUI için bir takım paketler (awt, util ve lang) sağlar.

Java programlama dili nesneye yönelik olguyu benimser ve dağıtık uygulamaların geliştirilmesi için araçlar sağlar. Bu tezde ağ sistemleri Java teknolojileri kullanılarak modellendi ve test edildi. Karmaşık sistemler hesaplama olarak daha büyük kaynaklara ihtiyaç duyduğu için, geliştirilen bileşenlerin artan Java performansından faydalanacak olması bir avantajdır. Java çalışma anı performansının arttırmak aktif bir araştırma alanıdır.

5.3 Ağ Modelleme Süreci

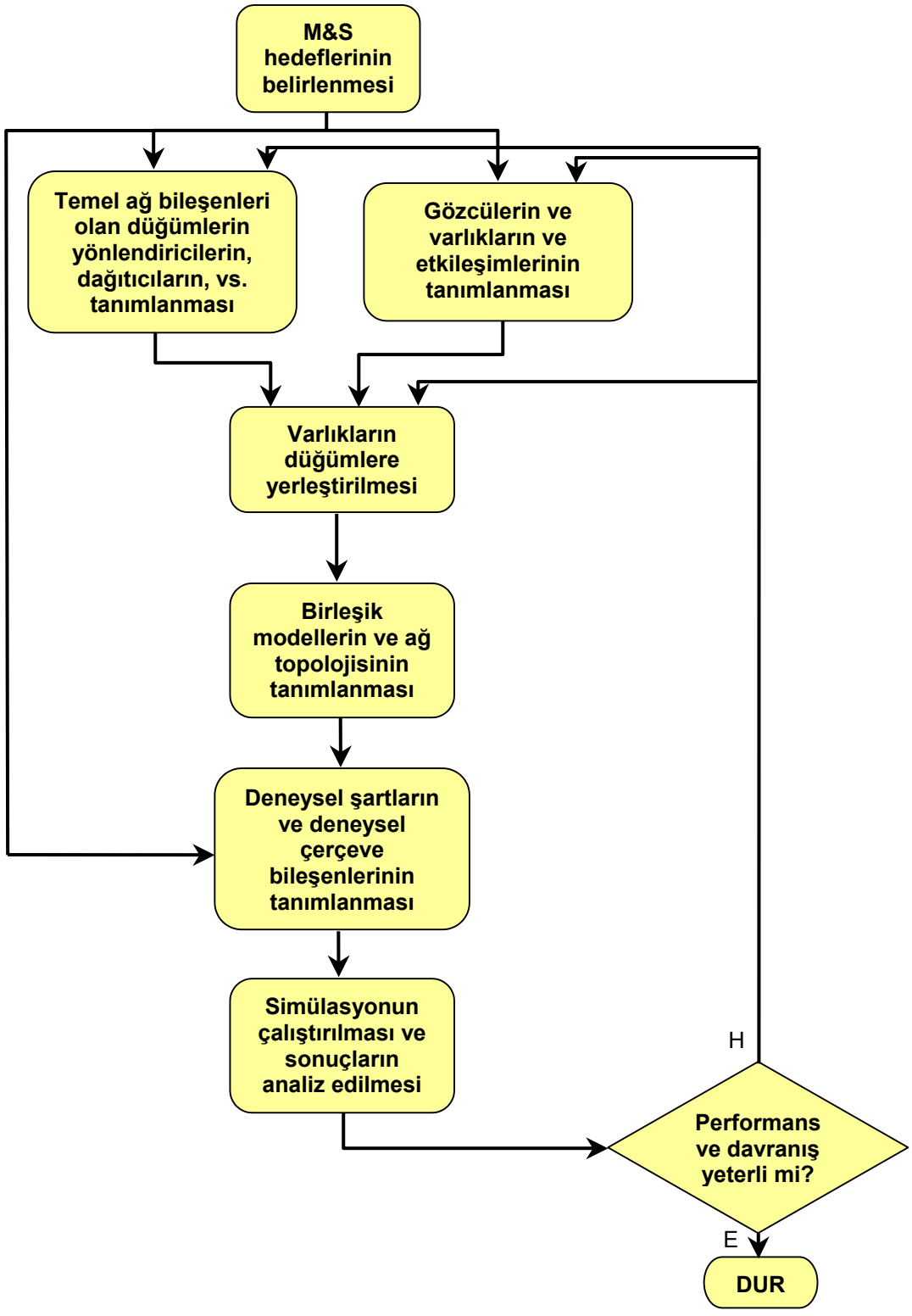
Dağıtık ağ sistemlerinin tasarım ve analizini sistematik bir şekilde gerçekleştirmek için bu sistemlerin modellenmesi ve simülasyon işlemleri bir takım aşamalara bölünür. Bu aşamalardan ilki modelleme ve simülasyon hedeflerinin belirlenmesidir (Şekil 5.1). Modelleme hedefleri, modelin sistemlerin tasarımında, yönetiminde ve kontrolünde tanımlanan rollerle ilişkilidir. Hedefler ifadesi, model tasarlama işleminin belirli problemler üzerine odaklanması vazifesini görür. Önceden hedefler bilinirse, bu hedeflere uygun deneysel çerçeveler geliştirilebilir.

İkinci aşamada simülasyon bileşenlerinin tanımlanması işlemi gerçekleştirilir. Bu aşamada simülasyonu oluşturan bütün bileşenler ve bu işlemlere ilişkin parametreler tanımlanır. Bu aşamada düğümlerin ve varlıkların ayrı ayrı ele alındığına dikkat edilmelidir.

Üçüncü aşamada, geliştirilen varlıklar ve düğümler birleştirilir. Bu bileşenler DEVSJAVA modelleme ve simülasyon ortamında birbiriyle bağlanarak değişik topolojiler ve ağ konfigürasyonları meydana getirilebilir.

Dördüncü aşamada, modelleme ve simülasyon hedeflerine ve geliştirilen modellere uygun bir deneysel çerçeve tanımlanır. Bu deneysel çerçeve modelin bir takım şartlar altında test edilmesine ve gözlem yapılmasına olanak tanır.

Son aşamada, simülasyon çalıştırılır ve sonuçlar gözlemlenir. Eğer sonuçlar kabul edilebilir bir aralıkta ise simülasyon işlemi sonlandırılır. Aksi halde, tekrar başa dönülür ve geliştirilen modellerin parametreleri ayarlanır. Böyle bu süreçte ileri geri hareket edilerek en uygun model geliştirilmiş olur.



Şekil 5.1 Dağıtık bir ağın modelleme ve simülasyon süreci

5.4 Ağ Modelleme Yaklaşımı ve Ağı Oluşturan Bileşenlerin Tasarımı

Dağıtık bir ağ sistemini modelleme amacına yönelik oluşturulan linkler, bu linkler aracılığıyla birbirleriyle haberleşen düğümler ve IP paketleri, 'temel ağ bileşenleri' olarak tanımlanır [61]. Tanımlanan düğümler ve linkler ile ayarlanabilir kapasiteleri kullanılarak çeşitli ağ konfigürasyonları ve topolojileri geliştirilebilir. Daha sonra bu bileşenlerin bir araya gelmesinden oluşan bir ağ modeli *'DEVS birleşik ağ modeli'* olarak adlandırılır. Burada kullanılan atomik ve birleşik modeller, Bölüm 3'te tanımlanan '*Paralel DEVS yaklaşımı'* kullanılarak tanımlanılır ve '*DEVSJAVA'* modelleme ve simülasyon ortamında tasarlanılır. Bu yöntemde, düğümler ve linklerde tanımlı dinamikler, ağ modelinin davranışını (çıkış zamanı gibi) belirlemek için kullanılabilir.

Bir ağ topolojisi, düğümler (bilgisayar, yönlendirici, vb.) ve linklerden oluşur. Düğümler, ağ üzerinde dolaşan varlıklar / gözcüler / gezgin görevliler için bir hesaplama altyapısı oluşturması yanında diğer düğümlerle mesajlaşma ve veri trafiği için bir takım arabirimlere ve araçlara sahiptir. Diğer düğümlere linkler kullanılarak bağlanan bir düğüm, ağın yapısına göre değişik modlarda çalışabilir (yönlendirici, dağıtıcı vb.). Dağıtıcı (hub) modunda çalışan bir düğüme bir link üzerinden gelen paketler, diğer linkler üzerinden dağıtılır. Düğümün yönlendirici (router) modunda çalışması durumunda, paket adres bilgisi bir yönlendirme kararı verebilmek için kullanılır ve bu bilgiye göre paket belirli bir link üzerinden ilgili adrese gönderilir. Ağı oluşturan diğer bileşenler olan linkler ise ağdaki anahtarlama elemanları arasında bir haberleşme yolu sağlar.

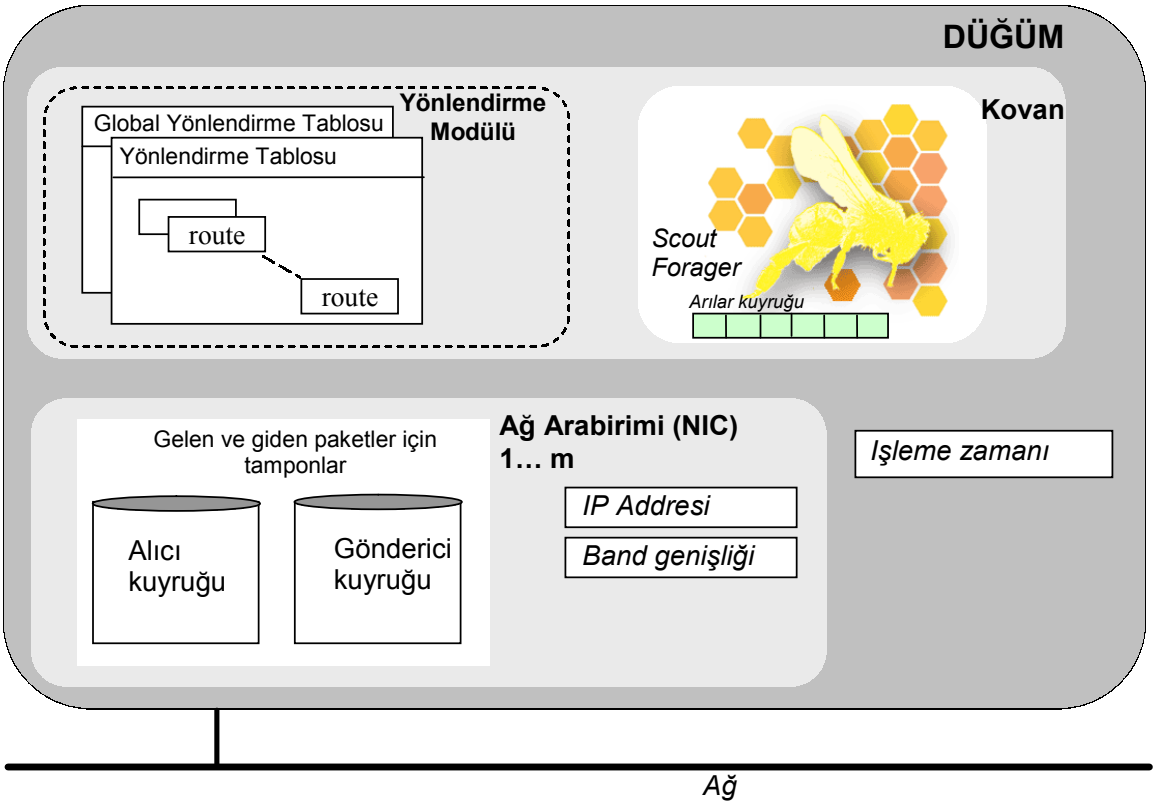
Yapılan çalışmada geliştirilen modellerle simülasyon deneyleri gerçekleştirmek için, bir modelin deney yapılacağı ve gözlemleneceği şartları tanımlayan '*DEVS deneysel çerçeve'* kavramından faydalanıldı. Bölüm 2'de detaylı olarak ifade edildiği gibi, tipik bir deneysel çerçeve bir üreteç ve dönüştürücüden oluşur. Ağ trafiğini üretmek ve link, düğüm, vb. ağ bileşenlerinin arızalanmaları ve hizmet verememeleri gibi ağın çalışması sırasında meydana gelebilecek olayları önceden programlamak ve zamanlamak maksadıyla bir *'olay üreteci'* kullanıldı. Bu işlemleri gerçekleştirmek için üreteç ağdaki ilgili bileşenlere DEVS mesajları gönderir.

5.4.1 Düğüm atomik modeli

Ağdaki her bir düğüm, aşağıda yapısı detaylı bir şekilde tanımlanacak olan paketleri işleyebilme ve paketleri uygun hedeflere yönlendirebilme yeteneğine sahip bir anahtarlama birimi olarak modellenmiştir. Modellenen düğümler iki veya daha fazla ağ bağlantısıyla (link) birbirine bağlanan DEVS atomik modelleridir. Düğümlerin davranış karakteristikleri, düğüm modu (dağıtıcı, yönlendirici, vb.), trafiğin işlenmesi için bant genişliği, işlem yapma hızı ve trafiği işleyebilecek kapasitede tampon boyutudur. Tanımlanan karakteristiklerle oynanarak değişik kapasiteye sahip ağ birimleri oluşturulabilir ve farklı ağ senaryoları geliştirilebilir.

Daha öncede ifade edildiği gibi bir düğüm farklı anahtarlama elemanlarına kolayca uygulanabilir. Bir yönlendirici ve dağıtıcı arasındaki en önemli fark yönlendirme işleminde karar mantığına sahip olunmasıdır. Yukarıda tanımlandığı şekilde, dağıtıcı modunda düğüm bütün sahip olduğu linklere trafiği yayarken, yönlendirici modunda bir son nokta (end to end) haberleşmesi için sadece ilgili linke trafik yönlendirilir. Yönlendirici modelleri belirli bir hedef noktası için gidiş linkini belirlemek maksadıyla bir yönlendirme karar mantığına ihtiyaç duyar. Bunu gerçekleştirmek için düğüm modeline yönlendirme tabloları yerleştirilebilir (Şekil 5.2).

Şekil 5.2, üç adet modülden oluşan bir düğüm mimarisini göstermektedir. Bu modüller yönlendirme modülü, ağ arabirimi (Nerwork Interface Card - NIC) ve kontrol amacıyla bir takım varlıklar olarak gözcüleri ağ üzerine yayan bir kovandır. Yönlendirme modülü, bir başka değişle IP İşleyici (IP Handler), yönlendirmenin yapıldığı birimdir. Her düğümde, paketler hedef düğümlerine yönlendirme modülü kullanılarak iletilir. Yönlendirme modülleri paket yönlendirme işlemini yönlendirme tablolarını ve yönlendirme algoritmalarını kullanarak gerçekleştirir. Daha sonraki bölümlerde de ifade edileceği gibi yönlendirme tabloları ağ içerisindeki bütün düğümler için girdilere sahiptir. Bu girdiler paketin belirli bir hedef için kullanacağı muhtemel yollardır. Yönlendirme modülü düğümün bulunduğu ağ için yerel bir yönlendirme tablosu ve diğer ağlarla haberleşmede kullanılan bir global yönlendirme tablosundan oluşur (Şekil 5.2). Bu yönlendirme tabloları ağın mevcut durumunu yansıtırlar ve uzaklık vektörleriyle (distance vectors) benzerlikler taşırlar.



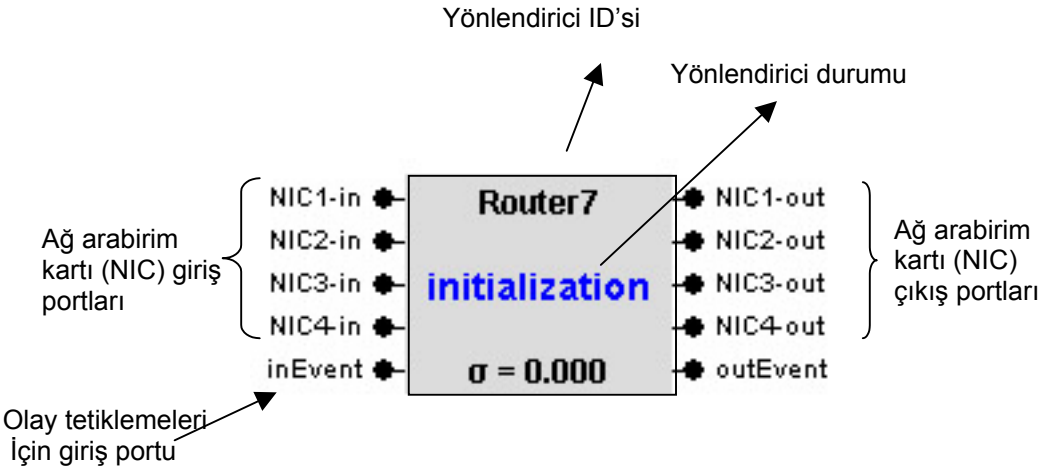
Şekil 5.2 Geliştirilen ağ düğümünün iç yapısı

Ağ arabirimi temel ağlar arası iletişim hizmetleri sağlarken, arabirim ağın topoloji (komşu düğümler) yönetiminden sorumludur ve gözcüler veya ağda dolaşan diğer varlıklar için altyapı sağlar. Ağ içinde her düğüm bir tanımlayıcı etikete / kimliğe sahiptir. Uzak bir düğümle haberleşebilmek için onun kimliği bilinmelidir. Ağ arayüzü veya haberleşme katmanı yeni bir düğümü / komşuyu keşfederek yönlendirme tablosuna ekleyebilir veya eğer erişilemez olduğunu fark ettiği düğümü / komşuyu yönlendirme veritabanından kaldırabilir. Gerçekleştirilen uygulamada, bu keşif ve kaldırma işlemleri gözcüler tarafından tespit edilip daha sonra arayüz tarafından doğrudan yerine getirilme şeklindedir. Kabul edilen düğüm gösteriminde, ağ arabirimi gelen ve giden paketler için sınırlı kapasitede iki adet kuyruğa sahiptir (Şekil 5.2).

Düğümü oluşturan son modül olarak bir kovan sistemi bir takım yapay görevlilerden ve gözcüler olarak adlandırılan varlıkları üretme işlevine sahip olacak şekilde tasarlanmıştır. Bu yapay varlıkların işlevi uygulamadan uygulamaya değişebilir. Örneğin, bir uzaklık vektörleri yönlendirme algoritması uygulamasında kovan,

düğümler arasında hareket eden ağ kontrol paketlerini, Link durumu algoritmasında LSA paketlerini oluşturabilir. Düğüme kovanı yerleştirmemizin asıl sebebi, sosyal böcek davranışının ve optimizasyon düzenlerinin esin kaynağı olduğu bir takım oğul zekası (swarm intelligence) kullanan yönlendirme algoritmalarının uygulanabilmesine, analizine, test edilmesine ve karşılaştırılmasına olanak tanımasıdır. Bölüm 6'da tanımlanacak olan oğul zekası yönlendirme yaklaşımında, kovan ağ kaynaklarını ayarlamak ve ağı izlemek için gözcüler ile birlikte diğer varlıkları kullanmaktadır.

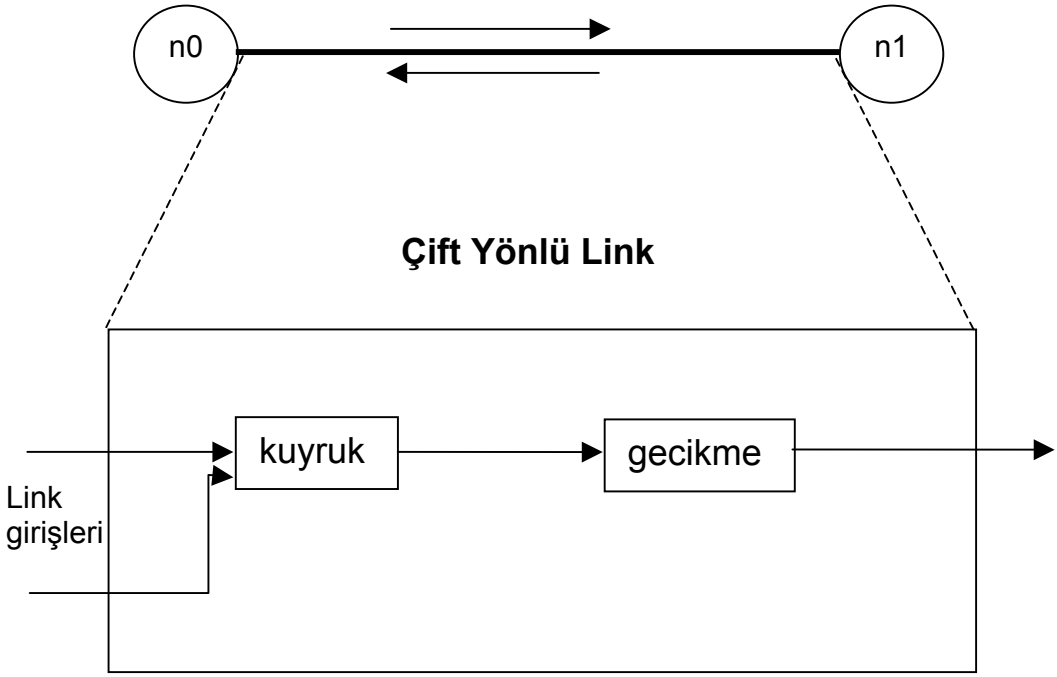
Yönlendirici modunda çalışan bir düğümün DEVSJAVA'da görsel olarak görünümü Şekil 5.3'de görülmektedir. Paralel DEVS modelleme yaklaşımı kullanılarak tanımlanan düğüm modelinin Java kaynak kodları ekteki CD'de verilmektedir.



Şekil 5.3 Dört adet arabirime sahip yönlendirici modunda çalışan düğüm atomik modelinin ekran çıktısı.

5.4.2 Link atomik modeli

Linkler ağın temel bileşenlerinden birisidir. Belirli bir düğüm genellikle coğrafik olarak komşusu olan düğümlere linkler aracılığıyla bağlanır. Bütün linkler haberleşme kanallarıdır ve dolayısıyla bant genişliği (bits/s) ve yayılım gecikmesi (s) ile karakterize edilen bit transfer hatları olarak görülebilir. Bu değerler değiştirilerek farklı tipte link modelleri oluşturulabilir.

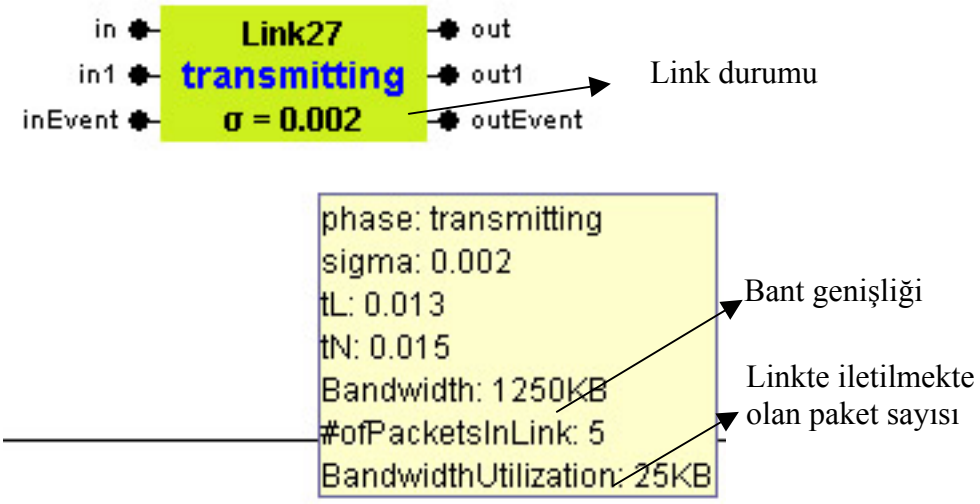


Şekil 5.4 Çift yönlü (dubleks) link modeli

Linkler her bir düğümün arabirimleriyle birleştirilerek farklı topolojide ağlar meydana getirilebilir. Her link sınırlı kapasiteye sahip bir tampona sahiptir. Gelen paketler tamponda bekler ve ilk gelen ilk çıkar (FIFO) stratejisi kullanılarak bir sonraki düğüme iletilir. Linkler çift yönlü olarak tasarlanmaları nedeni ile aynı anda her iki yönde trafik akışı mümkündür. Linkler toplam kapasitelerine kadar belirli bir bant genişliğinde trafiği taşıyabilir. Her link atomik modeli, iki adet düğümü çift yönlü bir şekilde bağlamak için giriş ve çıkış portlarına sahiptir (Şekil 5.4).

