ÇOK İŞLEMCİLİ AĞLARDA PARALEL AYRIK OLAY BENZETİMLERİNİN DAĞITIK İŞLETİM SİSTEMİ DONANIMI İÇİN ANAHTAR-TABANLI YAKLAŞIM

SWITCH-BASED APPROACH FOR DISTRIBUTED OPERATING SYSTEM HARDWARE OF PARALLEL DISCRETE EVENT SIMULATON IN MANY PROCESSOR NETWORK

OSMAN VOLKAN KARACA

PROF. DR. ALİ ZİYA ALKAR
Tez Danışmanı
DOÇ. DR. KAYHAN MUSTAFA İMRE
Yardımcı Tez Danışmanı

Hacettepe Üniversitesi
Lisansüstü Eğitim-Öğretim ve Sınav Yönetmeliğinin
Elektrik ve Elektronik Mühendisliği Anabilim Dalı için Öngördüğü
DOKTORA TEZİ olarak hazırlanmıştır.

2023

ÖZET

ÇOK İŞLEMCİLİ AĞLARDA PARALEL AYRIK OLAY BENZETİMLERİNİN DAĞITIK İŞLETİM SİSTEMİ DONANIMI İÇİN ANAHTAR-TABANLI YAKLAŞIM

Osman Volkan KARACA

Doktora, Elektrik ve Elektronik Mühendisliği Bölümü
Tez Danışmanı: Prof. Dr. Ali Ziya ALKAR
Eş Danışman: Doç. Dr. Kayhan Mustafa İMRE
Ocak 2023, 81 sayfa

Paralel ayrık olay benzetimlerinde, benzetim zamanın ilerletilmesi ve dağıtılması ile birlikte yayıncı ve abone düğümler arası veri iletiminin en düşük gecikme ile yapılması tüm benzetimin performansının artırılması için yüksek öneme sahiptir. Bu çalışmada ortaya konulan ağ hızlandırıcı yardımıyla paralel ayrık olay benzetimlerinin performansının artırılması hedeflenmiştir. Paralel ayrık olay benzetim altyapısı olarak yüksek düzeyli mimari (HLA) kullanılmıştır. Benzetim performansını artırmak için HLA'nın koşum zaman altyapısı (RTI) aracılığı ile sunduğu zaman yönetimi (TM) ve veri dağıtım yönetimi (DDM) servislerinin gecikmelerinin düşürülmesine çalışılmıştır. Bu servislerin gecikmelerini düşürmek için TM servisinde kullanılan mümkün olan en büyük lojik zaman (GALT) ve DDM servisinde kullanılan eşleştirme algoritmasının, anahtar-tabanlı yaklaşım ile uygulaması gerçekleştirilmiştir. Bu yaklaşımda düğümlerden ağ anahtarına taşınan algoritmalar ile paket yönlendirme ve güncelleme işlemleri ağ trafiğini azaltacak şekilde şişman ağaç ağ topolojisinde gerçekleştirilmiştir. Sonuçta bir dağıtık işletim sistemi olarak görev yapan RTI'nın sunduğu TM ve DDM servisleri çok işlemcili ağda anahtar donanımına aktarılmıştır. Literatüre çok işlemcili şişman ağaç ağlarda paralel ayrık olay benzetimlerinin dağıtık işletim sistemi donanımı için anahtar-tabanlı bir yaklaşım eklenmiştir.

Çalışma sonunda ortaya konulan akıllı anahtar (IS) ile konvansiyonel anahtar (CS), hesaplanan maliyet fonksiyonları ve paralel ayrık olay benzetici (PDES) üzerinde gerçekleştirilen benzetimler yardımıyla karşılaştırılmıştır. Bu karşılaştırmalarda düğüm sayısı 16, 54 ve 128 mesaj uzunluğu ise 64B ile1024B arasında taranmıştır. Ağda trafik olmadığı durumda hesaplanan maliyet fonksiyonlarında, GALT algoritmasında %72'ye ve eşleştirme algoritmasında %19'a varan bir performans iyileşmesi IS ile sağlanmıştır. Ölçeklenebilir IS ile GALT algoritmasında iyileşme ağ boyutu ile artmaktadır. PDES üzerinde yapılan GALT ve eşleştirme algoritmalarının bir arada kullanıldığı senaryo testlerinde ise farklı trafik yükleri altında %53'e varan bir performans iyileşmesi IS yardımıyla sağlanmıştır. Elde edilen iyileşmeler yayın/abone oranın düşük olduğu küçük parçalı uygulamalarda artmaktadır. Ayrıca GALT algoritmasının iyimser eş zamanlama uygulamalarında ortaya çıkan ağda kalan geçiş mesajları problemi IS yardımıyla çözülmektedir. Böylece benzetim zamanı geri almalar ortadan kalkmakta, dolayısıyla da benzetim performansı iyileştirilmektedir. Ağda GALT ve Eşleştirme algoritmalarını hesaplayan, geçiş mesajı kalmasına izin vermeyen ve ölçeklenebilir IS sayesinde küçük parçalı paralel ayrık olay benzetimlerinin performansı artırılmıştır.