Her link her simülasyon adımında ilişkili parametrelerine göre kuyrukta bekleyen paketleri işler. Kullanılan parametreler, bant genişliği ve gecikmedir. Bant genişliği bir birim zamanda bir linkin işleyebileceği / iletebileceği veri miktarı, gecikme ise bir paketin link üzerinden geçmesi için gereken minimum süredir. Şekil 5.5'te bir link atomik modelinin ekran çıktısı ve parametre ekranı görülmektedir.

Paralel DEVS modelleme ve simülasyon yaklaşımı kullanılarak tanımlanan bir link modelinin Java kaynak kodu ise ekteki CD içinde verilmektedir.



Şekil 5.5 Çift yönlü bir link ekran çıktısı ve parametre ekranı.

5.4.3 Ağ paketleri

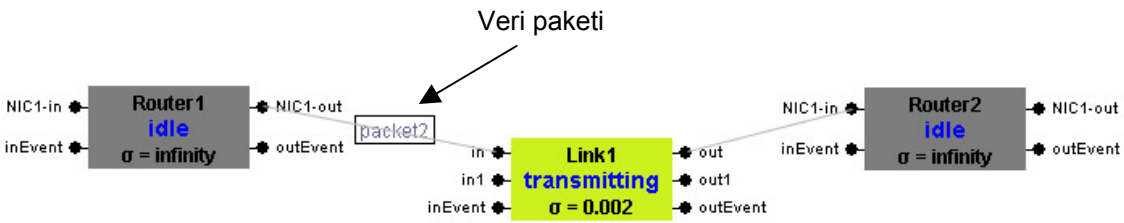
Ağı oluşturan bileşenler arasında DEVS mesajları şeklinde karşılıklı değişilen paketler, *veri* ve *kontrol* paketleri olarak ikiye ayrılabilir. Veri paketleri, bilgi taşıyan basit IP paketleri olarak düşünülebilir. Kontrol paketleri ise, ağ yönetim protokollerinin temelini oluşturan ağ izleme ve kontrol işlemini gerçekleştirme işlevini yerine getirme vazifesi görürler. Bütün uygulamalarda veri ve kontrol paketleri temel olarak aynı yapıya sahip olmalarına ve aynı bilgiyi taşımalarına karşılık, farklılıkları tanımlanan davranışlarındadır.

Kontrol paketleri veri paketlerinden farklı olarak uygulamalar tarafından oluşturulmazlar. Düğüm içerisindeki bir platform tarafından oluşturulup, ağa belirli bir görevi yerine getirmek için yollanırlar. Kontrol paketleri ağ yönlendirme veritabanları üzerinde çalışabilir ve gerektiğinde bu veritabanlarını güncelleyebilirler. Yapılan çalışmada tanımlanan kontrol paketleri, esnek yapıları sayesinde uygulamadan uygulamaya göre farklı şekil alabilir veya farklı biçime sokulabilirler. Bu özellik Java programlama dilinin esnekliği ve gücü sayesindedir. Örneğin; klasik bir algoritma olan OSPF algoritması uygulamasında kontrol paketleri LSA paketleri olurken, sosyal böcek optimizasyon algoritması örneklerinde ağı kateden bir karıncayı veya bir gözcü arıyı temsil edebilir. Bu esnekliği sağlamak için paket, özel uygulamaların gereksinimlerini karşılayacak

şekilde otonom hareket etme ve öğrenme kabiliyeti vb. yeteneklere sahip olacak şekilde modellenmiştir. Kontrol paketleri yönlendirme işlevini gerçekleştirecek bir düğümün ağı izlemesini ve trafiği ölçmesini sağlar.

Paketler genel olarak ağ bileşenleri arasında karşılıklı değişilen nesnelerdir (Şekil 5.6). Bir paket bir düğüm atomik modeli tarafından üretilebilir. Üretilen paket, giden paket kuyruğuna yerleştirilir ve ilgili ağ arabirimine bağlı bir link tarafından belirli bir gecikmeye tabi tutularak diğer düğümlere iletilir. Paket düğümlere vardığında, ilk olarak gelen paket kuyruğuna yerleştirilir. Veri paketleri kuyrukta varış sırasına göre yerleştirilip *FIFO* mantığına göre işlenirken, kontrol paketleri yüksek önceliğe sahip oldukları için kuyruğun en üstüne yerleştirilirler. Kontrol paketlerinin yüksek önceliğe sahip olmasının sebebi, kontrol paketlerini olabildiğince hızlı bir şekilde işleyip hedeflerine ulaşmalarını ve ağ kontrol işlevinin sağlıklı / verimli olarak yerine getirilmesini sağlamaktır.

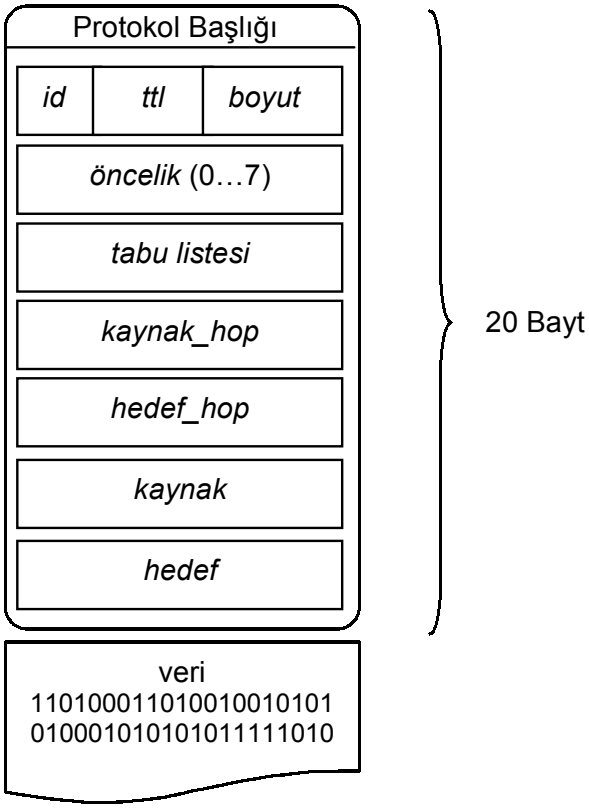
Daha öncede belirtildiği üzere, ağı oluşturan linkler ve düğümler sınırlı kapasiteye sahiptir. Örneğin, bir düğümün kuyruk kapasitesi yeterli boş alana sahip olmadığında, o düğüme gelen paketler atılır ve böylece veri kaybı oluşur. Bir paketin atılmasına ikinci bir neden paketin yaşama zamanıdır (time-to-live). Bu zamanı aşan bir paket otomatik olarak ağdan atılır.



Şekil 5.6 Bir veri paketinin DEVSJAVA ortamında bileşenler arasında hareketi.

Şekil 5.7, ağdaki bütün tipteki paketler için genel bir paket modelini göstermektedir. Kaynak kodu ekteki CD'de verilen paket modeli bir takım değişken alanlara sahiptir: *id*, *ttl*, *paket boyutu*, *öncelik*, *tabu listesi, kaynak adresi*, *hedef adresi*, *kaynak_hop adresi, hedef_hop adresi ve veriler.* Veri hariç diğer bütün değişkenler *Protokol Başlığını (Protokol Header)* oluşturur ve bizim uygulamamızda bütün

paket türleri için 20 Bayt'lık standart bir boyuta sahiptir. Paketin toplam boyutu paket verisinin boyutuna göre değişebilir ve paket boyutu değişkeninde saklanır.



Şekil 5.7 Genel bir IP Paket modeli.

Paketin işlenme sırası *0* ve *7* arasında değerler alabilen bir *öncelik (precedence)* değişkeniyle belirlenir. Bu değer veri paketleri için en düşük değer olan sıfırdır. Kontrol paketleri, *0* hariç diğer değerleri alabilir. Öncelik değeri, bir paketin kuyruklarda tam olarak yerleşme sırasını belirler.

Paket *'id'* değeri bir paketin kimliğini saptamaya yarar ve mesaj tanımlama alanı olan bu alan her bir paketin ağ içerisinde tanımlanabilmesine olanak tanır.

'*ttl'* değeri, bir paketin çok uzun bir süre ağı işgal etmesini engellemek için kullanılır. Uygulamalarda ttl değeri olarak hop sayısı kullanıldı.

'Tabu listesi' bir paketin iletim yolu boyunca ziyaret ettiği düğümlerin adreslerini tutar. Dolayısıyla özellikle ağ izleme ve kontrol işlemi gelişmiş metotlar

kullanılarak yapıldığında bu liste önemlidir. Ayrıca tabu listesi kontrol paketlerinin ağda sonsuz döngüye girmelerini engeller.

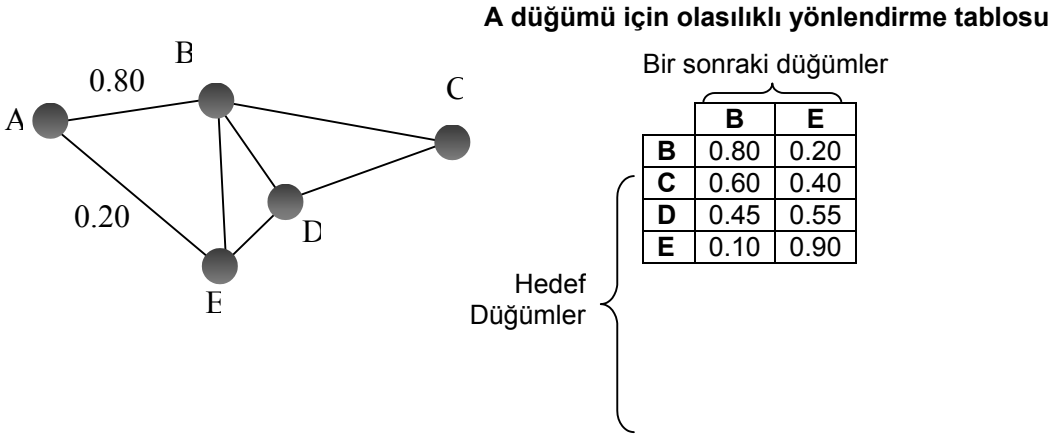
Kaynak ve *hedef adres* değişken alanları, paketin orijinal ve final düğümünün IP adreslerini içerir.

Veri alanı, paketin veri nesnesini taşır.

Geliştirilen *SwarmNet* ortamında, varış onaylama (arrival acknowledgement) ve hata bildirim gibi kaynağa yollanan bir takım paketler kapsam dışında tutulmuştur. Sadece basit bir kontrol akışı modellenmesi yapılmasının sebebi, bir ağın bütün ayrıntılarına odaklanmaktan ziyade yönlendirme algoritmalarının davranışını çalışabileceğimiz minimum bir ağ modelinin oluşturulmasına odaklanmamızdır.

5.4.4 Yönlendirme tabloları

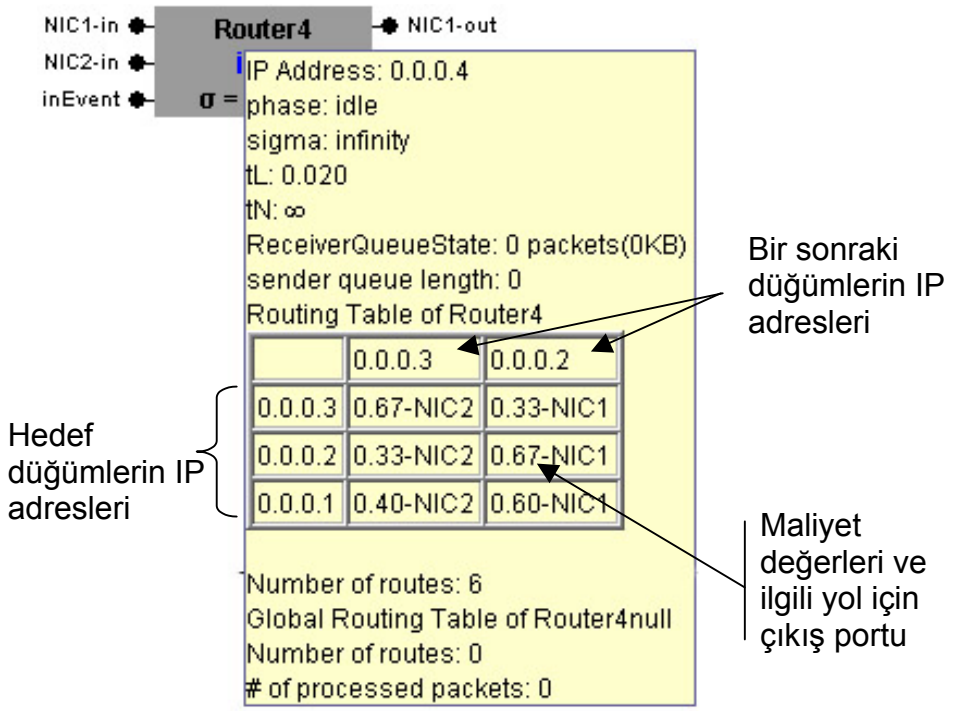
Her düğüm, içinde ağdaki muhtemel her bir hedef düğümün temsil edildiği bir yönlendirme tablosuna sahiptir. Bu yönlendirme tablosu komşu düğümlerin sütunları, kendisi hariç hedef düğümlerin ise satırları temsil ettiği bir matris olarak düşünülebilir (yani satırlar hedeflere, sütunlarda komşulara karşılık düşer) (Şekil 5.8). Yönlendirme tablosunun boyutu ilgili düğümün komşuları ve düğümün kendisi hariç ağdaki hedef düğümlerin çarpımı kadar olsa da, bizim uygulamamızda yönlendirme tablosunun boyutu düğümün keşfedebildiği düğüm ve komşu sayısı ile ilişkilidir. Her bir düğümü bir yönlendirme tablosuyla donatarak veri paketleri sistematik olarak ağ boyunca yönlendirilebilir. Bir düğüm bir paketi belirli bir hedefe doğru göndermeye ihtiyaç duyduğunda, hangi gidiş linkinin (veya DEVS atomik model çıkış portu) kullanılacağı kararı yönlendirme tablosunda belirtilen bilgiye dayanılarak verilir.



Şekil 5.8 Bir yönlendirme tablosu örneği.

Yönlendirme tabloları, yönlendirme algoritmalarıyla yakından ilişkilidir ve bir yönlendirme tablosunun oluşturulması ve güncellenmesi yönlendirme algoritmasının görevidir. Yönlendirme tablolarının oluşturulma, güncelleme ve yakınsama (convergence) hızı ağ trafiğinin dengeli olmasını (ağda boşta ve aşırı yüklü bir birimin olmaması) doğrudan etkilediğinden, yönlendirme algoritmalarının kalitesini ölçme aracı olarak kullanılabilir. Yönlendirme algoritmasına uygun olarak, bir yönlendirme tablosu statik algoritmalarda ağ kurulum aşamasında (ağda yük yokken), dinamik algoritmalarda ise ağın yük durumuna göre dinamik bir şekilde uyarlanarak kurulabilir. Dinamik yönlendirme algoritmalarında (oğul zekası tabanlı algoritmaları da kapsar), yönlendirme algoritmasına simülasyon çalışması sırasında ağ trafiği durumuna veya daha doğru ifadeyle ağın genel durumuna bağlı olarak yeni girişler eklenebilir ve mevcutlar kaldırılabilir.

Oğul zekası tabanlı yönlendirme algoritmaları genelde *olasılıklı (probabilistic)* yönlendirme yaparlar. Bu algoritmalarda kullanılan yönlendirme tabloları *0* ve *1* arasındaki olasılık değerlerini kullanırlar. Örnek bir yönlendirme tablosu Şekil 5.8'de gösterilmektedir. Bu tür algoritmalarda, bir hedef düğüm için bir linkin seçilme olasılığını veren yönlendirme tablosu değerleri ile orantılı olarak, belirli bir rasgele seçim düzeneği takip edilerek yollar seçilir. Belirleyici yönlendirmeyle karşılaştırıldığında bu yöntem, ağdaki veri akışı alternatif yollara bölündüğü için yük dengelemeyi sağlamayı mümkün kılar. *SwarmNet* ortamında bir düğümün olasılıklı yönlendirme tablosu Şekil 5.9'da görülmektedir.



Şekil 5.9 Bir yönlendirme tablosunun DEVSJAVA altında ekran görünümü.

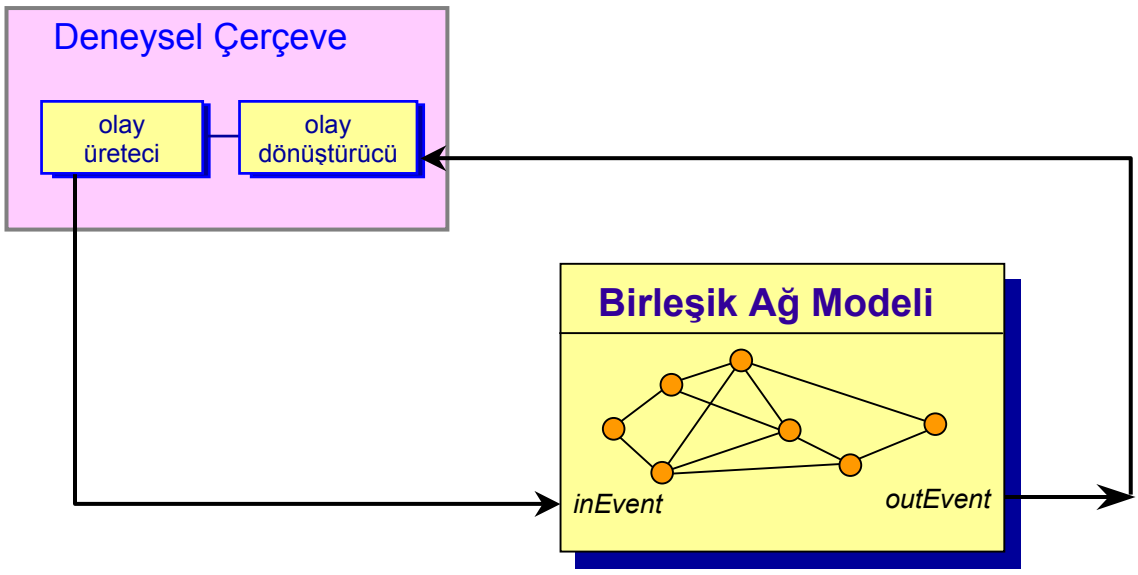
SwarmNet modelleme ve simülasyon ortamının Java dilinde gerçeklenmesinde, bir *RoutingTable* nesnesi *Route* nesnelerinden oluşur. *RoutingTable* nesnesi *IPHandler* sınıfının bir durum değişkenidir. *IPHandler* ise *Node* sınıfının bir durum değişkenidir. *RoutingTable* sınıfının içinde yollar Java kütüphanesinin *Vector* sınıfı olarak tanımlanmıştır. *RoutingTable* nesnesinin kaynak kodu ekteki CD'de verilmiştir.

5.5 SwarmNet Deneysel Çerçevesi ve Ağ Trafik Modeli

Yukarıdaki bölümlerde bir ağı oluşturan bileşenler tanımlandıktan sonra, bu ağın çalışması için uygun trafik üreteç modüllerinin tasarlanması gerekmektedir. Bölüm 2'de ifade edildiği gibi DEVS içerisindeki deneysel çerçeve kavramı modelleyicinin sistemi araştırmak istediği deneysel trafik şartlarını ve modeli çalıştırmak istediği senaryoyu tanımlamaya yarar.

Bir ağdaki yönlendirme algoritmaları, trafiği kaynaklardan hedeflere doğru ağ performans kriterlerini maksimize ederek yöneltme hedefine sahiptirler. Dolayısıyla yönlendirme algoritmaları incelenirken bu performans kriterleri göz önünde bulundurulmalıdır. Genellikle bir ağın performansı ölçülürken kriterler olarak '*ağ çıkışı' (throughput)*, ağ üzerindeki '*ortalama gecikme'* ve ağda kuyruklarda bekleyen '*paketlerin sayısı'* alınır [13]. Bir ağın çıkışı; bir birim zamanda ağdan geçip giden paketlerin sayısıdır veya bilgi miktarıdır. Belirli bir zaman aralığında varan bütün paketler tarafından maruz kalınan ortalama gecikme, ağın genel durumunun bir diğer yönünü verir. Linklerin ve düğümlerin kuyruklarında bekleyen paketlerin toplamı olan bekleyen paketlerin sayısı, ağ trafiğinin seviyesini belirlemeye yarar ve tıkanıklıkları gösterir.

Yapılan çalışmada, ağın performans kriterlerini ölçmek için DEVS deneysel çerçeve bileşenleri *SwarmNet* çerçevesine göre tanımlandı. Daha öncede belirtildiği gibi, deneysel çerçeve, en genel biçiminde sisteme muhtemel girişleri tanımlayan bir üreteçten, çıkış işlemini tarif eden bir dönüştürücüden (örneğin; çıkışla bütünleştirilen performans ölçümlerini hesaplama gibi) ve sistemin eşleştiği durumları (mantıksal ifadeleri) tanımlayan bir onaylayıcıdan oluşur. Yapılan çalışmada kullanılan deneysel çerçeve uyarlaması, Bölüm 2'de detayları verilen bir *olay üretecinden (event generator)* ve *olay dönüştürücüden (event transducer)* oluşur (Şekil 5.10).

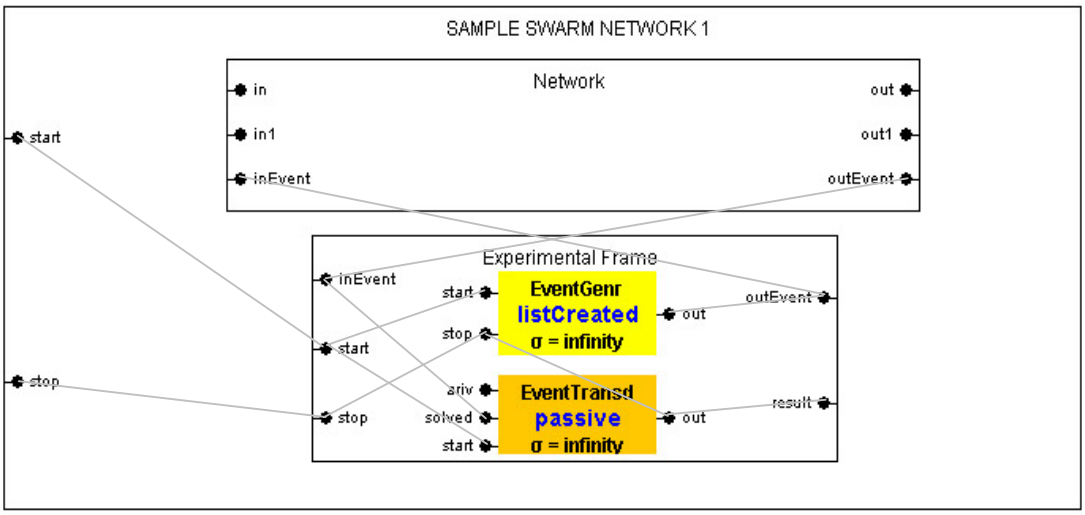


Şekil 5.10 Deneysel çerçeve.

Uygulamalar tarafından ağda iletilen veri paketlerini üreten olay üreteci, bir trafik kaynağı modelidir. Olay üreteci, belirli zaman aralıklarında periyodik olarak kaynağı ve hedefi rasgele seçilmiş veri paketleri üretebilir. Üretilecek paketlerin sayısı, boyutu ve üretilen paketler arasındaki zaman aralıkları seçilen olasılık modeli tarafından belirlenir. Çalışmada yapılan deneylerde rasgele seçimli bir trafik şekli kullanıldı. Üreteç, ağda oluşabilecek muhtemel olayları programlayabilir (örneğin link ve düğüm düşmesi gibi) ve algoritmaları dinamik bir ortamda test etmek için bileşen aksaklıkları ve ani parametre değişimleri gibi bir takım olayları üretebilir.

Her bir simülasyon çalışmasının sonuçlarını değerlendirmek için olay dönüştürücü kullanılır. Olay dönüştürücü ağ çıkışlarını gözler ve analiz eder. Gözlenen ve hesaplanan çıkışlar anlamlı sonuçlara dönüştürülür. Daha öncede ifade ettiğimiz gibi, ağ çıkışı, ortalama gecikme gibi bir takım kriterler dönüştürücü tarafından gözlemlenir, hesaplanır ve bir Excel dosyasında (comma seperated values - CSV) saklanır. Bu Excel dosyası daha sonra uygun grafikleri üretmek için kullanılabilir.

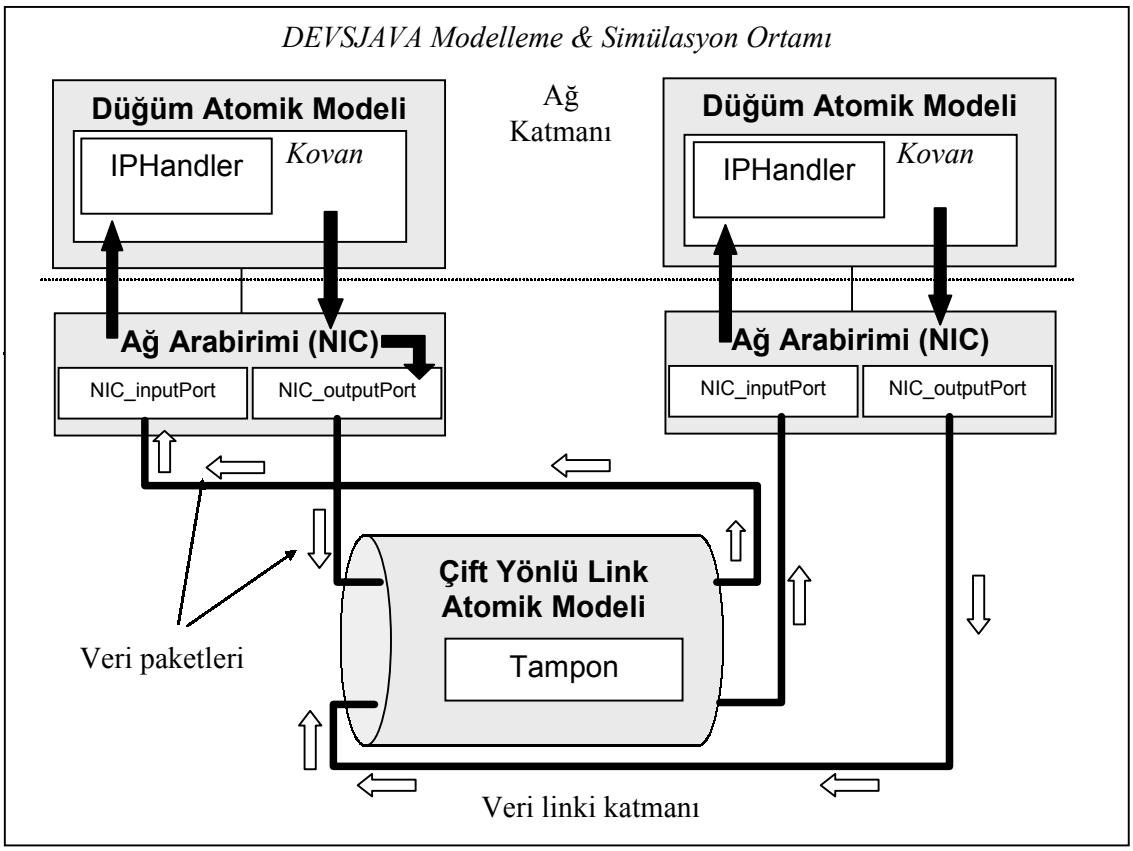
Bir olay üreteci ve olay dönüştürücüden oluşan deneysel çerçevenin DEVSJAVA ortamında ekran çıktısı ve bir ağ ile bağlantısı Şekil 5.11'de gösterilmektedir. Bir deneysel çerçeveyi oluşturan bileşenlerin Java kaynak kodu ekteki CD'de verilmiştir.



Şekil 5.11 Bir ağın deneysel çerçeveyle bağlantısı.

5.6 Birleşik (Coupled) Simülasyon Modelleri

Geliştirilen ve ayrıntıları yukarıda ifade edilen temel modeller (link, düğüm, vb.) DEVS birleşik model yaklaşımı kullanılarak, bir ağ oluşturacak şekilde birbirine bağlanarak değişik topolojilerde ve dinamik yapıda ağlar oluşturulabilir (Şekil 5.12). Ağı oluşturan bileşenlerin parametreleriyle oynanarak ağ değişik senaryolar altında çalıştırılıp test edilebilir. Daha öncede ifade edildiği gibi, böyle bir model geliştirmemizdeki temel neden yönlendirme algoritmalarını araştırmak ve test etmek olmasında rağmen, bileşenlerin değişken yapıları sayesinde *SwarmNet* ortamı, alternatif ağ konfigürasyonlarının ve topolojilerinin araştırılmasına ve değerlendirilmesi olanak tanımaktadır. Düğüm atomik modeli ve link atomik modeli DEVJAVA ortamı içerisinde bir ağ modeli oluşturacak şekilde birbirine bağlanarak değişik topolojiler oluşturulabilir (Şekil 5.13).

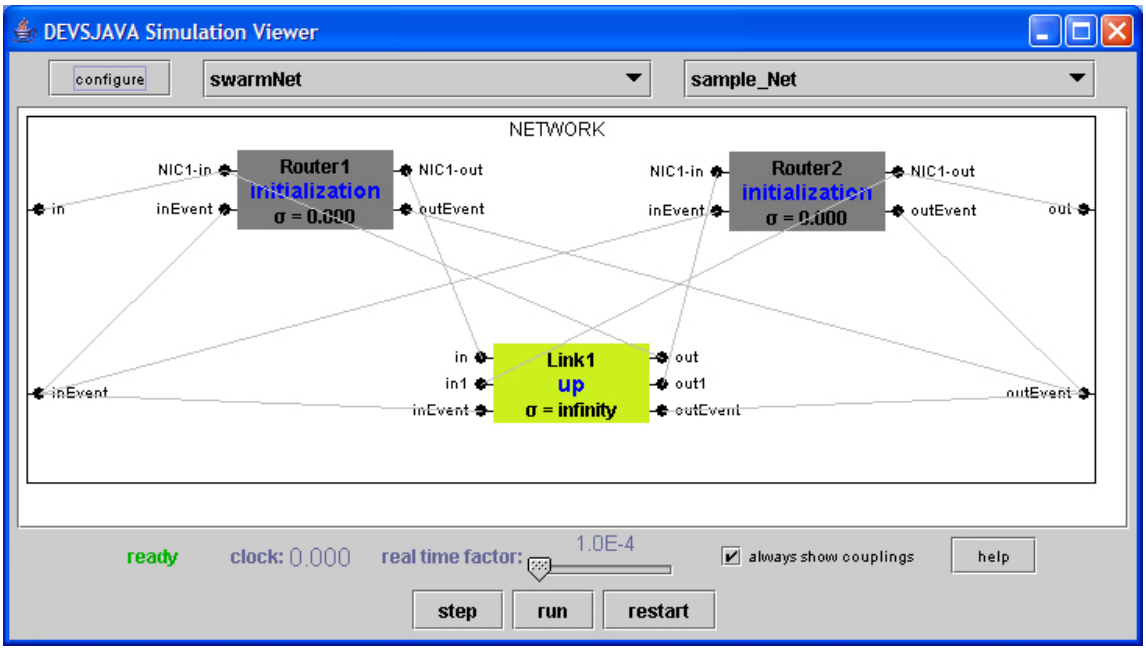


Şekil 5.12 Düğümlerin bir link aracılığıyla birbirine bağlanması

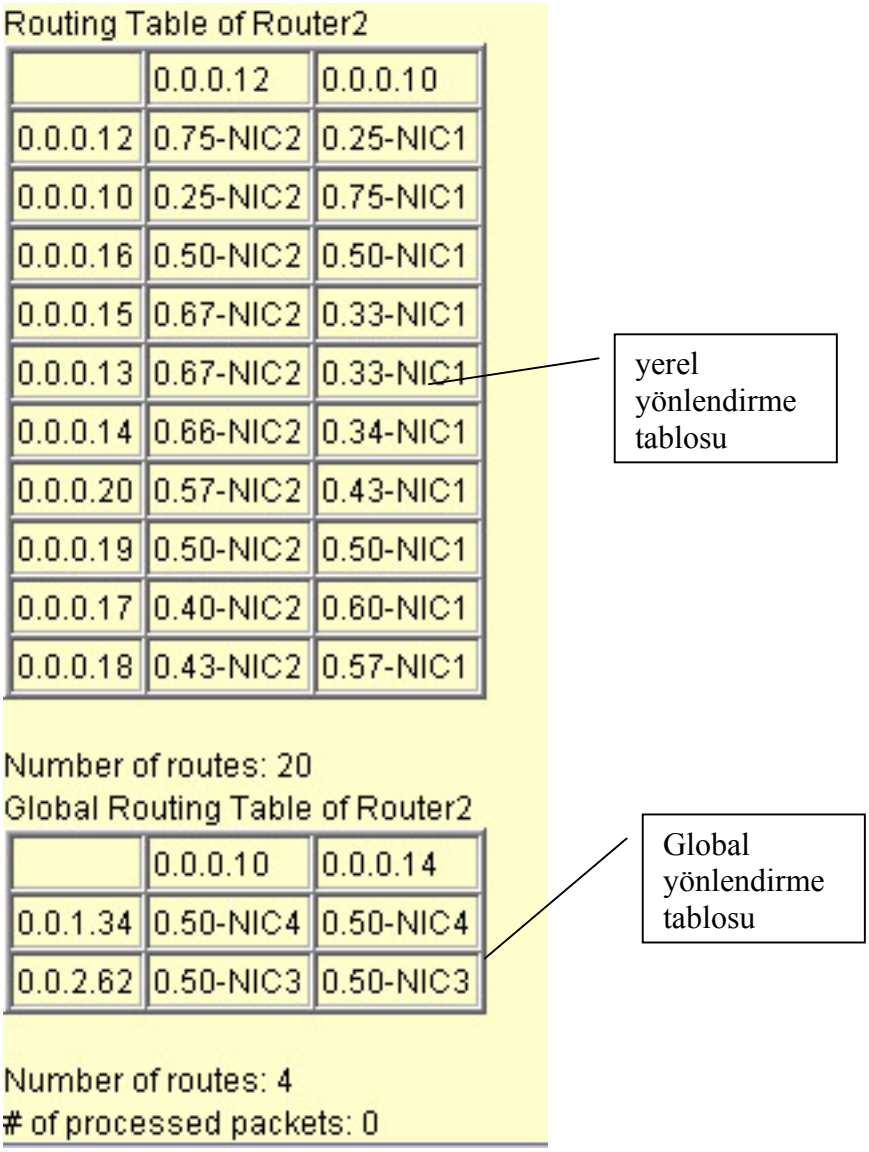
Bölüm 2 ve 3'te bahsedildiği gibi DEVS'in birleşim altında kapalılığı, birleşik modellerin daha büyük birleşik modeller altında atomik modeller olarak kullanılmalarına olanak tanımaktaydı. Bu çalışmada DEVS birleşim altında kapalılık özelliğini büyük ölçekli modelleri tasarlamak amacıyla kullanıldı. Dolayısıyla *SwarmNet* ortamı büyük ölçekli ağların araştırılmasına olanak tanır. Çalışma sırasında yapılan deneylerde yaklaşık 3500 düğüme sahip ağlar oluşturabildi. Daha büyük değerlere çıkılamamasının sebebi simülasyonlarda kullanılan bilgisayarlarının donanımının sınırlamalarıydı.

Büyük ölçekli ağlarla çalışırken karşılaştığımız en büyük problem ağdaki bileşenlerin sayısı çok fazla olduğundan dolayı oluşan yönlendirme veritabanının boyutunun bir düğümün belleğinin kapasitesini aşmasıydı. Bu problemi ortadan kaldırmak için ikinci bir yönlendirme tablosu tanımlandı.

Gerçekleştirilen sistemlerdeki her bir düğüm yerel yönlendirme tablosunun yanında diğer ağ kesimleri için bir global yönlendirme tablosuna sahiptir (Şekil 5.14). Aşağıdaki bölümde büyük ağların ölçeklenmesi sunulmuştur.



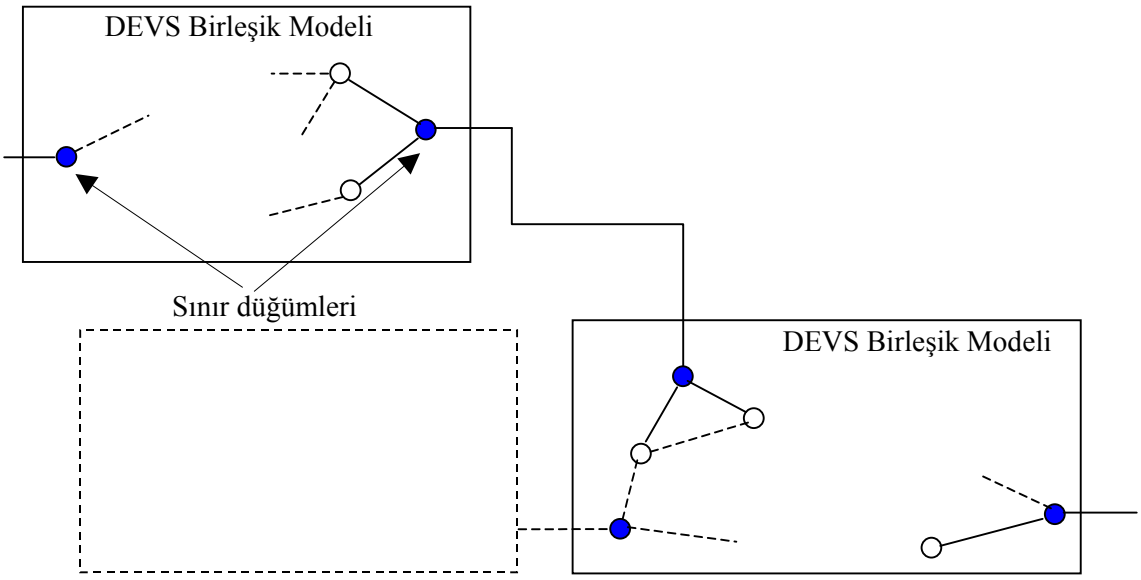
Şekil 5.13 Bir ağ modeli sentezi.



Şekil 5.14 Bir düğüm için global yönlendirme tablosu.

5.7 Birleşik Ağların Ölçeklenmesi

Büyük ölçekli ağların modellenmesinde karşılaşılan en büyük problem ölçeklenebilirliktir. Doğru bir ölçeklenebilirlik için bir kümeleme sistemi uygulanabilir (Şekil 5.15). Kümeleme bir alt ağı daha üst seviye bir ağ içerisinde bir düğüme indirgeyerek yönetilebilir ağ boyutları sağlar. Bu durumda, tüm düğümlerin toplamında bir ağ hiyerarşisi meydana gelir. Kümelere ayrılmış bir ağda düğümlerin adreslenmesi çoğu kez ağların hiyerarşisini yansıtır.

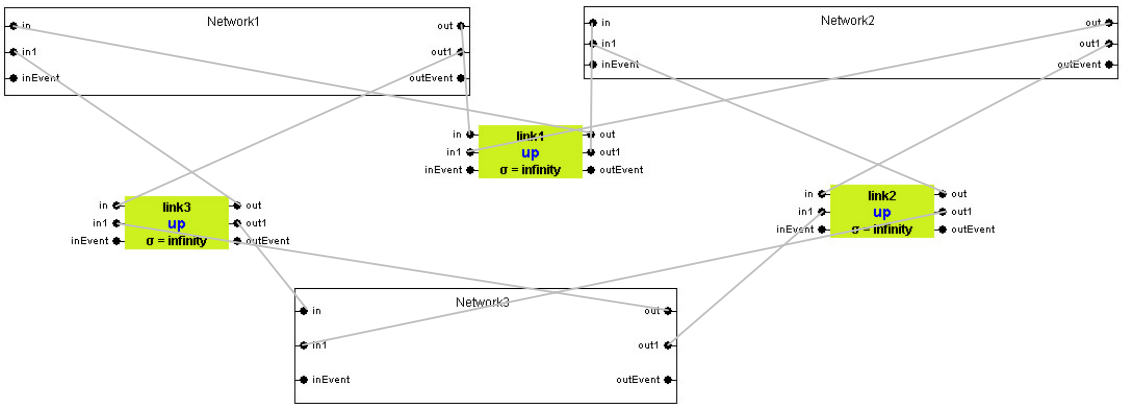


Şekil 5.15 Kümeleme yönteminin DEVS yaklaşımına uygulanması.

DEVS birleşim altında kapalılık özelliği birleşik bir modeli atomik bir model olarak ele aldığı için, DEVS birleşik modeli kümeleme yöntemiyle benzerlikler taşır. DEVS modelleme yönteminin hiyerarşik ve modüler yapısı, kümeleme yönteminin uygulanmasını kolaylaştırır. Yapılan uygulamada kümeleme düğümlerin adresleme seviyesinde halledilir ve her bir birleşik model bir takım '*sınır düğümlere'* sahiptir (Şekil 5.15). Bu sınır düğümleri diğer birleşik modellere bağlanmak için kullanılır. Sınır düğümleri içinde küme isimlerinin bulunduğu ilave bir yönlendirme tablosuna sahiptir. Bu yöntem ağ düğümleri içinde kayıtlı bulunan yönlendirme veritabanının boyutunu önemli ölçüde düşürür.

Benzer ölçeklenebilirlik problemleri yönlendirme algoritmaları içinde geçerlidir. Ağdaki her bir düğüm diğer bütün düğümlere bir gözcü veya uygulamadan uygulamaya değişen bir varlık göndermek zorunda olduğu için, oğul zekası kullanan yönlendirme algoritmaları ölçeklenebilirlik problemlerine sahiptirler. Eğer bir ağdaki düğümlerin sayısı *N* ise gönderilmesi gereken görevli varlıkların toplam sayısı *N x (N - 1)* olur. Büyük ağlar için, bu varlıklar tarafından üretilen trafiğin hacmi tehlikeli boyutlara varabilir ve uzak hedefler için varlıkların kaybolmalarının olasılığı yüksek olur. Bundan başka, bu varlıkların uzun seyahat süreleri, taşıdıkları bilginin geçersiz olmasına neden olabilir. Bu sorunları çözmede kullanılan pratik bir çözüm yolu, yukarıda açıklanan düğümleri kolonilere bölen kümeleme işlemidir.

Şekil 5.16'da kümeleme yöntemi kullanılarak birbirine bağlanmış üç adet ağ görülmektedir.



Şekil 5.16 Kümeleme yöntemine göre bağlanmış 3 adet ağ.

BÖLÜM 6. SWARMNET MODELLEME VE SİMÜLASYON ORTAMINDA YÖNLENDİRME ALGORİTMALARININ UYGULANMASI

6.1 Giriş

Yapılan çalışmada, geliştirilen '*SwarmNet'* modelleme ve simülasyon ortamının çalışmasını ve yönlendirme algoritmalarının bu ortamda uygulanabilirliğini göstermek amacıyla örnek çalışmalar gerçekleştirilmiştir. Yapılan çalışmanın ilk aşamasında; en yaygın kullanılan klasik yönlendirme algoritmalarından biri olan *'yönlendirme bilgi protokolü'* (Routing Information Protocol - *RIP*) algoritmasının '*SwarmNet'* içinde modellenmesi ve simülasyonu gerçekleştirildi. Uzaklık vektörü sınıfı yönlendirme algoritmalarının en yaygın kullanılanlarından biri olan *RIP* algoritması uygulanırken, '*SwarmNet'* ortamının klasik algoritmaları modelleme yeteneğinde olduğu gösterilmeye çalışıldı. Yapılan çalışmanın ikinci aşamasında; yüksek seviyeli bir dinamizme sahip olan bal arılarının nektar sahalarını ararken

göstermiş oldukları davranışı esin kaynağı olarak alan bir oğul zekası yönteminin geliştirilmesi, modellenmesi ve simülasyonu gerçekleştirildi. Daha öncede ifade edildiği gibi, *SwarmNet* ortamının temel geliştirilme hedeflerinden biri, gezgin görevli ve biyolojik temelli yeni nesil yönlendirme algoritmalarının incelenmesi ve geliştirilmesidir. Gerçekleştirilen iki uygulamadan elde edilen sonuçlar, biyolojik temelli merkezi olmayan / dağıtık yaklaşımların üstünlüklerini göstermek amacıyla karşılaştırıldı. Geliştirilen deneysel çerçeve yoluyla üretilen ve grafikler halinde sunulan çıkışlar ile gerçekleştirilen iki uygulamadan elde edilen çıkışlar karşılaştırılma işleminde tabi tutuldu. Gerçekleştirilen işlemlerin son aşamasında, *DEVSJAVA* ortamının yüksek performansı kullanılarak *SwarmNet* ortamının kapasitesini belirlemek ve *DEVS* modelleme ve simülasyon yaklaşımının paralel / dağıtık uygulamalardaki gücünü göstermek amacıyla büyük ölçekli ağların incelenmesi ve topoloji analizinin yapılması işlemleri gerçekleştirildi. Büyük ölçekli ağların incelenmesinde bir başka hedef; İnternet kapsamındaki ağların analizi ve yönetimidir. Yapılan çalışmalarda, büyük ağların modellenmesinde düğüm başına düşen yönlendirme veritabanını azaltmak için arı kolonilerinden esinlenerek özel bir '*kümeleme' (clustering)* yöntemi geliştirildi ve DEVS birleşik model kavramı kullanılarak *SwarmNet* içinde uygulandı. Kullanılan yöntemin uygulanabilirliği ve avantajları grafikler ile desteklendi.

6.2 RIP Yönlendirme Algoritması Uygulaması

Ağ kaynakları ve hedefleri arasında en az maliyetli yolları üreten en kısa yol algoritmaları, paket anahtarlamalı ağlar için en yaygın kullanılan *'yol-üretim'* algoritmalarıdır [3]. En kısa yol algoritmaları arasında, dinamik programlama kavramlarına dayalı algoritmaları oluşturan 'uzaklık vektörü' sınıfı en yaygın kullanıma sahiptir. Başlangıçta '*ARPANET'* içinde kullanılan 'uzaklık vektörü algoritmaları' günümüzde internette işlevini sürdürmektedir. Bir ağların ağı içinde paketlerin tam hedefine ulaşması ve bir paket tarafından takip edilmesi gereken "en iyi" yolun bulunması için, ağın bir takım yönlendirme mekanizmalarına sahip olması gerekir. Bölüm 4'te detaylandırılan yönlendirme konusu, çok karmaşık bir işlemdir ve araştırmacılar tarafından yüksek ilgi çeken konulardan birisidir. Bu kısımda yönlendirme ile ilgili detaylara girmek yerine, uygulanacak olan *'Yönlendirme Bilgi Protokolüne' (Routing Information Protokol - RIP)*

odaklanılmaktadır. Aşağıdaki kısımda, RIP yönlendirme algoritmasının göze çarpan özellikleri sunulmaktadır.