Anahtar Kelimeler: HLA, Paralel Ayrık Olay Benzetimi, Dağıtık İşletim Sistemi, Ağ Hızlandırıcı, FPGA.

ABSTRACT

SWITCH-BASED APPROACH FOR DISTRIBUTED OPERATING SYSTEM HARDWARE OF PARALLEL DISCRETE EVENT SIMULATON IN MANY PROCESSOR NETWORK

Osman Volkan KARACA

Doctor of Philosophy, Department of Electrical and Electronics Engineering
Supervisor: Prof. Dr. Ali Ziya ALKAR
Co- Supervisor: Assoc. Prof. Kayhan Mustafa İMRE
January 2023, 81 pages

Time advancement and publish/subscribe message distribution with minimum latency has the utmost importance for the performance of parallel discrete event simulations. Parallel discrete event simulations performance improvement is aimed at the designed network accelerator. High Level Architecture (HLA) is used as a parallel discrete event simulation infrastructure. Time management (TM) and data distribution management (DDM) services are provided by run time infrastructure (RTI). The TM and DDM services' latency is improved to increase the simulation performance. TM service's greatest available logic time (GALT) and DDM service's matching algorithms' switch-based approach has been developed to improve these services' latency. GALT and matching algorithms are offloaded to the Ethernet switch with this approach. Packet forwarding and modification by reducing fat tree-based networks' traffic are implemented on the switch. So RTI, which behaves like distributed operating system, services are offloaded to the switch hardware in many processors network. This reveals distributed operating system hardware of parallel discrete event simulation for many processors on the fat tree network.

Calculated cost functions and parallel discrete event simulator (PDES) results of the intelligent switch (IS) are compared with conventional switch (CS). Node sizes of 16, 54, and 128 and message lengths between 64B to 1024B are swept during these comparisons. Calculated cost functions show that the GALT algorithm is improved by up to 72% and the matching algorithm is improved by up to 19% with IS under no traffic conditions. GALT algorithm improvement is increased with network size by scalable IS. PDES test reveals that scenarios using GALT and matching algorithms implemented on IS improves the simulation performance by up to 53% under different traffic loads. Performance improvements are increased for low publish/subscribe ratio likewise the fine-grained applications. At last but not least, the transient message problem revealed during GALT algorithm's optimistic synchronization implementation is solved with this study. Simulation performance is improved by removing the rollback operation. Scalable IS computes the GALT and matching algorithms in the network and also solves the transient message problem, helping to improve the performance of fine-grained parallel discrete event simulations.

Keywords: HLA, Parallel Discrete Event Simulation, Distributed Operating System, Network Accelerator, FPGA.

TEŞEKKÜR

Uzun yıllar boyunca benimle paylaştıkları bilgi ve tecrübeleri ile bu tez çalışmasının tamamlanmasında büyük emeği geçen yardımcı tez danışmanım Doç. Dr. Kayhan İMRE ve tez danışmanım Prof. Dr. Ali Ziya ALKAR'a,

Değerli yorumları ile bu tez çalışmasının yön bulmasında katkılarından dolayı tez izleme komitesi üyelerim Prof. Dr. Cüneyt BAZLAMAÇCI, Prof. Dr. Cenk TOKER'e ve Doç. Dr. Umut SEZEN'e,

Değerli görüş ve önerileri için savunma sınavı jüri üyelerim Prof. Dr. Ece GÜRAN SCHMIDT, Doç. Dr. Harun ARTUNER ve Doç. Dr. Dinçer GÖKCEN'e,

TÜBİTAK SAGE'ye ve çalışma arkadaşlarıma,

Bugünlere gelmemi sağlayan anneme ve babama,

Birlikte geçireceğimiz güzel zamanlardan yaptıkları fedakârlık için sevgili eşim ile kızlarım Arya ve Ayda'ya,

Teşekkürlerimi sunarım.