6.2.1 Temel RIP uzaklık vektörü algoritması

İnternet türü ağlarda yaygın bir şekilde kullanılan uzaklık vektörü algoritmasının Bölüm 4'te açıklanması nedeni ile bu kısımda sadece RIP yönlendirme algoritması detaylandırılmaktadır. Örnek bir çalışma olarak temel bir uzaklık vektörü algoritması olan 'RIP algoritması' modellendi. RIP algoritmasının uygulanmasındaki temel amaç; *SwarmNet* ortamının klasik yönlendirme algoritmalarını modelleme ve simülasyon yeteneğinde olduğunu göstermektir.

RIP, ağ üzerinde düğümler arasında güncelleme mesajlarını dağıtmak amacıyla uzaklık vektörü algoritmasını ve yerel ağ altyapısını kullanmaktadır. Her bir yönlendirici kendisi ve ağdaki diğer tüm düğümler arasındaki "*uzaklık*" değerinden komşularını haberdar etmek için, komşularına periyodik olarak bir güncelleme mesajı gönderir. '*Uzaklık'* değeri daha önce de ifade edildiği gibi değişik değişkenlerle ifade edilebilir ve uygulamadan uygulamaya değişebilir. Bununla beraber, yapılan örnek uygulamalarda hesaplamayı kolaylaştırması nedeni ile ölçüt olarak *'hop sayısı'* seçilmiştir. Hop sayısı, doğrudan birbirine bağlı arabirimler için '1' olarak tanımlanmıştır.

RIP iki adet mesaj tipine sahiptir: *'istem' (request) ve 'cevap' (response)* mesajları. İstem mesajları komşu yönlendiriciye yollar hakkında sorular yöneltmek için kullanılırken, cevap mesajları bir isteme veya periyodik olarak tetiklenen güncellemelere cevap olarak kullanılmaktadır.

Periyodik güncellemeler her 30 sn.'de bir gönderilir. Tetiklenen bir güncelleme, yönlendirme tablosunun değişimine neden olan bir güncellemenin alındığı herhangi bir zamanda yapılabilir. RIP, yönlendirme bilgisinin ağ üzerinde oldukça yavaş yayıldığı bir algoritma olarak nitelendirildiği için, tetiklenen güncellemeler yönlendirme verisinin hızlı bir şekilde yakınsamasına (convergence) yardımcı olur. Yönlendirme algoritmalarının, genellikle bir yol hakkında tüm komşulardan gelen

bilgiyi kaydettikleri varsayılır. Kaydedilen yönlendirme veritabanına göre en düşük hop sayısına sahip olan seçilir.

Bu çalışmada uyguladığımız temel RIP yönlendirme algoritması en iyi şekilde aşağıdaki ifadeyle belirtilebilir: "D hedefi için M ölçütüyle G ağ geçidinden bir RIP güncelleme mesajı alınırsa, ilgili hedefin mevcut yolu ile karşılaştırılır. Eğer bu hedef için herhangi bir yol yoksa, G'ye eşit bir sonraki hop değerine sahip ve M+1 maliyetli bir yol oluşturulur. Mevcut yolun G'yi bir sonraki hop olarak belirtmesi durumunda, yolun maliyeti M+1 olarak belirlenir. Mevcut yolun maliyetinin M+1'den büyük olması durumunda, maliyeti 'M+1' olarak ve bir sonraki hop değeri G olarak belirlenir."

Gerçekleştirilen, uygulamalarda yer verilen bir başka önemli RIP özelliği; her bir yola atanan zaman aşımı değeridir. Herhangi bir hedef için en kısa yol yönlendirme tablosuna eklendiğinde, algoritmaya göre daha büyük bir ölçüt değerine sahip başka bir yol tarafından geçersiz kılınamaz. Bir yönlendiricinin arıza yapması durumunda, diğer yönlendiricilerin arızalı yönlendirici üzerinden giden yolların artık geçerli olmadığından haberi olmaz. Güncelleme mesajları aracılığıyla yolun geçerli olmadığını tespit eden yönlendiriciler, alternatif yollar bulmak zorundadır (eğer varsa). Herhangi bir alternatif yolun karşılaştırmada tercih edilmeyen bir ölçüte sahip olması olasıdır ve bu sebeple geçersiz yolları kaldırma yöntemi olmaksızın mevcut yol değiştirilmeyecektir. Bu problem RIP içinde, her yola ilişkili bir zaman parametresi katılarak çözümlenir.

6.2.2 SwarmNet'te uyarlama işlemi

Düğümler ve linkler gibi temel anahtarlama elemanlarından oluşan bir ağ üzerinde yönlendirme algoritmalarını uygulamak ve geliştirilen değişik trafik şekilleri altında test etmek için, DEVSJAVA altında çeşitli uygulamalar oluşturuldu. Düğümler ve linkler tanımlandıktan sonra, DEVS birleşik model tanımlamaları kullanılarak değişik topolojilerde ağlar meydana getirildi. Daha sonra deneysel çerçeve yoluyla ağ belirli bir trafik yüküne maruz bırakılıp, ağın davranışı izlendi ve elde edilen sonuçlar grafikler halinde bu bölümün sonunda sunuldu.

6.2.2.1 Düğüm ve link parametreleri

SwarmNet modelleme ve simülasyon ortamında değişik senaryoları çalışmak ve test etmek mümkündür. Geliştirilen örnek uygulamalarda yapılan deneylerde gerçek dünya şartları gerçekleştirilmeye çalışılmıştır ve bütün bileşenlerin parametreleri gerçek dünyaya uygun olarak seçilmiştir. Bununla birlikte, değişik senaryoları uygulamak amacıyla ağın davranışını etkileyen simülasyon parametreleri simülasyon çalışmasından önce ayarlanabilir.

RIP algoritması *SwarmNet* içinde modellenirken, Bölüm 5'te tanımlanan düğüm atomik DEVS modelleri yönlendiriciler olarak tasarlanmış ve davranışları RIP algoritmasına uygun hale getirilmiştir. Düğümler komşuları sayısı kadar arabirime sahiptir. Ağdaki bütün düğümler farklı IP adreslerine ve 1 MB'lık belleğe sahiptirler. Düğümler için 1MB'lık bellek miktarının seçilmesindeki amaç; yeterli boş alan olmaması nedeniyle düğümlerde oluşan paket kayıplarını azaltmak ve tıkanıklıkları linkler üzerinde gerçekleştirmektir. Geliştirilen uygulamada, ağda darboğazlar (bottleneck) oluşturmak amacıyla link kapasitelerinin sınırlandırılması yöntemi kullanıldı.

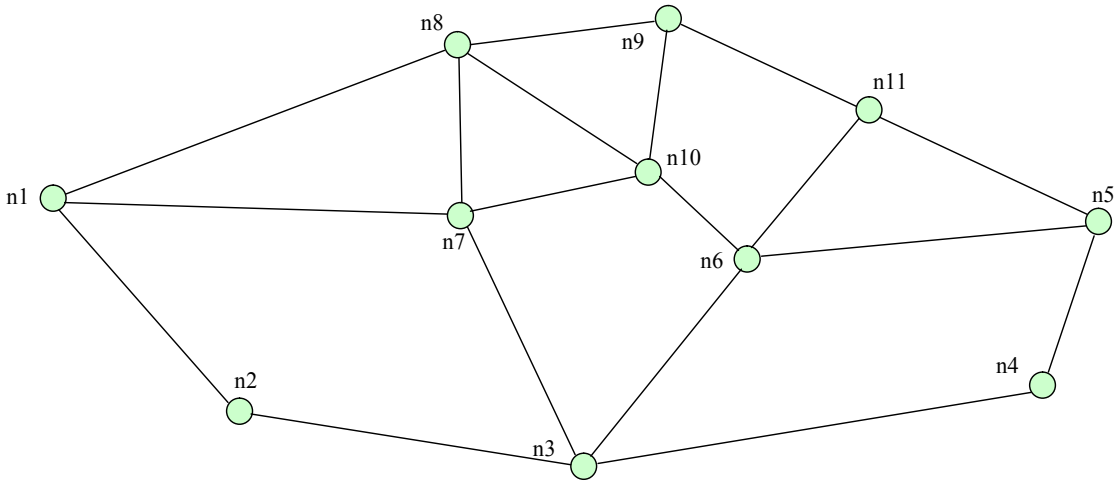
Düğümlerin paket işleme hızlarının *1 msn* seçilmesine rağmen bir ağ içerisindeki bütün düğümlerin parametreleri istenmesi durumunda farklı değerlerde seçilebilir. Bir düğüm birim zamanda sadece bir paket işleyebilir. Dolayısıyla bir düğüm işlem yaparken arabirimlerinde birine yeni bir paket geldiğinde bu belleğe yerleştirilirken, bellekte boş yer yoksa paket atılır. Düğüm bir paketi işledikten sonra, giden paket kuyruğuna yerleştirir ve ilgili linkten ağa yollar.

Sınırlı kapasiteye sahip link modelleri düğümler arasındaki bağlantıları temsil ederek iletişim hatlarını oluşturmaktadır. Linkler, iki düğüm arasında aynı anda çift yönlü (dublex) trafiği destekleyecek şekilde tasarlanmıştır. Ancak kodları üzerinde yapılacak çok az bir değişiklikle tek yönlü (simplex) linkler de modellenebilir. Bütün uygulamalarda kullandığımız link modeli çift yönlüdür. Deneylerde, linklerin yayılım gecikmeleri 1 msn ile 10 msn arasında, linklerin bant genişliği ise 193000 bayt/sn (1,5 Mbps) ile 1250000 bayt/sn (10 Mbps) arasında değişmektedir. Bir

linkin bant genişliğinin bir paketi iletmek için yeterli olmaması durumunda paket kaybı oluşacağı unutulmamalıdır.

6.2.2.2 Ağ modeli

Geliştirilen çerçevenin verimliliğini değerlendirmek için, çeşitli ağ modelleri oluşturuldu ve oluşturulan modellerle DEVSJAVA altında çeşitli deneyler gerçekleştirildi. Geliştirilen ağların bileşen sayısı 28 ile 3500 arasında değişmektedir. Bu çalışmada modellenen en basit ağ, 11 düğüme ve 17 çift yönlü linke sahip bir graf modelidir (Şekil 6.1). Yapılan çalışmalar sırasında gerçekleştirilen 11 düğüm ve 17 link sayısına sahip "basit ağı" oluştururken temel amacımız ağın çalışmasını ve uygulanan algoritmaların çalışmasını incelemek, daha fazla bileşen sayısına sahip büyük ölçekli ağların çalışılmasının amacı ise İnternet türü ağları modellemek ve davranışlarını gözlemlemenin yanında *SwarmNet* ortamının kapasitesini belirlemektir. Basit ağda bir düğüm genellikle coğrafik olarak komşu bir düğümle bağlanmıştır. Yönlendirme algoritmalarının incelenmesi, modellenmesi ve karşılaştırma işlemlerinin gerçekleştirilmesi için basit ağ yapısının kullanımı tercih edilmiştir. Çalışmalar sırasında kullanılan basit ağın bir deneysel çerçeve ile birlikte bağlantısı ve DEVSJAVA altında ekran çıktısı Şekil 6.2'de görülmektedir.

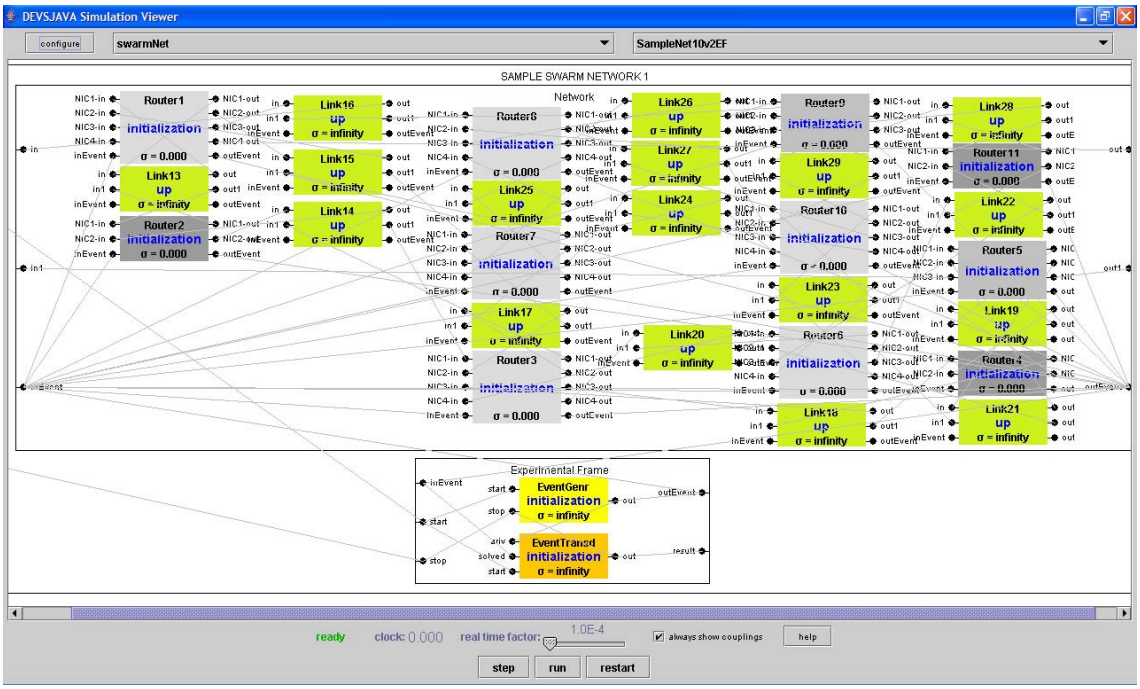


Şekil 6.1 11 düğümden ve 17 linkten oluşan basit ağ

6.2.2.3 Trafik modeli

Yapılan çalışmada, ayrık olay yaklaşımının kullanılması nedeni ile herhangi bir düğümün ağa bir paket göndermesi ve bir linkin veya düğümün çalışmaz duruma gelmesi, ağda meydana gelen olaylar olarak tanımlanıldı. Simülasyon sırasında ağda meydana gelen olayları önceden programlamak amacıyla kullanılan olay üreteci detayları Bölüm 2'de verilen deneysel çerçevenin bir bileşenidir.

Daha öncede belirtildiği gibi ağı oluşturan bütün elemanlar olay üreteciyle doğrudan bağlı bir '*inEvent'* girişine sahiptir. *SwarmNet* modelleme ve simülasyon ortamında bütün bileşenler '*inEvent'* portundan gelen mesajları diğer portlara göre daha farklı değerlendirir. Dolayısıyla, ağ çalışması sırasında herhangi bir link düğümü önceden programlanabileceği gibi paket olayları da trafiği üretmek için kullanılabilir. Ağdaki trafiği oluşturan bütün kontrol paketleri düğümlere yerleştirilmiş bulunan bir kovan tarafından ve veri paketleri trafik üreteci tarafından üretilir. Uygulamada, veri paketleri rasgele seçilmiş kaynak / hedef adresleri ile birlikte oluşturulur ve veri paketinin kaynak düğümünün '*inEvent'* portuna gönderilir. Daha sonra paketin takip edeceği yol mevcut yönlendirme tabloları tarafından belirlenir ve paket ağ içerisinde hedefine doğru yoluna devam eder. Paket yol boyunca her bir düğümde mevcut kapasiteyi rasgele seçilmiş bir boyut kadar düşürür. Eğer paketin yolu üzerinde herhangi bir düğüm üzerinde kapasite tamamen doluysa bu paket atılır.



Şekil 6.2 Basit bir ağın deneysel çerçeve ile birlikte DEVSJAVA ekran görünümü.

Paket hedefine vardığı zaman, hedef düğümün '*outEvent'* portundan olay dönüştürücüye yollanır. Olay dönüştürücü, olay üretecinden aldığı verilerle paketin verilerini karşılaştırır ve sonuçları bir dosyada saklar. Paketin ağdan atıldığı durumlarda paket olay dönüştürücüye erişemeyeceği için kayıp paketlerin sayısı olay dönüştürücü tarafından kolayca belirlenebilir. Olay üreteci tarafından üretilen trafik tek biçimli rasgele bir dağılımla üretilir.

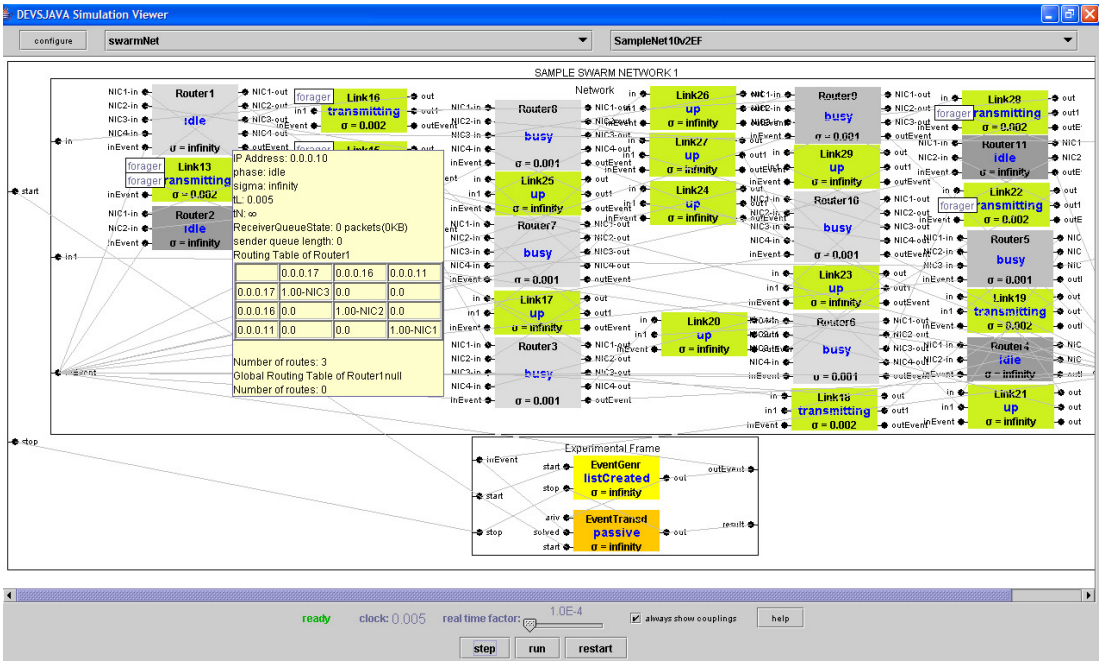
Simülasyon deneylerinde, yukarıda verilen ağ modeli 1 sn boyunca trafiğe maruz bırakılarak sonuçlar gözlemlenmektedir. Olay üretecinden her 1ms'de rasgele bir paket üretilmesi nedeni ile 1s boyunca ağa 1000 adet paket yollanmaktadır. Böyle bir ağır trafik yükü İnternet ortamıyla benzerlikler taşımakta ve paketlerin boyutu 100 Bayt ile 100 KBayt arasında değişmektedir. 1 s'lik süreden daha fazla sürelerde simülasyonlar deneyleri yapılsa da, sonuçların değişmediği gözlemlendiği için simülasyon süresi bütün deneylerde 1 s'dir.

6.2.2.4 Yönlendirme veritabanının kurulması

Ağı oluşturan düğümler linkler ile birbirine bağlanıp ağ oluşturulduğunda, bileşenler bulundukları ortamlar hakkında herhangi bir bilgiye sahip değildirler. Düğümlerin hesaplama birimleri olarak işlev görebilmeleri için bütün bir ağın görüntüsüne / bilgisine sahip olmaları gerekmektedir. Bunu sağlamak için, bütün düğümler komşularıyla kendilere arasındaki linklere *'1'* maliyet değeri atanmış başlangıç yönlendirme tablolarını oluştururlar. Bu başlangıç tablolarını oluşturma işleminde, ağ içerisindeki bütün bileşenler *kurulum* (initialization) aşamasındadır ve her bir düğüm komşu düğümlere RIP algoritmasına göre birer istem mesajı gönderir (Şekil 6.2). Bu uygulamada, yönlendirici istem mesajları ve cevap mesajları olarak sırasıyla '*scout'* ve '*forager'* paketleri kullanılır.

Ağ içerisinde iletilen paketler düğümler tarafından değerlendirilerek, başlangıç yönlendirme tabloları oluşturulur. Şekil 6.3'te basit ağda linkler üzerinde '*forager'* paketlerin iletilmesi ve başlangıç yönlendirme tablolarının oluşturulması görülmektedir. Daha sonra, başlangıç tablolarını taşıyan mesajlar ağda dağıtılarak bütün düğümlerin ağın bütününü temsil eden bir yönlendirme veritabanına / tablolarına sahip olması sağlanır. *SwarmNet* ortamı içerisinde kurulum aşaması olarak adlandırılan bu işlem, ağ yönetimi literatüründe '*yakınsama'* olarak adlandırılır. Basit ağ deneyinde RIP algoritması için yakınsama değeri yaklaşık 95 ms olarak son derece düşük bir değer elde edildi. Bütün ağ modelleri için yakınsama değerleri büyük ölçekli ağların incelendiği Şekil 6.12'de verilmektedir.

Yakınsama kavramı sadece ağın kurulumu sırasında değil, çalışma esnasında yönlendirme tablolarının güncellenmesinde de önemli bir ölçüttür. Yakınsama, ağ veritabanının güncellenmesi sırasında, ağdaki bütün düğümlerin aynı veritabanına sahip olması işlemidir. Yönlendirme algoritmalarının hızlı yakınsama sürelerine sahip olması istenir.



Şekil 6.3 Basit ağda başlangıç yönlendirme tablolarının oluşturulması ve *forager* paketleri.

6.2.3 Simülasyon sonuçları

RIP yönlendirme algoritması Şekil 6.1'de gösterilen ağa uygulanarak, DEVSJAVA altında çeşitli simülasyon deneyleri gerçekleştirildi. Deneyler sonucunda elde edilen sonuçlar grafikler halinde aşağıda sunulmaktadır. Ağ içerisinde gerçekleşen olayların izlenmesini kolaylaştırmak amacıyla bileşenler alacakları duruma göre renklendirilmiştir ve çalışma anında ağdaki bazı olaylar (paket kaybı, link tıkanıklığı, vb.) görsel olarak izlenebilmektedir. Görselliği / kullanışlılığı artırmak ve simülasyon esnasında ağın bütün kısımlarını izlemek için *SwarmNet* ortamına farklı özellikler eklenmiştir.

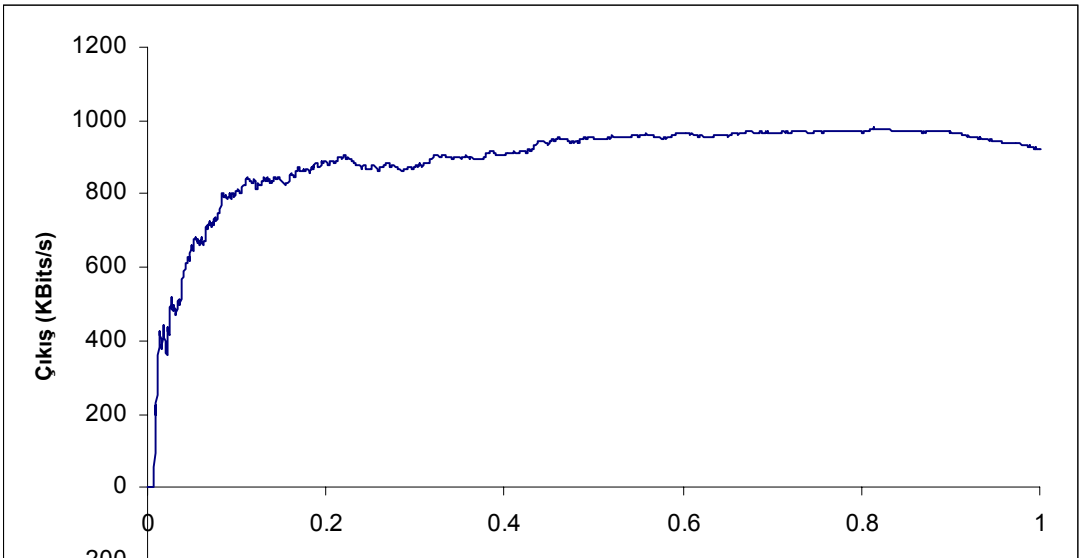
Yönlendirme algoritmalarının test edilmesi işleminde *ağ çıkışı*, *ortalama paket gecikmesi* ve ağda birimler üzerindeki *ortalama yük değişimi* kullanıldı. Ağ çıkışı (throughput); bir birim zamanda ağ üzerinden geçen paketlerin sayısıdır. Belirli bir zaman aralığında olay dönüştürücüye varan bütün paketler tarafından maruz kalınan 'ortalama gecikme' ağın genel durumu hakkında bilgi veren bir diğer parametredir. Bunlardan başka, düğümlerin ve linklerin tamponlarında / kuyruklarda bekleyen paketlerin sayısı, tıkanıklıkları belirlemeye olanak tanıyan diğer bir ölçüttür.

Performans değerlendirme ölçütleri kullanılarak, olay dönüştürücü içinde bir takım hesaplamalar yoluyla simülasyon sonuçları elde edilebilir.

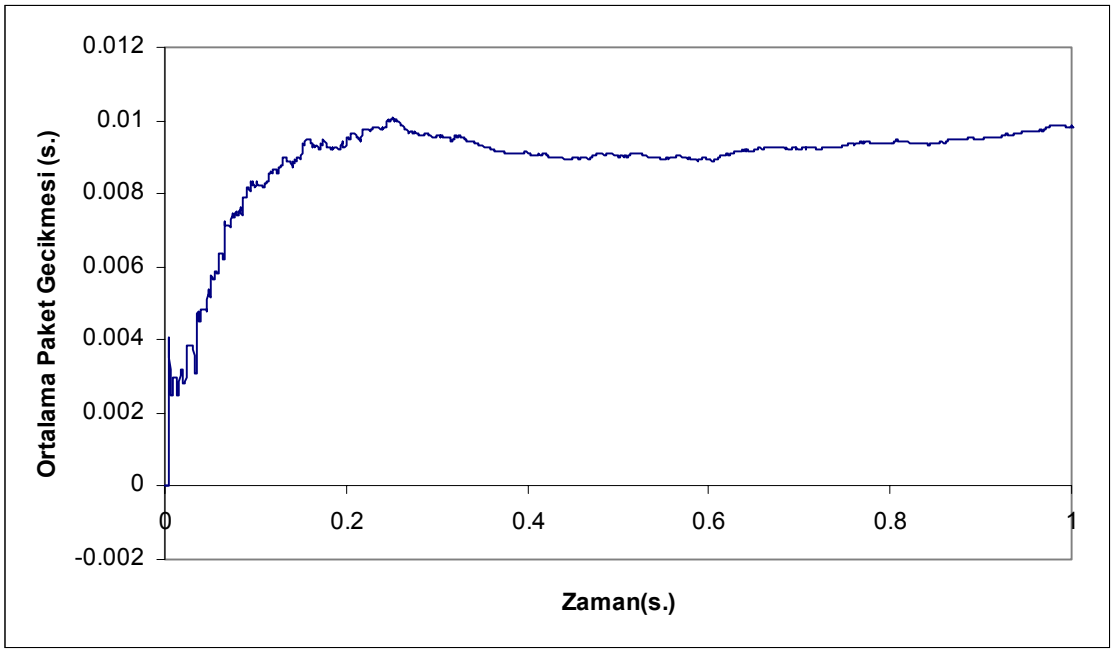
SwarmNet altında 1s'lik simülasyon zamanı P4 2.4 işlemcili ve 512 MB belleğe sahip bir bilgisayarda birkaç dakikalık bir zamanda tamamlanmıştır. Sonuçlar olay dönüştürücü tarafından yorumlanmakta ve CSV (comma seperated values) dosyalarında saklanmaktadır. Olay dönüştürücü tarafından üretilen CSV dosyaları yardımıyla elde edilen ve Şekil 6.4, Şekil 6.5, Şekil 6.6'da verilen grafiklerde, yukarıda bahsedilen performans ölçütlerinin zaman üzerinde değişimi görülmektedir.

Şekil 6.4'te, ağ çıkışı 1 msn gibi bir sürede en yüksek değerini almakta ve simülasyon sonuçlanana kadar ortalama değerini korumaktadır. Ağ çıkışına hızlı bir şekilde ulaşılması ve çıkışın çalışma boyunca sabit kalması, geliştirilen algoritmanın yük dengeleme yaptığının göstergesidir. Bununla birlikte, düğümlerin sınırlı sayıda link kullanılarak birbirine bağlı olmasına ve ağır trafik şartlarına rağmen RIP uygulamasında herhangi bir paket kaybı yaşanmamaktadır. Ancak, bazı linklerde ve düğümlerde geçici olarak aşırı yüklenme gözlemlenmiştir. Aşırı trafik artışlarına sistem hızlı cevap verdiğinden dolayı ağ çıkışında aşırı dalgalanmalar görülmemektedir.

Şekil 6.5'te 1 saniye boyunca ağ üzerinden geçen paketlerin maruz kaldığı ortalama gecikmenin zaman üzerinde değişim grafiği görülmektedir. Ortalama gecikme; paket üretecinin paketi ürettiği zaman ile paketin olay dönüştürücüye vardığı zaman arasındaki fark olarak hesaplanmaktadır. Şekilden de görülebileceği gibi paketler için ortalama gecikme 9 ms olarak ölçülmüştür. Gecikmenin tutarlı ve kabul edilebilir bir seviyede kalması, ağın tıkanıklıkları yönetebildiğinin bir göstergesidir.



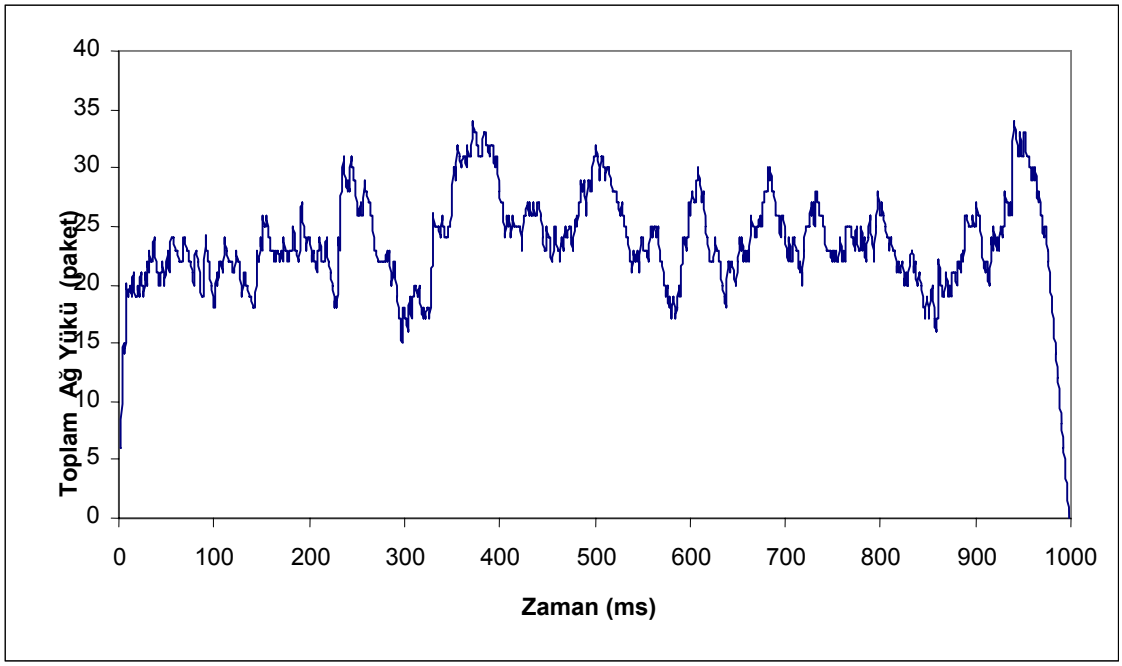
Şekil 6.4 Ağ içerisinde oluşan trafik çıkışı



Şekil 6.5 Ortalama paket gecikmesi

Şekil 6.6'da ağ bileşenleri üzerindeki yükün zamanla değişimi görülmektedir. Ağ yükü birim zamanda ağda kuyruklarda bekleyen paketlerin sayısı olarak olay dönüştürücü tarafından hesaplanmaktadır. Kuyruklarda bekleyen paket sayısının birim zamandaki ağ üzerindeki toplam yük olarak ele alındığı bu değerlendirmede paket kayıpları ve kontrol paketleri ihmal edilmektedir.

Bu grafikte toplam yük değeri hiçbir zaman aşırı değerler almamakta, yükün artma eğilimi gösterdiği durumlara ağ gecikmeden cevap vermekte ve yük değişiminin belirli bir aralıkta kalması sağlanmaktadır. Yönlendirme sistemi yükün artış eğilimine milisaniyeler mertebesinde cevap vermektedir. Bu sebeple yük değişimi grafiği, inişler ve çıkışlar şeklindeki bir testere dişini andırmaktadır.



Şekil 6.6 Ağ yükünün zamanla değişimi.

Sonuç olarak, RIP yönlendirme algoritması uygulamasında simülasyon süresince bütün ağ kaynaklarının dengeli bir şekilde kullanıldığı ve yönlendirme tablolarının tutarlı olduğu gözlemlenmiştir. Yukarıdaki grafiklerden kapasite ve kaynakların kullanımının son derece dengeli olduğu anlaşılmaktadır. Grafikler *SwarmNet* ortamının bir ağ simülatörünün fonksiyonlarını yerine getirdiğinin göstergesidir. Bu sebeple, sonuçlar bir ağ simülatörü olarak tasarlanan *SwarmNet* modelleme ve simülasyon ortamının geçerlenmesi ve doğrulanması işleminin birer parçalarıdır.

6.3 Arılarla Yönlendirme Uygulaması

Sosyal canlı topluluklarının dağıtık kontrolde ortaya çıkabilen grup-seviyeli adaptif özelliklerin en önemli örneklerine sahip olmaları nedeni ile imalat, planlama, taşıma ve iletişim alanlarında çözülmesi zor kaynak tahsisi problemlerinin çözümünde temel bir ilham kaynağı olmuşlardır. Bu sebeple, sosyal topluluklarda karşılaşılan geri besleme mekanizmalarını, koloni organizasyonu ve koloni düzenlenmesi konularının daha iyi bir şekilde anlaşılması, klasik merkezi kontrol yönteminin

ihtiyaca cevap veremediği alanlarda önemli "gerçek dünya" avantajları sağlamaktadır [19].

Yapay zeka alanı, farklı mühendislik, yönetim, kontrol ve hesaplama problemlerini çözmek için biyolojik bilgi ve tekniklerden faydalanır [17]. Doğal sistemler, basit yapılı organizmaların, dinamik bir şekilde birbirleriyle etkileşerek son derece karmaşık görevleri yerine getirebilen sistemlerin oluşturabileceğini göstermektedir. Örnek olarak; grup halinde yaşayan canlıların etkileşiminden ortaya çıkan oğul zekası, bilgisayar ağlarının yönetimi gibi karmaşık optimizasyon problemlerinin çözümünde ilham kaynağı olmuştur. Oğul zekası kullanan yönlendirme algoritmaları aynı anda etkileşim yapan birçok varlığın etkileşimine dayanarak ağlardaki çeşitli problemleri çözebilmektedir. 4. bölümde oğul zekasına ve karıncaların davranışlarından esinlenilerek geliştirilmiş algoritmalar açıklanmıştır. Bu kısımda geliştirilen modelleme ve simülasyon çerçevesinde bal arılarının davranışlarından üretilen basit kuralların ağ yönlendirme işlemine uygulanması gösterilmektedir. Ayrıca, büyük ölçekli ağların modellenmesi ve simülasyonu için ekolojik yaklaşımlar sunulmaktadır. Daha sonra elde ettiğimiz sonuçlar daha önceki uygulamaların sonuçlarıyla karşılaştırılmaktadır.

6.3.1 Bal arılarının nektar arama davranışları

Balarıları (*Apis mellifera*) ve karıncalar gibi büyük ölçekli biyolojik sistemler, kendilerini ölçekleme, uyarlama ve hayatta kalma konularında gelişmiş mekanizmalara sahiptir [49]. Örnek olarak inceleyebileceğimiz arı kolonisinde, kovan içerisinde halledilen işlerin çoğu herhangi bir merkezi kontrol otoritesine bağlı olmadığı için (hatta çoğu durumda kraliçe arıya bile) arı kolonileri çok büyük sayıda bireye sahip kolonilere ölçeklenebilirler. Otonom olarak hareket eden arılar, yerel şartlardan ve diğer arılarla yapılan yerel etkileşimlerden etkilenmektedirler. Örneğin, arılar kovanı inşa ederken sadece tamamlanmış altıgen hücrelerin yapısını takip ederler ve bu işlem esnasında kovanın inşasını kontrol eden herhangi bir yönetici arı yoktur. Arı kolonisi dinamik şartlara kendisini uyarlayabilir ve enerji tüketimine bağlı olarak besin kazancını optimize edebilir: Kovandaki bal miktarının çok düşük olması durumunda, büyük miktarda bal toplayıcı arının kovanı terk ederek nektar aramaya gitmesine karşılık, kovanın bal bakımından dolu olması

durumunda, arıların çoğu kovanda kalarak istirahat ederler. Arı kolonisi tek bir arıya (kraliçe arıya bile) bağlı olması nedeni ile, kovandaki bazı arılar ölse bile koloni yaşamını sürdürür. Gerçekte, bir arı kolonisinin istenen karakteristikleri olan ölçeklenebilirlik, uyarlanabilirlik ve yaşamlarını sürdürebilirlik tek bir arıda bulunmaz, daha çok koloni içindeki bütün arıların kolektif hareketlerinden ve etkileşimlerinden ortaya çıkar.

Böcek toplulukları, bireylerin yüksek derecede koordineli ve entegre bir birim oluşturduğu karmaşık adaptif sistemlerdir [50]. Merkezi olmayan basit yapılı bir kontrolle, işçiler birlikte çalışma yeteneğine ve kolektif olarak bir bireyin yapamayacağı görevleri çözme yeteneğine sahiptir [21]. Balarısı, hakkında bir çok araştırma yapılmış bir sosyal canlı örneğidir ve adaptif grup davranışının ortaya çıkmasını anlamada mükemmel bir modeldir. Balarısının grup seviyesindeki koordineli aktivitelerinin ve koordinasyonu sağlamak için işçi arılar tarafından kullanılan mekanizmaların bir çoğu bilinmektedir [51]. Bir adaptif birim olarak bireyleri bir birine bağlamada önemli bir faktör bireyler arasındaki *işaretler (cues)* ve *sinyaller* biçimindeki bilgi transferidir [52]. Sinyal, kasıtlı olarak yapılan bir iletişim aksiyonudur. Sinyale örnek olarak balarısı sallanma dansı verilebilir. İşaret, bilgi taşıyan yapı veya davranıştır. Örneğin, nektar ararken harcanan zaman ve kovandaki karbon dioksit seviyesi işarettir.

Balarılarında an az 17 sinyal ve bunun iki katı kadar işaret keşfedilmiştir [53]. Bu iletişim araçlarından dört veya beş sinyal ve bir takım işaretler, hem çiçeklerden nektar toplama hem de toplanan nektarı peteklere depolama gibi işleri kapsayan nektar hasat yapma prosesiyle uğraşan işçi arıların sayını düzenler. Bal arılarında, nektar arama işlemi iki alt göreve bölünmüştür. Nektar arama işleminde bal araştıran işçi arılar malzemeyi toplarlar ve kovana götürürler. Ancak, toplanan nektarı kendileri depolamak yerine, hücrelere depolama işlevini yerine getiren alıcı arılara transfer ederler. Nektar mevcudiyeti ve toplama işlemi hızla ve kestirilemez bir şekilde dalgalanabilir. Bu durum, toplam bal araştırma çalışmasını değiştirmeyi ve araştırıcı ile alıcı arı gruplarının bağıl çalışma kapasiteleri arasında bir dengeyi gerekli kılar [56]. Bu çalışma kapasitelerinin eşleşmediği durumlarda, işçiler üzerinde verimsiz bir şekilde iş bölümü yapıldığından nektar işleme oranı azalır. Balarıları, sürekli değişen ve geçici nektar kaynaklarından faydalanmak için daha

fazla işçi arıyı kaynaklara yönlendirmek ve nektar araştıran arıları ile kovanda balları depolayan arıların bağıl çalışma kapasitelerini dengelemek için son derece mükemmel mekanizmalara sahiptir [51]. İşçi arıların bu şekilde görev paylaşımına tabi tutulmaları beş farklı sinyal ile yapılır: "sallanma dansı", "titreme dansı", "dur sinyali", "sarsma sinyali" ve "işçi arı yönlendirme" [50].

Balarısı kolonilerinde, *gözcüler* verimli nektar sahalarını araştırır ve daha sonra kovandaki boşta bulunan işçi arıları bu bölgelere sallanma dansı yardımıyla yönlendirir. Kovanda gerçekleşen bu işlem literatürde '*keşfet-yönlendir' (scout-recruit) sistemi* olarak adlandırılır [52]. Bu sistem aracılığıyla, koloni iki farklı nektar kaynağıyla karşı karşıya kaldığı zaman, tutarlı bir şekilde *en yüksek kazançlı* kaynağa odaklanır. Kazanç, nektarın lezzetinin, erişilebilirliğinin, miktarının ve kovandan uzaklığının bir fonksiyonu olarak tanımlanmıştır. Ayrıca, eğer nektar kaynağının kazancı / karlılığı değişirse, koloni o kaynağa olan ilgisini değiştirir.

Bal araştıran arılar yüksek enerji gerektiren nektar toplama işleminde ilave bir problemle karşılaşırlar: nektar varlığının değişken doğası ve dağılımı. Örneğin, birkaç gün içerisinde çevre şartları nedeni ile nektar oranının azlığından nektar fazlalığına değişebilir. Mevsimsel değişimler, iklim nektar toplamayı ve koloninin çalışma kapasitelerinin ayarlanmasını zor kılar. Bir koloni, bal araştırmak mümkünken kendi işçilerini kovanda çalışmaz halde bırakamaz. Ancak, nektar sahalarını araştırmak oldukça masraflı olduğu için çevrede az nektar olduğu durumlarda bütün potansiyel işçi arıların bal aramaya çıkması mümkün değildir. Bir koloni, nektar varlığını izlemek için yeteri kadar gözcüye gereksinim duyar ve iyi bölgeler bulunduğunda kovanda bal depolayan arıları kısıtlamayacak sayıda gözcü arının olmasını garanti eder.

6.3.2 Keşfet-yönlendir sistemi algoritması

Bu kısımda yukarıda ifade edilen balarılarının basit kurallarından ve davranışlarından esinlenerek yeni bir yönlendirme algoritması geliştirilmektedir. Bu algoritma diğer klasik algoritmalardan farklı olarak otonom ve zeki bir şekilde ağ yönlendirme problemlerini çözmeye çalışmaktadır. Temel çıkış noktamız, arıların nektar arama işlemi sırasındaki gözcülerin kovandaki işçi arılara en iyi nektar

sahasını sunması ve kovandaki arıları nektar sahalarına yönlendirmesi işlemi olarak adlandırılan keşfet-yönlendir sisteminin, ağda hedefe yönlendirilmesi gereken paket trafiğine uygulanmasıdır.

Bu tür algoritmalar dinamik topolojilere klasik algoritmalardan daha iyi bir şekilde adapte olabilir. Ayrıca, geliştirilen algoritmanın neden olduğu düşük ağ trafiği, hızlı yol keşfi gibi özellikleri de ilave avantajlarıdır. Oğul zekası ve dağıtık ağ yönetimi yaklaşımlarının avantajları Bölüm 4'te detaylandırılmıştır.

Algoritma, bal arılarının yaşantısıyla ağ sistemleri arasında bir takım benzerlikler kurularak geliştirilmiştir. Bu benzerlikler Tablo 6.1'de verilmektedir. Geliştirilen modelde, her bir düğüm bir arı kovanına sahiptir ve ağın bütünü arılar için bir nektar toplama bölgesi olarak kabul edilmektedir (Şekil 6.7). Balarıları nektar ararken olabildiğince çok miktarda nektar toplamak isterler. Nektar belirli bir link veya yol boyunca uçarak toplanabilir. Bir yol boyunca toplanan nektar miktarı, yolun hop sayısıyla ters orantılıdır. Diğer bir değişle, daha yüksek nektar miktarı daha düşük hop sayısına sahip bir yol boyunca uçulduğunda toplanabilir.

Ağın çalışması esnasında, yapay gözcü arılar ağda dolaşarak nektar sahalarını veya düğümler arası en kısa yolları araştırır. Daha sonra kovandaki işçi arılar olarak kabul edilen veri paketleri bu yollar üzerinden hedeflere yönlendirilir. Sonuçta, ağdaki bir paketi ağ üzerinde hedefine gönderme işleminde en iyi yolu belirlemek ve paketleri bu yoldan yönlendirmek ile bir gözcü bal arısının kovandaki işçi arıları en iyi nektar sahalarına yönlendirmesi arasında basit bir ilişki kurulmaktadır.

Tablo 6.1 Balarıları-dağıtık sistemler arasında kurulan ilişkiler.

Bal Arıları	**Ağ sistemi karşılığı**
Nektar sahası	Ağ
Nektar	Boştaki ağ kaynakları
Kovanlar	Düğümler
Gözcü Arılar (scouts)	Kontrol paketleri
Bal toplayan arılar (foragers)	Veri paketleri
Sallanma dansı ve diğer işaretler	Yönlendirme tablosu

Balarıları bir takım danslar ve işaretlerle iletişim yaparken, yapay arı sisteminde iletişim yönlendirme tabloları aracılığıyla olur. Sadece gözcü arılar yönlendirme tablolarını güncelleyebilir. Veri paketleri veya işçi arılar bu yönlendirme tablolarını kullanarak hedef noktalarına ulaşabilirler.

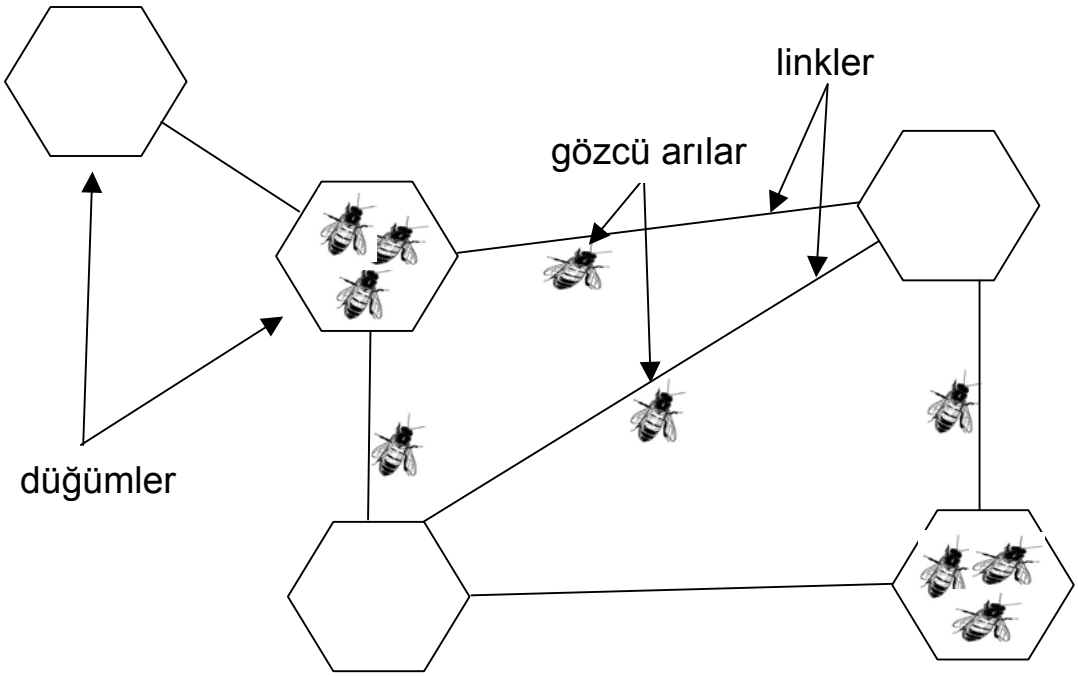
Geliştirilen uygulamada, bal arılarının nektar arama davranışlarının önemli yönleri korunmuş olsa da, dağıtık, dinamik bir ağ ortamına uyarlarken bir takım değişiklikler, yorumlar ve soyutlamalar yapılmaktadır. Balarılarının bal arama davranışlarının matematiksel olarak ifade edilmesine rağmen, geliştirilen uygulama kural tabanlıdır. Kural tabanlı yaklaşım sistemin gerçeklenmesini ve uygulanmasını kolaylaştırmaktadır. Ayrıca yüksek hızda çalıştıklarından dolayı sistem cevabı hızlıdır.