İÇİNDEKİLER

ŞEKİLLER DİZİNİ

ÇİZELGELER DİZİNİ

SİMGELER VE KISALTMALAR

Simgeler

L	Mesaj uzunluğu (Byte)
τ	Hat bant genişliği (Hertz)
t_{clk}	Saat periyodu (Saniye)

Kısaltmalar

ADC	Analog-Sayısal Dönüştürücü (Analog to Digital Converter)
AN	Otomatik Anlaşma (Auto- Negotiation)
ASIC	Uygulamaya Özel Tümleşik Devre (Application Specific Integrated Circuit)
AXIS	Gelişmiş Genişleyebilir Akış Arayüzü (Advanced eXtensible Interface Stream)
BTAR	Herkese-Yayın Tabanlı Zaman İlerleme İsteği (Broadcast-Based Time Advance Request)
CS	Konvansiyonel Anahtar (Conventional Switch)
DA	Hedef Adresi (Destination Address)
DDM	Veri Dağıtım Yönetimi (Data Distribution Management)
DIS	Dağıtık İnteraktif Benzetim (Distributed Interactive Simulation)
DMSO	Savunma Modelleme ve Benzetim Dairesi (Defense Modeling and Simulation Office)
DoD	Savunma Bakanlığı (Department of Defense)
DSD	Dinamik Senaryo Desteği
EVB	Deneme Kartı (Evaluation Board)
FCS	Çerçeve Kontrol Kelimesi (Frame Check Sum)
FIFO	İlk Giren İlk Çıkar (First In First Out)
FPGA	Alanda Programlanabilir Kapı Dizisi (Field Programmable Gate Array)
FTAR	Şişman Ağaç Tabanlı Zaman İlerleme İsteği (Fat Tree-based Time Advance Request)
GALT	En Büyük Uygun Mantıksal Zaman (Greatest Available Logic Time)

GMD	Geçiş Mesajı Desteği
GMII	Gigabit Ortam Bağımsız Arayüz (Gigabit Media Independent Interface)
GNSS	Küresel Konumlama ve Uydu Sistemi (Global Navigation and Satellite System)
GPU	Grafik İşlemci Ünitesi (Graphics Processing Unit)
GVT	Global Sanal Zaman (Global Virtual Time)
HBM	Yüksek Bant Genişlikli Hafıza (High Bandwidth Memory)
HLA	Yüksek Düzeyli Mimari (High Level Architecture)
ICT	Bilgi ve Haberleşme Teknolojisi (Information and Communication Technology)
IGMP	İnternet Grup Yönetim Protokolü (Internet Group Management Protocol)
IP	Fikri Mülkiyet (Intellectual Property)
IS	Akıllı Anahtar (Intelligent Switch)
LBTS	Alt Sınır Zaman İzi (Lower Bound Time Stamp)
LRC	Yerel RTI Bileşeni (Local RTI Component)
LRU	Hatta Değiştirilebilir Cihaz (Line Replaceable Unit)
MAC	Ortam Erişim Kontrolcüsü (Media Access Controller)
MII	Ortam Bağımsız Arayüz (Media-Independent Interface)
OMT	Obje Model Şablonu (Object Model Template)
PCS	Fiziksel Kodlama Alt Katmanı (Physical Coding Sublayer)
PDES	Paralel Ayrık Olay Benzetici (Parallel Discrete Event Simulator)
PE	İşlem Yapan Düğüm (Processing Element)
PHY	Fiziksel Seviye Gönderici-Alıcı (Physical Layer Transceiver)
PLL	Faz Kitlenmeli Döngü (Phase Locked Loop)
PMA	Fiziksel Ortam Bağlantısı (Physical Medium Attachment)
RS	Yönlendirme Uzayı (Routing Space)
RTI	Koşum Zaman Altyapısı (Run Time Infrastructure)
SA	Kaynak Adresi (Source Address)
SITL	Döngüde Sistem (System in The Loop)
SNI	Seri Ağ Arayüzü (Serial Network Interface)
SOCA	Obje Sınıf Özelliği Aboneliği

	(Subscribe Object Class Attributes)
SoC	Yonga Üzeri Sistem (System On-Chip)
TAG	Zaman İlerleme Onayı (Time Advance Grant)
TAR	Zaman İlerleme İsteği (Time Advance Request)
TDA	TM/DDM Hızlandırıcı (TM/DDM Accelerator)
TEMAC	Üç Kipli Ethernet MAC (Tri Mode Ethernet MAC)
TM	Zaman Yönetimi (Time Management)
TSO	Zaman İzi Sıralı (Time Stamp Order)
TW	Zaman Atlama (Time Warp)
UAV	Özelik Değeri Güncelleme (Update Attribute Value)
VHDL	VHSIC Donanım Tanımlama Dili
	(Hardware Description Language)
VHSIC	Çok Yüksek Hızlı Tümleşik Devre
	(Very High Speed Integrated Circuit)
VLAN	Sanal Yerel Ağ (Virtual Local Area Network)