Algoritmada tüm yönlendirme kararları RIP algoritmasından farklı olarak rasgele yapılır. Bunu gerçekleştirmek için düğümler Bölüm 5'te açıklanan olasılıklı yönlendirme tablolarını kullanır. Olasılıklı yönlendirme tablolarında her bir hedef düğümün belirli bir olasılık değeriyle seçilmesi nedeni ile trafiği alternatif yollar üzerine yaymak mümkündür. Bu yöntem her bir paketin işlenmesini kolaylaştırır ve yeni yolların klasik yönlendirme sistemlerinden daha hızlı keşfedilmesine neden olur.

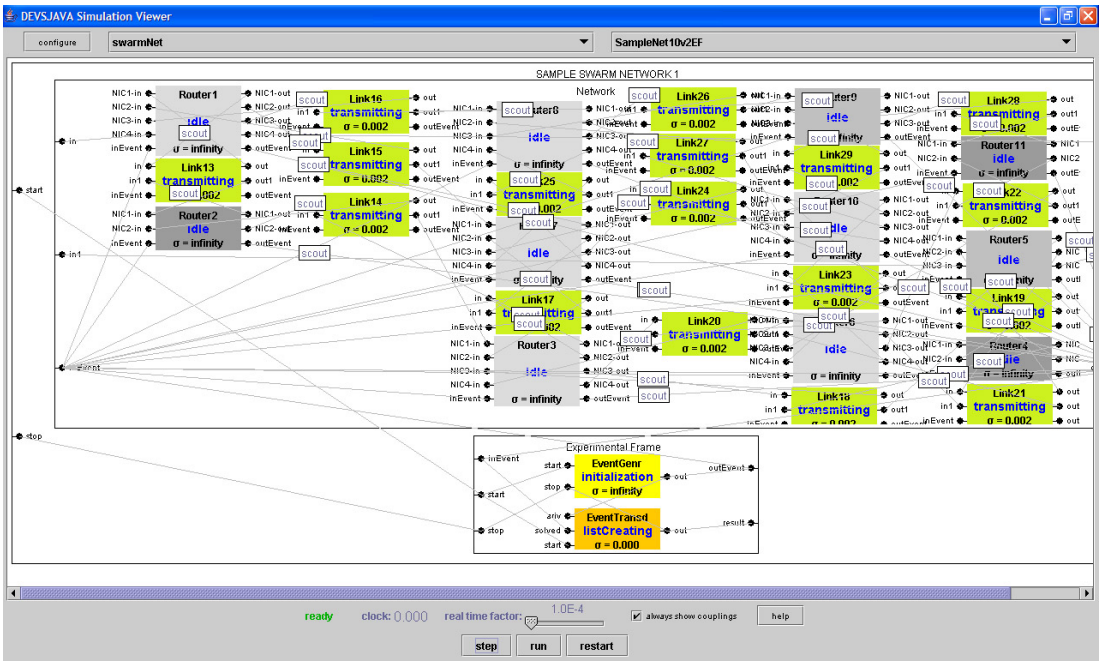
Gözcüler nektar ararken belirli bir zaman miktarıyla sınırlandırılır. Buradaki amaç; iyi yolları arayan arıların ağ kaynaklarını daha fazla işgal etmesini önlemektir. Ayrıca gözcü arılar ziyaret ettikleri linklerin listesini tutarlar. Bu liste kullanılarak ağda meydana gelebilecek döngüler engellenmiş olur.

Oğul zekası uygulamasında ağın kurulumu diğer algoritmalara göre biraz farklıdır. Ağı oluşturan düğümler başlangıçta ortamları hakkında herhangi bir bilgiye sahip değildirler. Düğümler komşularını keşfetmek için bütün arabirimleri üzerinden komşularına birer gözcü gönderirler (Şekil 6.8). Daha sonra, gözcülerinin elde ettiği bilgilere göre düğümler ilkel yönlendirme tablolarını oluşturur. Bu yönlendirme tablosunda her bir komşu düğüm için maliyet değeri olarak '1' değeri atanır. Daha

sonra algoritma ağ trafiği altında, uzaklık vektörü algoritmasıyla benzer bir şekilde bütün ağı kapsayan yönlendirme tabloları oluşturulur.



Şekil 6.7 Bir kovan ağının yapısı.



Şekil 6.8 Gözcü (scout) arıların ağda dolaşmalarını gösteren *SwarmNet* ekran çıktısı.

Daha öncede ifade edildiği gibi, geliştirilen algoritma basit kurallardan oluşur. Bu kurallar aşağıda sıralanmıştır:

- Bir paketin bir hedefe gönderilmesi gerektiği zaman, kaynak düğümü bir miktar gözcü arı üretir ve komşuları üzerinden ağa gönderir. Daha sonra bu gözcüler ağ üzerinde dolaşırlar ve ağın mevcut durumu hakkında bilgi toplarlar.

- Her bir düğümde gözcüler bir sonraki düğümü, o düğümün yönlendirme tablosundaki olasılık değerlerine göre seçerler.

- Bir gözcü arı hedefine ulaştığı zaman kendi kovanına yani kaynak düğümüne geri döner.

- Bütün gözcüler kovana geri döndüğü zaman, her bir arının kat ettiği yolların maliyet değerleri (hop sayısı) karşılaştırılır ve en düşük maliyete sahip yol seçilir.

- Gözcüler bir düğümü iki kez ziyaret edemezler. Bunu önlemek için her bir gözcü ziyaret ettiği düğümlerin listesini tutar (tabu list).

- Bir gözcü yeterince bant genişliğine sahip olmayan bir linki kullanamaz.

- Bir gözcü TTL değerini aştığı zaman otomatik olarak kendini ağdan kaldırır.

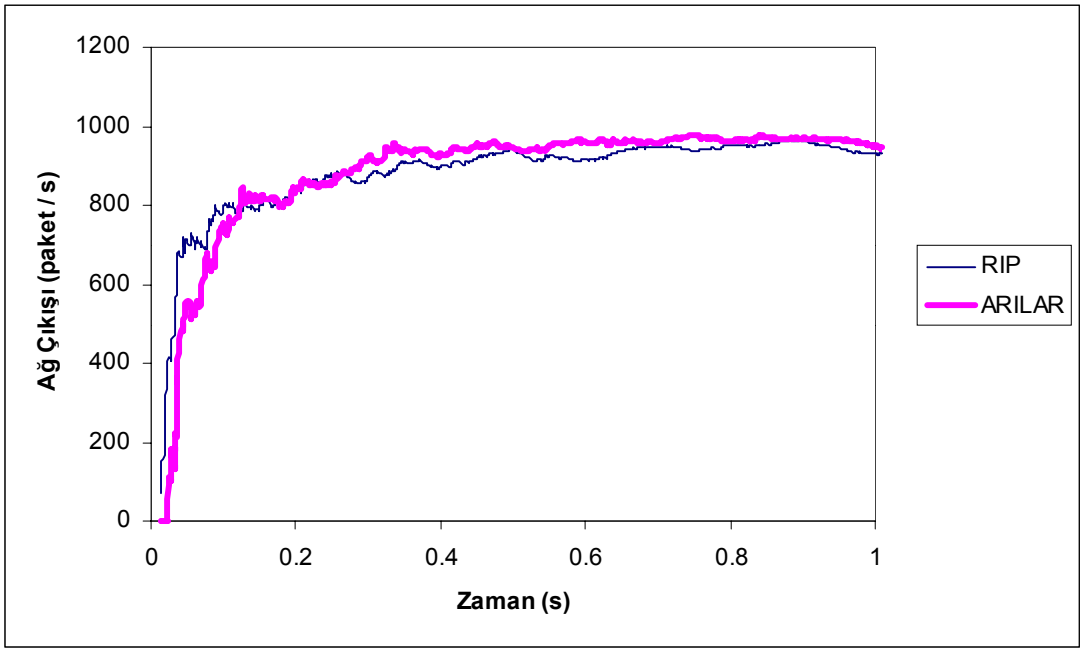
Yukarıdaki kurallar kullanılarak ağ yönetimi gerçekleştirilir ve tıkanıklıklar önlenebilir. Geliştirilen modelde, düğümler ve linklerin davranışları uygulamaya uygun olarak değiştirilmiştir. Yönlendirme tabloları diğer yönlendirme algoritmalarından farklı olarak olasılık değerlerinden oluşur. Bir hedefe giden bütün yolların olasılık değerlerinin toplamı 1'dir. Köklerini ekolojik sosyal canlı sistemlerinden alan olasılıklı yönlendirme yaklaşımı, yükün sadece tek bir yol üzerinden çok alternatif yollardan akışına izin verir. Paketlerin formatı ve ağ

arabirim kartlarının yapısı RIP algoritmasıyla aynıdır. Geliştirilen algoritmanın çalışma ve programlama prensiplerini gösteren akış diyagramları bu tezin sonunda EK A, B ve C'de verilmektedir.

6.3.3 Simülasyon sonuçları

Çalışma sırasında geliştirilen algoritma, RIP uygulamasında kullandığımız parametreler, ağ topolojisi ve trafik şartları kullanılarak test edildi. Daha sonra geliştirilen algoritmadan elde edilen çıkışlar ile RIP algoritmasından elde edilenler karşılaştırıldı. Karşılaştırmadan elde edilen sonuçlar grafik halinde Şekil 6.9 ve Şekil 6.10'da verilmektedir. Deneyler sonucunda elde edilen grafiklerden, ekoloji tabanlı algoritmanın RIP algoritmasına göre daha iyi performans gösterdiği görülmektedir.

Şekil 6.9'da verilen ağ çıkışı grafiklerinde, balarılarının optimum bir yük dengelemesi ile RIP algoritmasına göre daha iyi bir çıkış verdiği görülmektedir. Grafikten, geliştirilen algoritmanın daha geç maksimum değerini aldığı, ancak kararlı durumda daha yüksek bir çıkış verdiği anlaşılmaktadır. RIP algoritmasındaki gibi ağır trafik şartlarına rağmen paket kaybı gözlemlenmemektedir. Birim zamanda ağda bulunan kontrol paketi sayısı RIP algoritmasından fazla olduğu için ağ çıkışı RIP algoritmasından belirgin bir şekilde farklı değildir.

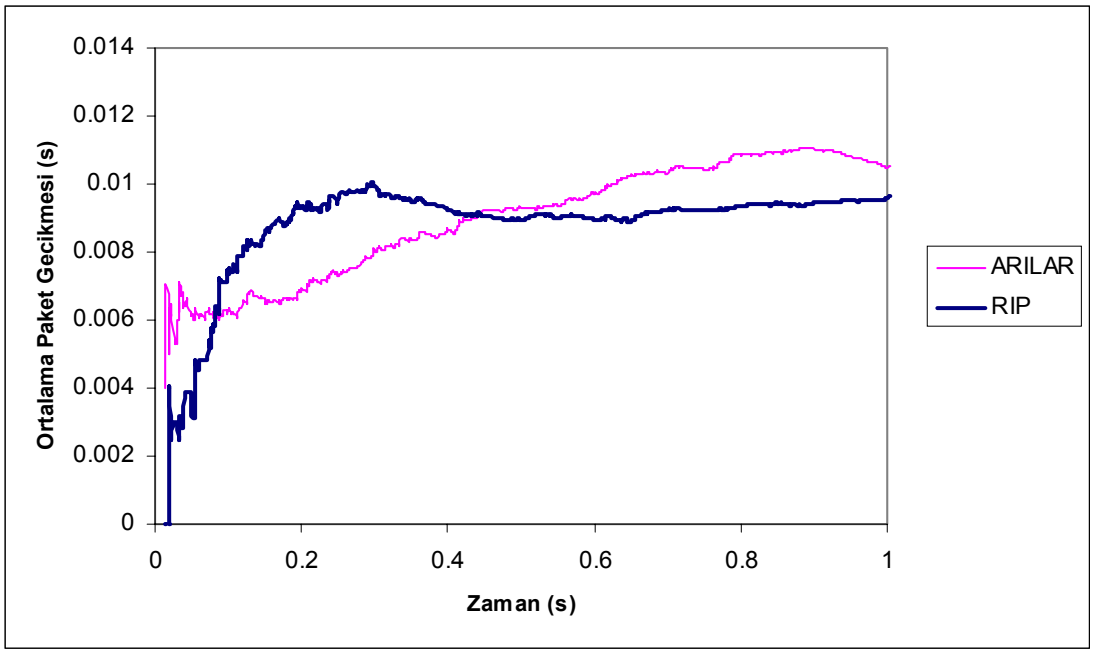


Şekil 6.9 RIP ve Arılarla gerçekleştirilen algoritmalarının ağ çıkış değerlerinin karşılaştırılması.

Şekil 6.10'da, her iki algoritmanın ortalama paket gecikmesi grafikleri görülmektedir. Ortalama gecikme, 0,5 ms'ye kadar RIP'e göre çok düşük değerler almakta daha sonra simülasyon sonuçlanıncaya kadar yüksek kalmaktadır. Bunun sebebi, olasılıklı yönlendirmenin paketleri iki düğüm arasındaki en kısa yolun haricindeki yollara yönlendirmesidir. Dolayısıyla paketlerin hedeflerine ulaşması daha fazla vakit almasına karşılık grafiğin ortalama değeri RIP ile aynı olmaktadır (9 ms).

Sonuç olarak, merkezi klasik yaklaşımlardan çok sosyal canlıların davranışlarında esinlenilerek geliştirilen algoritma, yaygın olarak kullanılan klasik bir algoritma olan RIP'e göre daha iyi performans göstermiştir. Önerilen algoritma, alternatif yolların keşfedilmesi ve kullanılması konusunda daha verimli ve hızlıdır. Ayrıca, aşağıdaki kısımda ele alınacak olan uygulamada görüleceği gibi, ölçeklenebilirlik konusunda son derece esnektir. Kontrol paketlerinin boyutu çok küçük olduğu için ağa fazladan yük getirmez. Olasılıklı yönlendirme ağdaki değişimlere daha çabuk adapte olduğu, herhangi bir hedef düğüm için alternatif yollar tanımlandığı için linklerin kopması ve düşmesi durumlarında ağ trafiği etkilenmez.

Geliştirdiğimiz model ölçeklenebilir, adaptif ve sağlam ağ uygulamalarının modellenmesi ve tasarımı için örnek bir çerçeve olmuştur. Önerilen algoritma, detaylı işlemler ve formüllerden daha çok kural tabanlı olduğundan simülasyon oldukça hızlıdır.



Şekil 6.10 Ortalama paket gecikmesi değerlerinin karşılaştırılması.

6.4 Büyük Ölçekli Ağların İncelenmesi

Bölüm 5'te ifade edildiği gibi büyük ölçekli ağların modellenmesinde karşılaşılan en büyük problem ölçeklenebilirliktir. Düğümlerde yönlendirme işleminde kullanılmak üzere bulunan veritabanı ağın boyutuyla artacağından, büyük bir ağın yönlendirme veritabanı bir düğümün işleyebileceğinden çok daha fazla olur. Bu durumda yönlendirme veri tabanının uygun bir basitlikte yeniden yapılandırılması gerekmektedir. Yönlendirme veritabanını basitleştirmek amacıyla Bölüm 5'te ayrıntılandırılan kümeleme sistemi geliştirilen ağlara uygulandı. Kümeleme, bir alt ağı daha üst seviye bir ağ içerisinde bir düğüme indirgeyerek yönetilebilir ağ boyutları sağlayabilmektedir. Bu durumda, tüm düğümlerin toplamında bir ağ hiyerarşisi meydana gelmektedir. Kümelere ayrılmış bir ağda düğümlerin adreslenmesi çoğu kez ağların hiyerarşisini yansıtır.

DEVS birleşim altında kapalılık özelliği birleşik bir modeli atomik bir model olarak ele aldığı için, DEVS birleşik modeli kümeleme yöntemiyle benzerlikler taşır. DEVS yaklaşımının hiyerarşik ve modüler yapısı, kümeleme yönteminin uygulanmasını kolaylaştırır. Yapılan uygulamada kümeleme düğümlerin adresleme seviyesinde halledilir ve her bir birleşik model bir takım '*sınır düğümlere'* sahiptir.

Bu sınır düğümleri diğer birleşik modellere bağlanmak için kullanılır. Sınır düğümleri içinde küme isimlerinin bulunduğu ilave bir yönlendirme tablosuna sahiptir. Bu yöntem ağ düğümleri içinde kayıtlı bulunan yönlendirme veritabanının boyutunu önemli ölçüde düşürür.

Bu kısımda, değişik boyutlara sahip ağlar yanında kolonilere bölünmüş ağlar oluşturuldu. Küçük boyutlu ağlar birebir kodlanarak oluşturulurken, büyük boyutlu ağlar yinelemeli bir algoritma kullanılarak meydan getirildi. Daha sonra değişik trafik şartları altında bu ağlar ayrı ayrı test edildi ve ağların performansı karşılaştırmalı olarak grafikler halinde sunuldu. Ağların isimleri, bileşen sayıları ve kaç koloniden oluştukları Tablo 6.2'de verilmektedir.

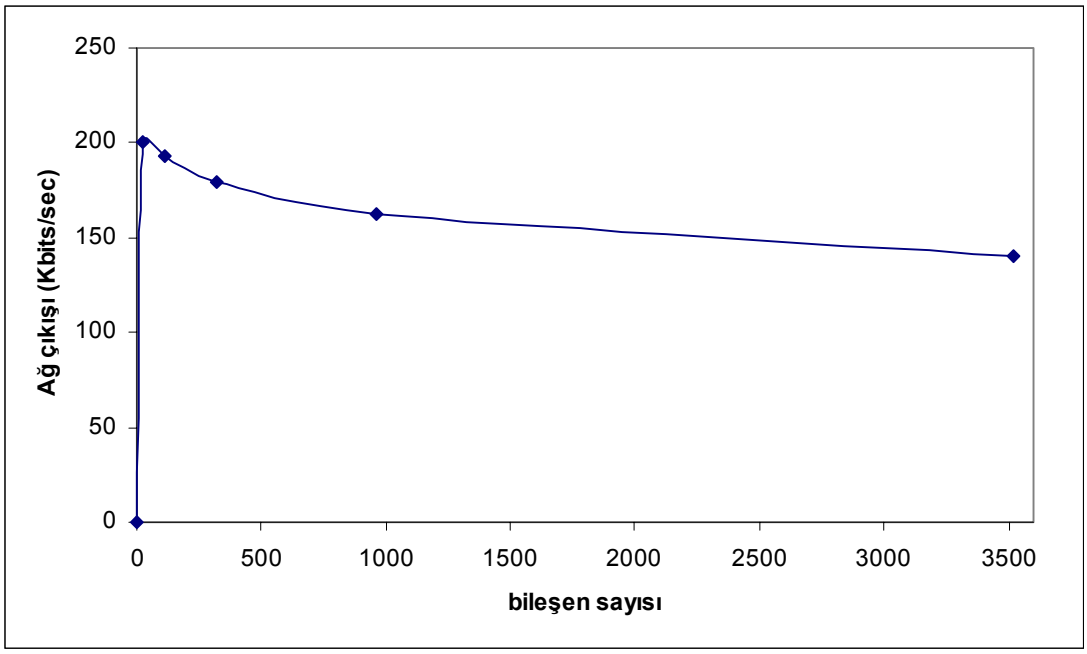
Tablo 6.2 Büyük ölçekli ağ modelleri

Ağ adı	Bileşen sayısı	Koloni sayısı
NET 1	116	4
NET 2	319	11
NET 3	960	87
NET 4	3520	125

6.4.1 Simülasyon sonuçları

Büyük ölçekli ağların yönetimi için geliştirilen yaklaşım çeşitli deneylerle geçerlenmeye ve doğrulanmaya çalışılmıştır. Değişik boyutlarda / yapıda ağlar geliştirilip benzer deneysel şartlarda test edilerek, bileşen sayısı ve ağ performans kriterleri arasındaki ilişki grafikler halinde sunulmaktadır. Geliştirilen simülatör altında, küçük ve normal ölçekli ağların simülasyon süreleri gerçek zamanda dakikalar, buna karşın büyük ölçekli ağların simülasyon süreleri saatler mertebesini bulmaktadır.

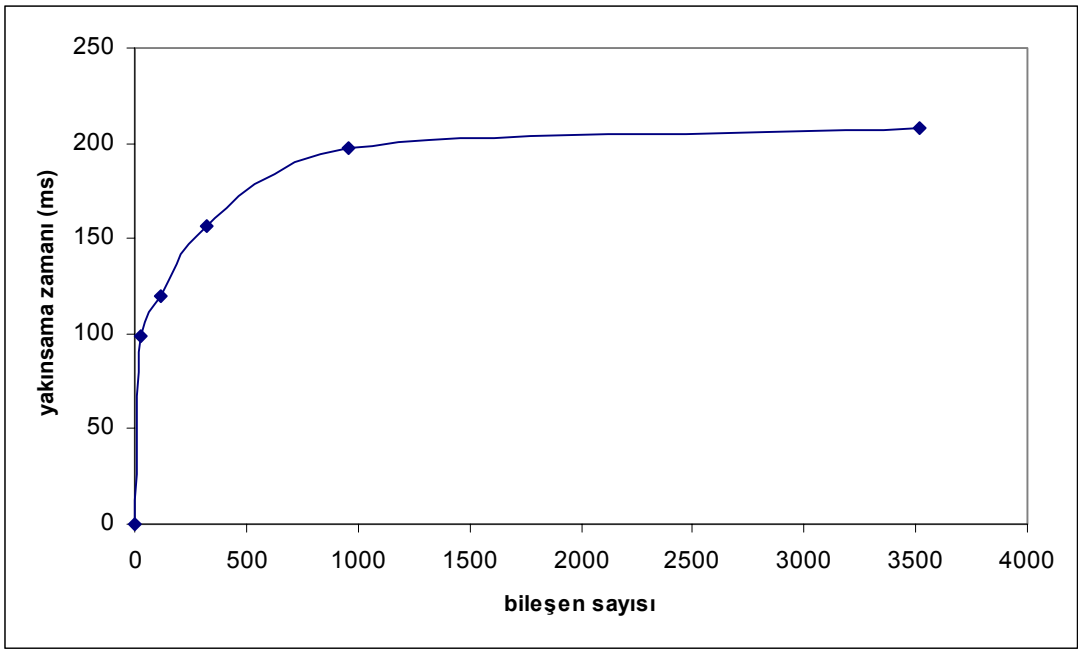
Şekil 6.11'deki grafikte, ağ çıkışı ve bileşen sayısı arasındaki ilişki görülmektedir. Grafikten, bileşen sayısı artarken ağ çıkışının kademeli olarak düştüğü görülmektedir. Ağ çıkışının düğüş göstermesinin temel sebebi; büyük ölçekli ağlarda oluşan trafik yüküne paralel olarak paket kaybının daha fazla olmasıdır. Ancak büyüyen ağ boyutlarında oluşan kayıplar ve performans düşümleri kabul edilebilir sınırlardadır.