1. GİRİŞ

Paralel ayrık olay benzetimlerinde, benzetimin ana kısmı adımlama olarak adlandırılmaktadır. Bu kısımda pozisyon, hız ve kuvvet hesapları gibi benzetim algoritmaları çalıştırılmaktadır. Adımlama kısmında benzetimin ihtiyaç duyduğu veriler ağdan alınır ve diğer benzetimlerin ihtiyaç duyduğu veriler ağa gönderilir. Benzetim zamanının ilerletilmesi ve onaylanmasının beklenmesi gibi ağda trafik oluşturan ve gerçekleşme süreleri ağ trafiğine bağlı bulunan işlemler adımla kısmında gerçekleşir [1].

Yüksek düzeyli mimaride (İng. High Level Architecture, HLA) benzetimin adımlama kısmında gerçekleşen ve ağ trafiğine bağlı olan işlemler, dağıtık işletim sistemi gibi görev yapan koşum zaman altyapısının (İng. Run Time Infrastructure, RTI) zaman yönetimi (İng. Time Management, TM) ve veri dağıtım yönetimi (İng. Data Distribution Management, DDM) servisleridir. Bu çalışmada ağ trafiğine bağlı bu servisleri şişman ağaç ağ topolojisi için tasarlanan akıllı Ethernet anahtarına (İng. Intelligent Switch, IS) aktararak, ayrık olay benzetimlerinin performansının artırılması hedeflenmiştir.

Tutucu ve iyimser eş zamanlama (İng. Conservative and optimistic synchronization) yöntemlerinde kullanılan en büyük uygun mantıksal zaman (İng. Greatest Available Logic Time, GALT) algoritmasının iyimser eş zamanlama yönteminde çalışması sırasında ortaya çıkan geçiş mesajları (İng. Transient message) ilerlenen zamanın geri alınmasına (İng. Rollback) neden olmaktadır. Bu çalışmada tasarlanan IS donanımı yardımı ile ağda kalan geçiş mesajı problemine çözüm bulunması da hedeflenmiştir. Ağda kalan geçiş mesajlarının ortadan kalkması ise dolaylı olarak paralel ayrık olay benzetimlerinin performansını artıracaktır.

İşlem yapan düğümlerde (İng. Processing Element, PE) RTI yardımıyla gerçekleştirilen TM servisinin GALT ve DDM servisinin eşleştirme algoritmaları konvansiyonel Ethernet anahtarına (İng. Conventional Switch, CS) aktarılmıştır. Böylece bu algoritmaların anahtar-tabanlı yaklaşımının uygulandığı IS gerçekleştirilmiş ve PE'lerin işlem yükleri azaltılmıştır. Sonuç olarak; uygulama parçacıkları arasında kaymanın (İng. Jitter) azaltıldığı, zaman tahmin edilebilirliğinin (İng. Timing predictability) artırıldığı, içerikte yapılan atamaların hızlandırıldığı (İng. Context switch), yazılımdaki aşırı yüklenmenin

(İng. Software overhead), olaylara (İng. Event) cevap verme süresinin ve kod için harcanan yerin azaltıldığı daha esnek RTI uygulamalarının önü açılmaktadır. Herhangi bir benzetim sisteminin performansını artırmak amacıyla işlemci çekirdek sayısı artırma [2] ya da sanal ortak hafıza (İng. Virtual shared memory) mimarisinde iyileştirme yapmak [3] da tercih edilebilecek yöntemler arasındadır. Fakat bu tez çalışmasında temel olarak TM ve DDM servislerinin performansını iyileştirerek paralel ayrık olay benzetimlerin performansının artırılması yani maliyet fonksiyonlarındaki hesaplamanın yanında haberleşme gecikmesinin de azaltılması hedeflenmiştir.