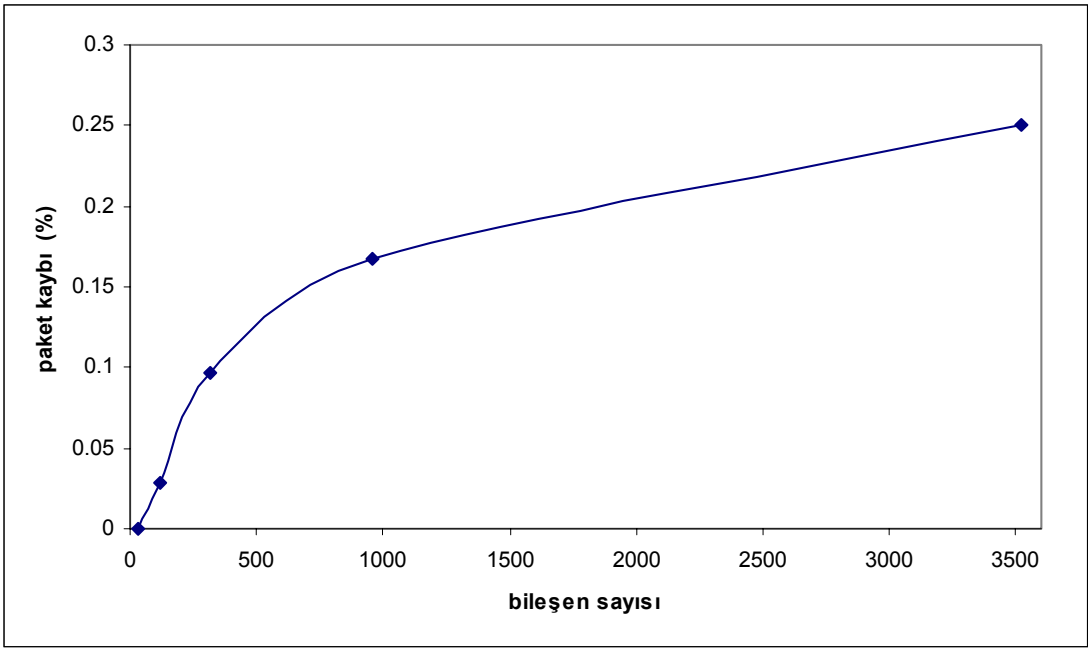
Şekil 6.11 Ağ çıkışının bileşen sayısıyla değişimi.

Şekil 6.12'de, değişik boyuttaki ağların yakınsama zamanları grafikle ifade edilmiştir. Bütün grafiklerde yakınsama zamanının milisaniyeler mertebesinde kalması, geliştirilen simülatör ve algoritmanın en önemli üstünlüklerindendir. Daha öncede ifade edildiği gibi, hızlı bir yakınsama işlemi ağ veritabanının oluşturulmasına ve güncellenmesine odaklanan algoritmaların en önemli hedeflerinden biridir.

Grafikten, yaklaşık 800 bileşene kadar yakınsama değerinin hızla arttığı, daha yüksek bileşen sayılarında ise yakınsama değerinin daha düşük artış eğilimi gösterdiği gözlemlenmektedir. Bunun iki adet sebepten kaynaklandığı tespit edilmiştir. Birincisi, büyük ölçekli ağlar oluşturulurken, bütün bileşenleri tek tek tanımlayıp birbirine bağlamak yerine yinelemeli bir algoritmanın kullanılmasıdır. Bu tür ağlar birbirine benzer ağların tekrarlanması sonucu meydana getirilmiştir. Ağlar arasındaki benzerlikten nedeniyle yönlendirme tabloları birbirinin aynısı oldukları için yönlendirme tablolarının oluşturulması ve güncellenmesi için geçen süre nispeten düşük kalmıştır.



Şekil 6.12 Ağların yakınsama değerlerinin bileşen sayılarıyla değişimi.



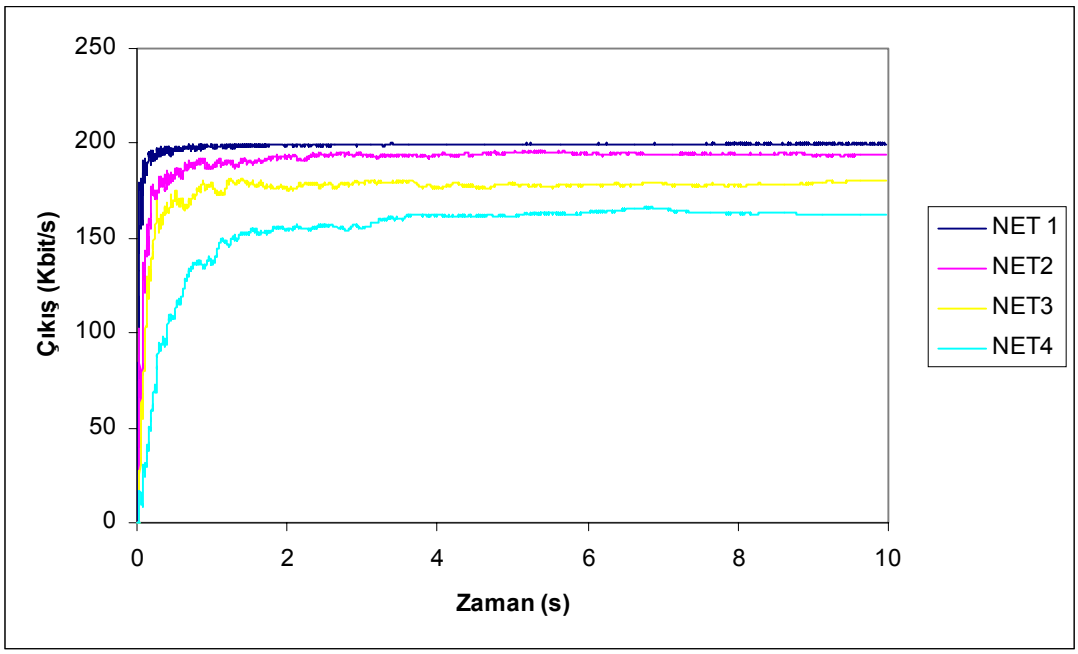
Şekil 6.13 Bileşen sayıları ve (%) paket kayıplarının değişimi.

İkinci sebep, DEVS metodolojisinin yüksek bir paralelizme sahip olmasıdır. DEVS bileşenleri paralel bir çalışma şekline sahip olduğu için bileşen sayısındaki artış toplam simülasyon zamanına yansımamaktadır.

Şekil 6.13'te verilen grafikte paket kaybı ve bileşen sayısı arasındaki orantı görülmektedir. Yaklaşık olarak 50 bileşenli bir ağda paket kaybı yaşanmamasına rağmen, 50'den fazla bileşene sahip ağlarda % 25'e kadar paket kaybı yaşanmaktadır.

Yukarıda grafikleri verilen deneyler ve test işlemleri 1saniyelik simülasyon süresine sahiptir. Daha büyük simülasyon sürelerinde geliştirilen ağ modellerinin davranışlarını test etmek için çeşitli deneyler yapıldı. Aşağıdaki grafiklerde 10 saniye boyunca değişik boyuttaki ağların çıkış ve gecikme grafikleri karşılaştırmalı olarak sunulmuştur.

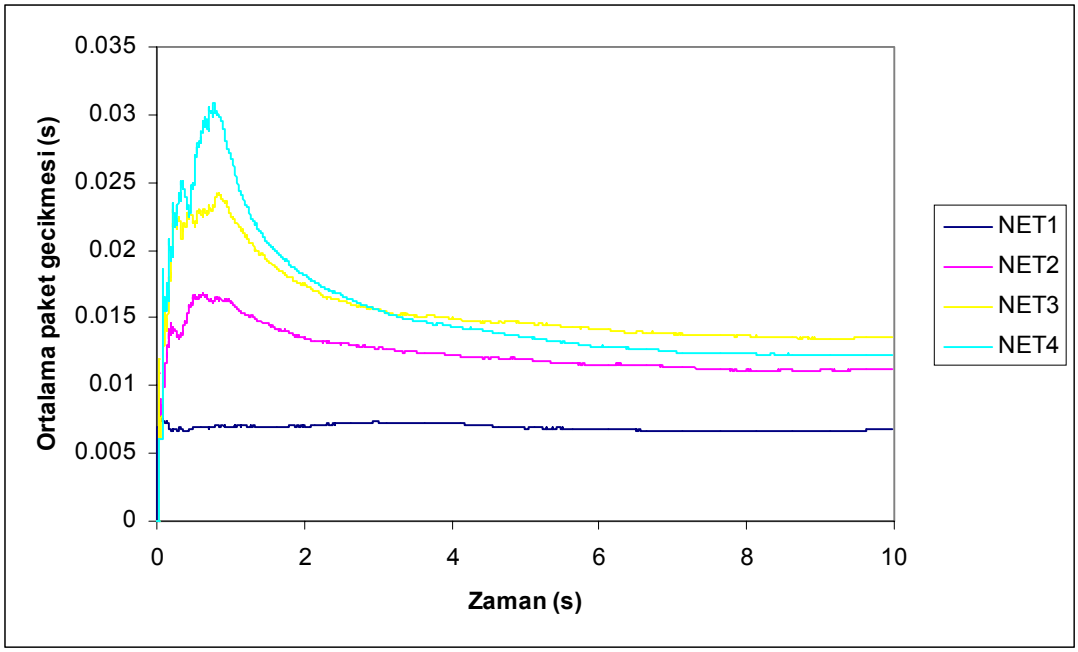
Şekil 6.14'te, Tablo 6.2'de listelenen ağların 10 saniye boyunca ağ çıkışlarının grafiği verilmektedir. Bütün ağlar, benzer bir davranış göstermekte ve belirli bir süre sonunda maksimum ağ çıkış değerini almaktadır. Daha sonra bu değer korunarak simülasyon tamamlanıncaya kadar sabit kalmaktadır. Bu durum bütün boyuttaki ağların yük dengelemeyi tutarlı bir şekilde gerçekleştirdiğini göstermektedir.



Şekil 6.14 Değişik boyuttaki ağların 10 sn boyunca çıkışları.

Şekil 6.15'te verilen grafikte, test ağları için ortalama paket gecikmesi değerleri verilmektedir. Ağ boyutları artarken paketlerin ağdaki seyahat süreleri nispeten arttığı için bileşen sayısı büyük ağlarda gecikme daha büyük olmaktadır.

Sonuç olarak, büyük ölçekli ağlar gibi oluşturulması ve simülasyonu zor uygulamalarda elde edilen sonuçlar *SwarmNet* ağ modelleme ve simülasyon ortamının üstünlüklerini göstermektedir. Donanım olarak gelişmiş bilgisayarlar kullanılarak daha büyük ağlar test edilebilir.



Şekil 6.15 Büyük ölçekli ağların değişik ortalama paket gecikme çıkışları.

BÖLÜM 7. SONUÇLAR VE DEĞERLENDİRME

Bu çalışmada dinamik ağ sistemlerine ve ağ sistemlerinin yönetimine odaklanan bir modelleme ve simülasyon uygulaması gerçekleştirilmiştir. Dağıtık sistemler, DEVS modelleme ve simülasyon metodolojisi kullanılarak yazılım ve donanım parçalarıyla birlikte modellenmiş, daha sonra özel uygulamalar gerçekleştirilerek geliştirilen modelin geçerlenmesi ve doğrulanması sağlanmıştır. Geliştirilen modeller Java

programlama dilinde kodlanmış ve ortaya çıkan uygulama programı *SwarmNet* olarak adlandırılmıştır. Geliştirilen modeller zeki bileşenler içeren dağıtık sistemler gibi karmaşık yapıları ve sistemleri kapsamaktadır. Özet olarak, gerçekleştirilen çalışma üç kademeden oluşmaktadır:

i- Dinamik ağ sistemlerine ve ağ sistemlerinin yönetimine odaklanan modelleme ve simülasyon uygulaması gerçekleştirilmiştir.

ii- Dağıtık sistemler yazılım ve dananım parçalarıyla modellenmiştir.

iii- Geliştirilen simülatör altında büyük ölçekli ağ modelleri oluşturulup, yönlendirme algoritmaları test edilmiştir.

DEVS nesneye yönelik modellemeye imkan tanıdığı için yeniden kullanılabilir parçalara sahip modeller geliştirilmiştir. Yeniden kullanılabilirliğin modellerden yeni modeller geliştirmeyi mümkün kılması nedeni ile, ağ gibi aynı modelin yüzlerce kez kullanıldığı uygulamalarda yeniden kullanılabilirlik özelliği son derece önemlidir. Kullanılan DEVS modelleme yaklaşımının modüler ve hiyerarşik modeller kurmaya elverişli yapısının sağladığı avantajlar, bilgisayar ağları gibi ayrık olaylı sistemlerin modellenmesinde sistem teorisi tabanlı bir metodoloji sağlamış ve uygun arabirimli hiyerarşik modüllere sahip sistemleri tasarlamayı olanaklı kılmıştır. DEVS'in nesneye yönelik yapısı; bir ağı oluşturan düğümlerin, linklerin, yazılım varlıklarının, deneysel çerçevelerin modüler bir yapıda tasarımını, yeniden kullanımını ve sistemler sisteminin oluşturulmasını olanaklı kılmaktadır. DEVS sağladığı semantikler ve ifade biçimleriyle yüksek dinamizme sahip bileşenlerin tanımlanmasını ve yazılım olarak gerçeklenmesini kolaylaştırmıştır. DEVS yaklaşımının matematiksel formatı durum değişkenlerinin değişimlerine ve sabit-parçalı bir formatta olan grafiklerin üretilmesine odaklandığı için simülasyon sonuçlarının yorumlanmasını kolaylaştırmıştır.

DEVS modelleme ve simülasyon yaklaşımının paralel çalışan sistemler için uyarlanmasıyla ortaya çıkan '*Paralel DEVS'*, modern hesaplama teorisinde çok önemli bir yere sahip olan paralelliğin gerçeklenmesini olanaklı kılması yanında, bilgisayarların sıralı çalışan mimarileri nedeniyle ortaya çıkan sınırlandırmaları

ortadan kaldırmıştır. Ayrıca bilgisayar ağları ve ekolojik sistemler gibi son derece dinamik ve paralel çalışan sistemlerin modellenmesi ve simülasyonunda Paralel DEVS son derece uygun bir yapı sunmaktadır.

DEVS yaklaşımının ve Java programlama dilinin sağladığı esneklik, değişen ortama adapte olabilen zeki bileşenlerin tasarımını kolaylaştırmıştır. Java programlama dilinin tamamen nesneye yönelik bir programlama dili olması nedeniyle DEVS metodolojisinin uygulanmasına oldukça elverişlidir. Java'nın esnek yapısı, zeki bileşenler içeren sistemlerin kodlanmasını kolaylaştırmıştır. Java'da gerçekleştirilen uygulamaların web tarayıcılar altında çalışabilmesi, oluşturulan işlemlerin İnternet üzerinden yayınlanabilmesini sağlamakta ve İnternet kullanılarak uygulamaların paylaşılması, uzaktan eğitim gibi günümüzün ileri teknolojileri için elverişli bir altyapı oluşturmaktadır.

DEVS birleşim altında kapalılık özelliği; düğüm ve link atomik modelleri gibi bileşenlerden oluşan ağların meydana getirilmesine, daha sonra bu ağların diğer ağlar içerisinde kullanılmasına ve hiyerarşik ağların kurulmasına olanak tanımaktadır. DEVS birleşik model kavramı ağlardaki kümeleme ve ağı yönetilebilir parçalara bölme yaklaşımlarının modellenmesini / uygulanmasını kolaylaştırmaktadır.

Bu çalışmada geliştirilen SwarmNet'i oluşturan bileşenlerin modelleri, yazılım, donanım ve iletişim sistemleri, aygıtları, parçaları, vb. gerçeklemede DEVS matematiksel yaklaşımının uygunluğu bir takım uygulamalarla desteklenmektedir. Yapılan ilk uygulama RIP yönlendirme algoritmasının modellenmesidir ve *SwarmNet* ortamının geçerlenmesi ve doğrulanmasına yöneliktir. Bölüm 6'da ayrıntılandırıldığı gibi, maksimum ağ çıkışına hızlı bir şekilde ulaşılması ve çıkışın çalışma boyunca sabit kalması, geliştirilen algoritmanın sağlıklı çalıştığının ve yük dengeleme yaptığının göstergesidir. Bununla birlikte, düğümlerin sınırlı sayıda link kullanılarak birbirine bağlı olmasına ve ağır trafik şartlarına rağmen RIP uygulamasında herhangi bir paket kaybı yaşanmamaktadır. Aşırı trafik artışlarına sistem hızlı cevap verdiğinden dolayı ağ çıkışında aşırı dalgalanmalar gözlenmemiştir.

SwarmNet modelleme ve simülasyon ortamı, biyolojik organizmaların davranışlarından türetilen yönlendirme algoritmaları ve ağ yönetimi yaklaşımlarının çalışılmasını, araştırılmasını, simülasyonunu ve diğer yöntemlerle karşılaştırılmasını olanaklı kılmaktadır. Bu ifadeyi / teoriyi destekleme amacıyla balarılarının davranışlarından kural tabanlı bir algoritma geliştirilmiş, modellenmiş ve elde edilen sonuçlar RIP algoritmasıyla karşılaştırılmıştır. Sosyal canlıların davranışlarında esinlenilerek geliştirilen algoritma, yaygın olarak kullanılan klasik bir algoritma olan RIP'e göre daha iyi performans göstermiştir. Önerilen algoritma, alternatif yolların keşfedilmesi ve kullanılması konusunda daha verimli ve daha hızlı olması yanında ölçeklenebilirlik konusunda da son derece esnektir. Kontrol paketlerinin boyutunun çok küçük olması nedeni ile ağa fazla yük getirmemektedir. Olasılıklı yönlendirme ağdaki değişimlere daha çabuk adapte olmuştur. Ayrıca herhangi bir hedef düğüm için alternatif yolların tanımlı olması nedeni ile linklerin kopması durumunda ağ trafiği etkilenmemektedir. Önerilen algoritma, detaylı işlemler ve formüllerden daha çok kural tabanlı olduğundan simülasyon oldukça hızlıdır.

Son uygulamada, ekolojik yaklaşımlar kullanılarak büyük ölçekli ağlar yönetilebilir kolonilere bölünmüştür. Böylece, geliştirilen çerçeve kullanılarak oldukça büyük boyutlardaki ağları modellemek mümkün olmuştur. Elde ettiğimiz sonuçlar, *SwarmNet* ortamının İnternet türü ağların modellenmesi ve simülasyonu için de uygun olduğunu göstermiştir.

Geliştirdiğimiz modelleme ve simülasyon ortamı, ölçeklenebilir, adaptif ve sağlam ağ uygulamalarının modellenmesi ve tasarımı için örnek bir çerçeve olmuştur. Teorik modelleme ve simülasyon bilgisinin pratiğe dökülerek uygulanması, ülkemizde son derece zayıf olan modelleme ve simülasyon biliminin uygulamalı olarak kavranmasını ve öğrenilmesini sağlamıştır. Bu çalışma DEVS metodolojisinin geniş bir uygulama alanına uygulanabilirliğini desteklemiştir.

BÖLÜM 8. TARTIŞMA VE ÖNERİLER

Yapılan çalışmada geliştirilen S*warmNet* modelleme ve simülasyon ortamı, bilgisayar ağlarının yönetimi ve karmaşık / geniş ağ uygulamalarıyla ilgili konuları çalışmaya uygundur. Bu çalışmada, geliştirilen ortamın değişik özelliklerini yansıtmak üzere üç adet uygulama gerçekleştirilmiştir:

i- RIP yönlendirme algoritmasının uygulanması,
ii- Bir oğul zekası yönlendirme algoritmasının geliştirilmesi ve uygulanması,
iii- Büyük ölçekli ağların davranışlarının incelenmesi.

Bu uygulamalar gerçeklenirken belirli bir soyutlama / basitleştirme yapılarak sistemler tasarlanmıştır. Tasarlanan bu sistemler üzerinde, DEVS metodolojisinin ve Java dilinin gelişmiş özellikleri kullanılarak detaylı / yeni uygulamalar geliştirilebilir. geliştirilebilecek uygulamalara örnek olarak aşağıda sunulan çalışmalar gerçekleştirilebilir:

Modellemede büyük ölçekli ağların kurulması ve topoloji üreteci: Büyük ölçekli ağların çalışılması, gerek donanım ve gerekse yazılım sınırlamaları nedeniyle bir problem oluşturmaktadır. Yazılım alanındaki sınırlamalar genellikle büyük boyutta sistemleri elle kurmanın mümkün olmadığı durumlarda ortaya çıkmaktadır. OPNET, COMNET, NS2, vb. bilgisayar ağı simülatörleri, birkaç yüz mertebesinde düğüme sahip ağları çalışmak için uygunken, binlerce düğüme sahip internet türü ağları çalışmak için uygun araçlar değildirler. Bu çalışmada *SwarmNet* ortamının binler mertebesinde düğümü (maksimum 3520) modelleyebildiğini gözlememize rağmen, bu boyutta ağların bileşenlerinin bağlantılarının elle tanımlanmasının ciddi bir problem oluşturduğunu gözlemlendi. Bu konuda yardımcı olması maksadıyla bir topoloji üretecinin geliştirilmesi kaçınılmazdır. Uygun bir algoritma kullanılarak mantıklı bağlantılara sahip istenen boyutta bir ağ otomatik olarak oluşturulabilir. Bölüm 5'te ayrıntılı şekilde açıklanan ve Bölüm 6'da uygulanan kümeleme veya diğer bir değişle ağı kolonilere ayırma yöntemi kullanılarak topolojiler belirli bir

algoritma kullanılarak üretilir. Düğüm sayısı, koloni sayısı, link ve düğümün oranlanmasıyla hesaplanan bağlantı oranı, vb. parametreler önceden verilerek, DEVSJAVA altında çalışabilecek büyüklükte ağlar kurulabilir. Topoloji üreteciyle birlikte SwarmNet büyük ölçekli ağları çalışmak için ideal bir ortam haline gelebilir.

Büyük ölçekli ağların simülasyonunda HLA teknolojisinden faydalanma: İnternet gibi büyük ölçekli ağları simüle etmek, sınırlı donanıma / işlemciye sahip bir masaüstü bilgisayarın kullanılmasıyla mümkün olmayabilir. Bir bilgisayarın sınırlarını aşan bir simülasyon işlemi, birden fazla bilgisayara bölünerek gerçeklenebilir. Bu gibi durumlarda, paralel ve dağıtık simülasyon teknolojileri kullanılmalıdır. Yüksek seviyeli yapı (High Level Architecture - HLA), simülatörleri paralel ve dağıtık çalıştırma konusunda gelişmiş ve uygun bir teknolojidir. HLA teknolojisi kullanılarak birkaç bilgisayarda bulunan büyük ölçekli bir ağ modeli, birkaç alt modele bölünerek HLA üzerinden bütünleştirilir ve böylece büyük modelleri simüle etmek mümkün olur.

Tam / eksiksiz işleve sahip bir ağ simülatörü: Bu çalışmada ağ modelleme ve simülasyon konusunda DEVS metodolojisi kullanılmıştır. Bu çalışmada ortaya konduğu gibi DEVS yaklaşımı, ayrık olaylı sistemleri modellemede büyük üstünlüklere sahiptir. Bu tezde, ağın belirgin bir çok özelliği modellenmiş olsa da, ağdaki birçok işlev ihmal edilerek modellenmemiştir. Ağın bütün yönleri ve içeriği göz önüne alınarak kapsamlı bir ağ simülatörü geliştirmek mümkündür. DEVSJAVA haricinde özel arabirimler tasarlanarak, eğitim ve geliştirme konularında araştırmacılara yardımcı olacak eksiksiz bir '*DEVSNET'* ağ simülatörü DEVS yöntemi kullanılarak geliştirilebilir.

Geliştirilen SwarmNet ortamında yapılan deneylerin OPNET, COMNET, vb. simülatörlerinde yapılarak sonuçların karşılaştırılması: Bu çalışmada, geliştirilen modellerin geçerlenmesi ve doğrulanması klasik bir algoritma aracılığıyla gerçekleştirildi. Geliştirilen simülatörü geçerlemek ve doğrulamak için yaygın olarak kullanılan simülatörlerden faydalanılabilir. Benzer ağlar benzer trafik şartlarında geliştirilen simülatör ve piyasada bulunan simülatörlerde test edilerek, sonuçlar karşılaştırılabilir. Yaygın olarak kullanılan ve doğrulukları tartışılmayan

simülatörlerle, geliştirilen simülatörün karşılaştırılmasından elde edilecek sonuçlar *SwarmNet* simülatörünün uygunluğunu / geçerliğini belirlemeye yardımcı olacaktır.

KAYNAKLAR

[1] A. S. Tanenbaum, *Computer Networks*, Prentice-Hall Inc., 1996.

[2] Zeigler, B.P., Mittal, S., Modeling and Simulation of Ultra-large Networks: A Framework for New Research Directions, supported by NSF Grant ANI-0135530, ULN Workshop, July 2002 (addendum to the ULN Workshop 2001) http://www.acims.arizona.edu/EVENTS/ULN/ULN_doc2.pdf

[3] Steenstrup, M. E. (Ed.). (1995). Routing in Communications Network. Prentice-Hall.

[4] E. Bonabeau, F. Henaux, S. Guérin, D. Snyers, P. Kuntz, G. Theraulaz, Routing in telecommunication networks with "Smart" ant-like agents, in: Proceedings of IATA'98, Second International Workshop on Intelligent Agents for Telecommunication Applications, Lectures Notes in Artificial Intelligence, Vol. 1437, Springer, Berlin, 1998.

[5] R Schoonderwoerd, O. Holland, J. Bruten, and L. Rothkrantz. Ant-based load balancing in telecommunication networks. Adaptive Behavior, 5(2):169{207, 1996.

[6] http://www.dei.isep.ipp.pt/docs/arpa.html, History of ARPANET

[7] D. Bertsekas and R. Gallager. *Data Networks*, Prentice-Hall, 1992.

[8] Network Simulator-2, http://www.isi.edu/nsnam/ns/

[9] Opnet Modeler, http://www.opnet.com/products/modeler/home.html

[10] Xiang Zeng, Rajive Bagrodia, Mario Gerla; "GloMoSim: a Library for Paralel Large-scale Wireless Networks", *Proceedings of the 12th Workshop on Paralel Simulations -- PADS '98*, May 26-29, 1998 in Banff, Alberta, Canada

[11] Parallel/Distribute NS, http://www.cc.gatech.edu/computing/compass/pdns/

[12] Zeigler B. P., Discrete event Abstraction: an emerging paradigm for modeling complex adaptive systems, Festschrift in honor of John H. Holland, L Brooker

[13] M. Heusse, S. Guerin, D. Snyers, and P. Kuntz. Adaptive agent-driven routing and load balancing in communication networks. *Advances in Complex Systems*, 1:234–257, 1998.

[14] Appleby, S. and Steward, S. Mobile software agents for control in telecommunications Networks. *BT Technology Journal, Vol. 12, No.2*. 1994.

[15] Bieszczad A., White T., Pagurek B., Mobile Agents for Network Management. In IEEE Communication Surveys, September 1998

[16] S. Lipperts and B. Kreller, "Mobile agents in telecommunications networks – a simulative approach to load balancing," *Proc. 5th Intl. Conf. on Information Systems, Analysis and Synthesis*, 1999.

[17] Panta Lucic and Dušan Teodorovic Transportation Modeling: An Artificial Life Approach Proceedings of the 14th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'02) 1082-3409/02 $17.00 © 2002 IEEE

[18] M. Wang and T. Suda, "The Bio-Networking Architecture: A Biologically Inspired Approach to the Design of Scalable, Adaptive, and Survivable/Available Network Applications," *Proc. of the 1st IEEE SAINT*, January 2001.

[19] E. Bonabeau, F. Henaux, S. Guerin, D. Snyers, P. Kuntz, and G. Theraulaz, "Routing in telecommunications networks with "smart" antlike agents", *Proc. Intelligent Agents for Telecommunications Applications '98*.

[20] G. Di Caro and M. Dorigo, "AntNet: a mobile agents approach to adaptive routing", Tech. Rep. IRIDIA/97-12, Université Libre de Bruxelles, Belgium

[21] Seeley TD (1995) The wisdom of the hive: the social physiology of honey bee colonies. Harvard University Press, Cambridge

[22] Bernard P. Zeigler, Herbert Praehofer, and Tag Gon Kim. Theory of Modelling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems. Academic Press, second edition, 2000.

[23] Bernard P. Zeigler. Theory of Modelling and Simulation. Robert E. Krieger, Malabar, Florida, 1984.

[24] Hans Vangheluwe. Multi-Formalism Modelling and Simulation, Doktora tezi, Universiteit Gent Faculteit Wetenschappen, 2001.

[25] Bernard P. Zeigler. Multifacetted Modelling and Discrete Event Simulation. Academic Press, London, 1984.

[26] Zengin, A., Ekiz, H., Firat, C., The High Level Architecture (HLA) For Distributed Simulations, 3rd International Symposium on Intelligent Manufacturing Systems, August 30 . 31, 2001, Sakarya

[27] Osman Balci. Principles of simulation model validation, verification, and testing. Transactions of the Society for Computer Simulation International, 14(1):3–12, March 1997. Special Issue: Principles of Simulation.

[28] Brian Magee. Popper. Fontana Press (An Imprint of HarperCollins Publishers), London, 1985.

[29] Bernard P. Zeigler. "A Theory-based Conceptual Terminology for M&S VV&A", Simulation Interoperability Workshop (SIW), 1999.

[30] George J. Klir. Architecture of Systems Problem Solving. Plenum Press, 1985.

[31] Zengin, A., Ekiz, H., Fırat, C., Köker, R., Paralel ve Dağıtık Ayrık Olaylı Simülasyonlar ve Karşılıklı Çalışabilirlik, ELECO 2002, 2002, Bursa

[32] Zengin, A., Ekiz, H., Dağıtık Simülasyon Uygulamaları ve Yüksek Seviyeli Yapı (HLA), Elektrik - Elektronik Bilgisayar Mühendisliği 9.Ulusal Kongre ve Sergisi, 19 - 23 Eylül 2001, Kocaeli

[33] B. P. Zeigler, Theory of Modelling and Simulation, John Wiley, New York, 1976.

[34] W.F. Clocksin and C.S. Mellish. Programming in Prolog. Springer Verlag, third edition, 1987.

[35] Tadao Murata, "Petri Nets: Properties, Analysis, and Applications", Proceedings of the IEEE, Vol. 77, No. 4, April 1989

[36] O'Hare, G. and Jennings, N., Eds., *Foundations of Distributed Artificial Intelligence,* John Wiley and Sons, 1996.

[37] Lesser, V. R., Multiagent Systems: An Emerging Subdiscipline of AI, ACM Computing Surveys, vol. 27, no. 3, ACM Press, Sept. 1995, pp. 340-342.

[38] Decker, K. S., Distributed Problem Solving: A Survey, *IEEE Trans. on Systems, Man, and Cybernetics,* vol. 17, no. 5, Sept. 1987, pp. 729-740.

[39] Georgeff, M. and Lansky, A., Reactive reasoning and planning. In Proceedings of the Sixth National Conf. on Artificial Intelligence (AAAI-87), pp. 677-682, Seattle, WA, 1987.

[40] Nwana, H. S., Software Agents: An Overview, Knowledge Engineering Review, vol. 11, no 3, Sept. 1996, pp.1-40.

[41] Green, S. et al., Software Agents: A review, Technical Report, Department of Computer Science, Trinity College, Dublin, Ireland.

[42] http://java.sun.com/products/embeddedjava

[43] Dijkstra, E.W. A Note on Two Problems in Connexion with Graphs *In Numerische Mathematik vol. 1. 1959.*

[44] C. Lynch, Packet Radio Networks, New York: Pergamon Press, 1987

[45] Beni G. and Wang J. Swarm intelligence in cellular robotics systems. *Proc. NATO Adv. Workshop on Robotics and Biological Systems*, 1989.

[46] White T., Pagurek B., and Oppacher, F., Connection Management using Adaptive Agents. In Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'98), July 12th-16th, 1998, pp. 802-809.

[47] M. Roth, S.Wicker, *Termite: Emergent Ad-Hoc Networking*, The Second Mediterranean Workshop on Ad-Hoc Networks, Medhia, Tunisia, 2003.

[48] I. Kassabalidis, M.A. El-Sharkawi, R.J. Marks II, P. Arabshahi, and A.A. Gray, "Swarm intelligence for routing in communication networks," *IEEE Globecom 2001*, Nov 25-29, 2001, San Antonio, Texas.

[49] Seeley TD (1985) Honeybee ecology: a study of adaptation in social life. Princeton University Press, Princeton

[50] Anderson C, Ratnieks, FLW, 1999. Worker allocation in insect societies: coordination of nectar foragers and nectar receivers in honey bee (*Apis mellifera*) colonies. Behav Ecol Sociobiol 46:73–81.

[51] Seeley TD, 1994. Honey bee foragers as sensory units of their colonies. Behav Ecol Sociobiol 34:51–62.

[52] Anderson, C. & Ratnieks, F. L. W. 1999. Task partitioning in foraging: general principles, efficiency and information reliability of queueing delays. In: *Information Processing in Social Insects* (Ed. by C. Detrain, J. L. Deneubourg & J. M. Pasteels), pp. 31–50. Basel: Birkhauser Verlag.

[53] Camazine, S. (1991). Self-organizing pattern formation on the combs of honey bee colonies. *Behav. Ecol. Sociobiol.* **28**: 61.76.

[54] Glomosim: http://pcl.cs.ucla.edu/projects/glomosim/

[55] SSFNet, http://www.ssfnet.org/

[56] Seeley TD, 1997. Honey bee colonies are group-level adaptive units. Am Nat 150:S22–S41.

[57] DEVS-DOC http://acims.eas.asu.edu/PUBLICATIONS/PDF/hild_phd.pdf

[58] Hild, D.R., H.S. Sarjoughian, B.P. Zeigler, "DEVS-DOC: A Co-Design Modeling and Simulation Environment." IEEE SMC-Part A, Vol. 32, No. 1, pp. 78-92.

[59] Unified Modeling Language, www.omg.org/uml/

[60] R. Y. Rubinstein and B. Melamid, *Modern Simulation and Modeling*, John Wiley & Sons Inc., 1998.

[61] Zengin A., Sarjoughian H., Ekiz H., "Biologically Inspired Discrete Event Network Modeling", ESS2004, Budapest / Hungary, 2004.

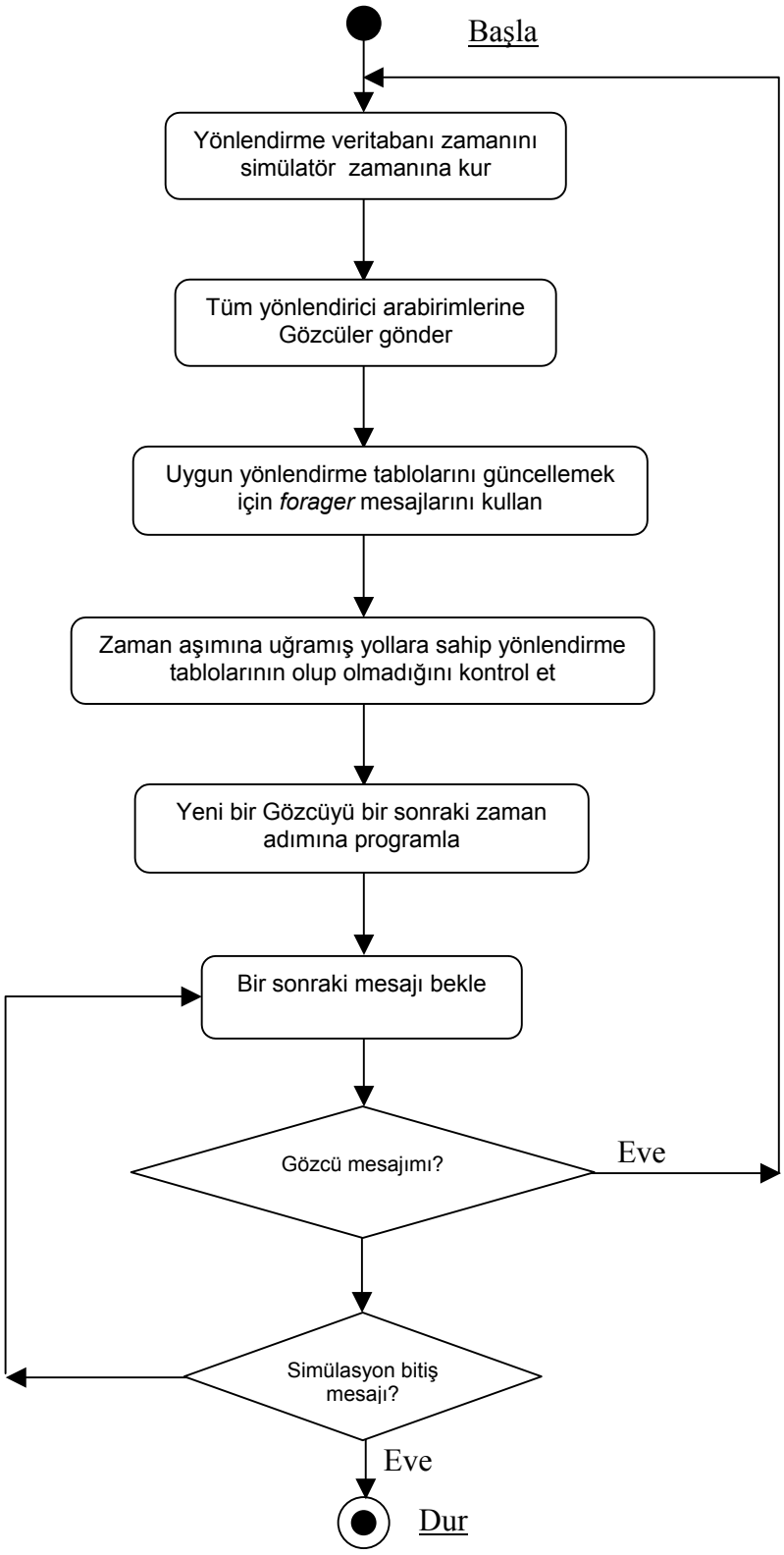
ÖZGEÇMİŞ

Ahmet Zengin 1975 yılında Sapanca'da doğdu. İlk ve orta öğrenimini Sapanca'da, lise tahsilini Sapanca Lisesinde tamamladı. 1993 yılında girdiği Sakarya Üniversitesi Mühendislik Fakültesi Elektrik ve Elektronik Bölümünden 1997 yılında mezun oldu. 1998 yılında SAÜ Teknik Eğitim Fakültesi Elektronik ve Bilgisayar Eğitimi Bölümü Bilgisayar Bilimleri Ana Bilim Dalına Araştırma Görevlisi olarak atandı.

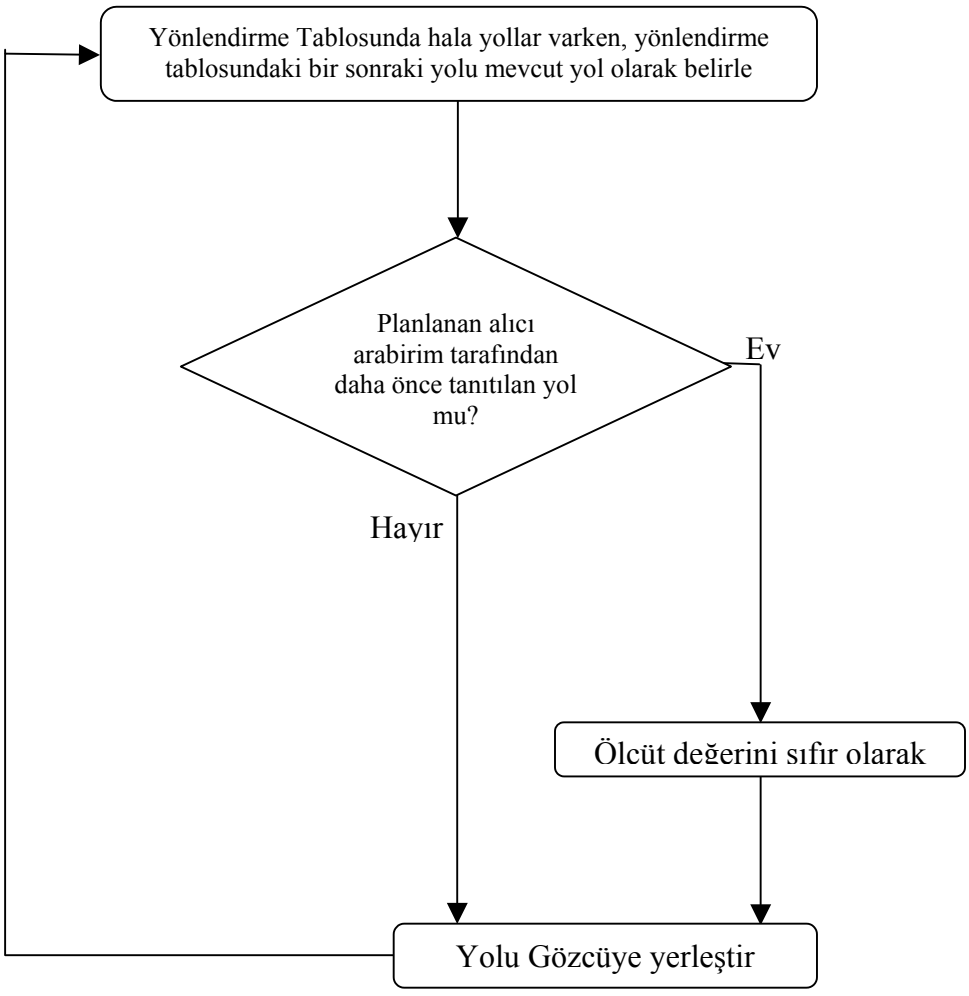
1997 başladığı "Simülasyonların Etkileşimli Olarak Çalışması ve HLA" isimli yüksek lisans tez çalışmasını 1999 yılında tamamladı ve yüksek mühendis ünvanını aldı. Aynı yıl Sakarya Üniversitesi Fen Bilimleri Enstitüsü Elektrik-Elektronik Mühendisliği Anabilim Dalında doktora eğitimine başladı. Mart 2003 yılından itibaren Amerika Birleşik Devletlerinde Arizona State Üniversitesi bünyesinde ACIMS modelleme ve simülasyon merkezinde bir yıl boyunca DEVS modelleme ve simülasyon formalizmi ile yapay zeka yardımıyla ağ kontrolü ve ağ modelleme üzerine çalışmalarda bulundu.

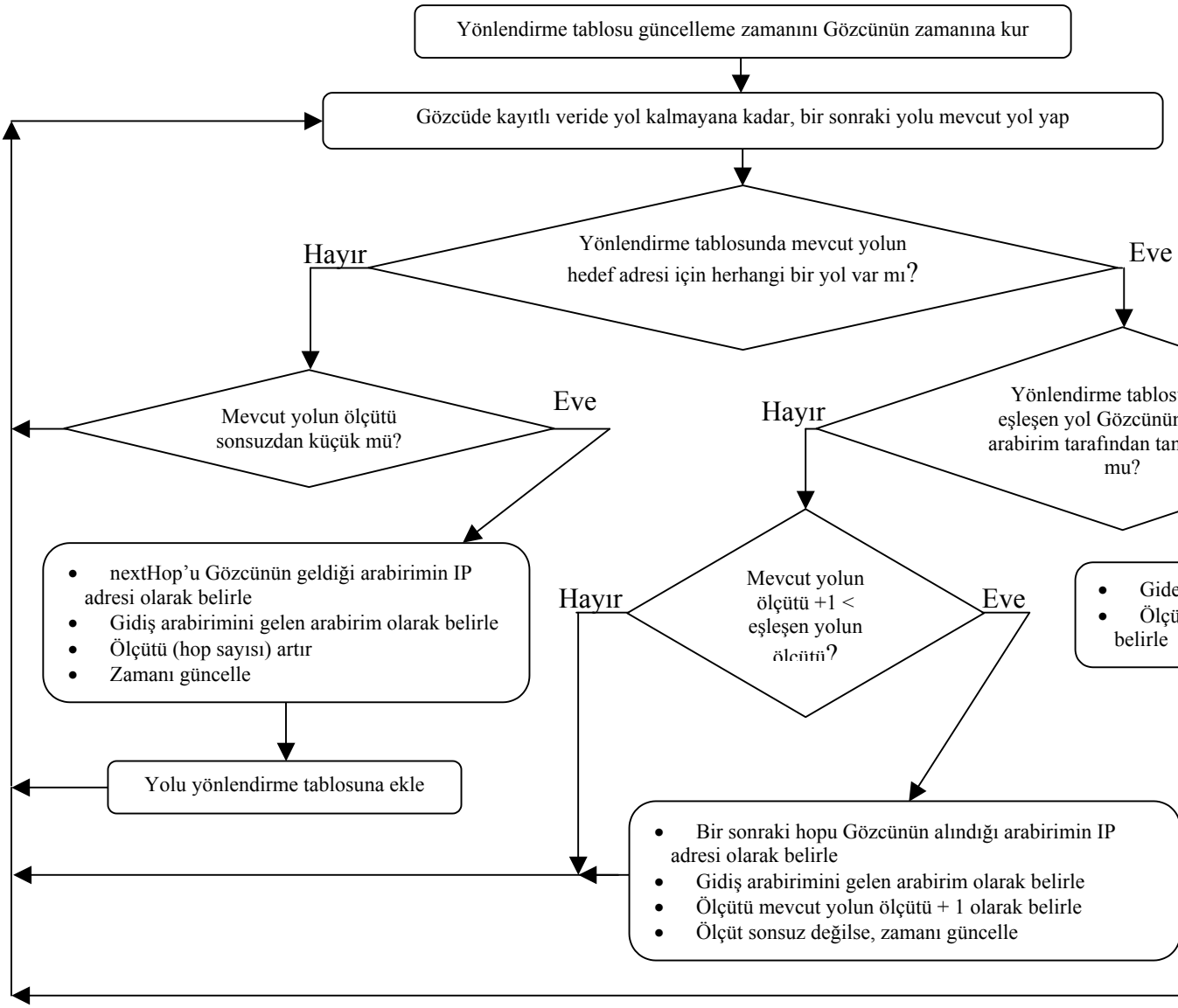
Ahmet Zengin halen Sakarya Üniversitesi Teknik Eğitim Fakültesinde görevine devam etmektedir.

EK A. KEŞFET-YÖNLENDİR ALGORİTMASI



EK B. GÖZCÜ'NÜN HANGİ YOLLARI TAŞIYACAĞINI BELİRLEYEN ALGORİTMA




Yönlendirme tablosu güncelleme zamanını Gözcünün zamanına kur
Gözcüde kayıtlı veride yol kalmayana kadar, bir sonraki yolu mevcut yol yap
Yönlendirme tablosunda mevcut yolun hedef adresi için herhangi bir yol var mı?
Hayır
Eve
Mevcut yolun ölçütü sonsuzdan küçük mü?
Eve
Hayır
Yönlendirme tablos
eşleşen yol Gözcünür
arabirim tarafından tar
mu?
nextHop'u Gözcünün geldiği arabirimin IP adresi olarak belirle
Gidiş arabirimini gelen arabirim olarak belirle
Ölçütü (hop sayısı) artır
Zamanı güncelle
Mevcut yolun ölçütü +1 < eşleşen yolun ölçütü?
Hayır
Eve
Gide
Ölçü
belirle
Yolu yönlendirme tablosuna ekle
Bir sonraki hopu Gözcünün alındığı arabirimin IP adresi olarak belirle
Gidiş arabirimini gelen arabirim olarak belirle
Ölçütü mevcut yolun ölçütü + 1 olarak belirle
Ölçüt sonsuz değilse, zamanı güncelle