1.1 Değerlendirme Metrikleri

Bu bölümde literatüre yapılan katkının daha iyi anlaşılabilmesi için paralel ayrık olay benzetimlerinin performans değerlendirme metrikleri ortaya konmuştur. Paralel ayrık olay benzetim sistemlerinde benzetime katılan PE'lerin sayısı yüzler hatta binler seviyesinde olabileceği için en önemli performans metriklerinden bir tanesi ölçeklenebilirliktir (İng. Scalability). Ölçeklenebilirlik, benzetime katılan PE sayısı artışı ile benzetim performansının düşüşünü ilişkilendirmektedir. Tamamen ölçeklendirilebilir bir benzetimde, benzetime katılan düğümlerin haberleşme yükü nedeniyle gecikme süresinde bir artış yaşanmaz. Bir diğer performans metriği gecikmedir (İng. Latency). Gecikme birimi saniye olarak verilir. Performans karşılaştırmalarında ilgili algoritmayı en düşük gecikme ile çalıştıran yaklaşım en başarılı olarak değerlendirilmektedir. Değerlendirme metriklerinden maliyet eklenen donanımın toplam donanıma göre büyüklüğü, dolayasıyla eklenen lojik devrenin kapı maliyeti olarak ele alınmıştır. PE işlem yükü metriği ise ilgili algoritmanın benzetimde kullanılan PE'ye getirdiği yükün varlığı ve yokluğu olarak değerlendirilmektedir. Dinamik senaryo desteği (DSD) metriği abone mesajlarındaki sık bölge değişimi sonucunda artan eşleştirme algoritması gecikmesinin iyileştirilmesine yönelik bir çözümün varlığını belirtmek için verilmiştir. Eşleştirme algoritmasının karşılaştırılmasında kullanılan bir metriktir. Geçiş mesajı desteği (GMD) metriği ise GALT algoritmasında kullanılan bir metriktir. Geçiş mesajları için bir donanımsal çözümün varlığını belirtmek için kullanılmıştır.

Benzetim sisteminde Ethernet anahtarı olarak CS bulunduğu durumda GALT algoritması için; Bölüm 4.2'de verilen herkese yayın-tabanlı ve Bölüm 4.3'de verilen şişman ağaç-tabanlı yaklaşımlar kullanılmaktadır. Ethernet anahtarı olarak IS bulunduğu durumda ise

Bölüm 4.1'de verilen anahtar-tabanlı yaklaşım kullanılmaktadır. GALT algoritmasının değerlendirmesinde bu yaklaşımlar kullanılmıştır. Eşleştirme algoritmasında ise CS'de kullanılan bölge-tabanlı yöntem (Bölüm 4.5) ile IS'de kullanılan anahtar-tabanlı yöntem (Bölüm 4.4) karşılaştırılmıştır. Ölçeklenebilirlik karşılaştırmasında Bölüm 4'de GALT algoritmasının farklı yaklaşımları için verilen maliyet fonksiyonu analizleri kullanılmıştır. Gecikme değeri karşılaştırmasında GALT ve eşleştirme algoritması için maliyet fonksiyonları ve [4] çalışması ile elde edilen sonuçlar kullanılmıştır. Maliyet karşılaştırmasında, Bölüm 5.3'de verilen IS'ye ait kaynak tüketimleri kullanılmıştır. Eklenen donanımın geri kalan donanıma yani CS donanımına oranı incelenmiştir. PE işlem yükü, DSD ve GMD metrikleri ise CS ve IS'nin tasarımlarının farklarından kaynaklanmaktadır. CS ve IS tasarımları incelenerek karşılaştırılmaktadırlar.

1.2 Literatüre Yapılan Katkılar

Bu çalışma ile literatüre paralele ayrık olay benzetimlerinde kullanılan GALT ve eşleştirme algoritmalarının ağ hızlandırıcı ile uygulandığı, ağda geçiş mesajı kalmasına izin vermeyen anahtar-tabanlı bir yaklaşım eklenmiştir. Anahtar-tabanlı yaklaşım ile çalışan GALT algoritması Bölüm 4.1'de hesaplanan maliyet fonksiyonlarında gösterildiği üzere herkese yayın-tabanlı yaklaşımda olduğu gibi düğüm sayısına veya şişman ağaç-tabanlı yaklaşımda olduğu gibi anahtar kapı sayısına bağlı değildir. Şişman ağaç-tabanlı yaklaşım k değerine bağlı olduğundan düğüm sayısı 16'dan 128'e çıktığında gecikme %10 civarında artmaktadır. Herkese yayın-tabanlı yaklaşımda bu değişim %100'ü geçmektedir. Bölüm 6.1'de yapılan karşılaştırmalara göre ağda trafik olmayan durumda GALT algoritmasında 128 düğüm ve 64B mesaj için maliyet fonksiyonlarında %76 iyileşme elde edilmiştir. Yine Bölüm 6.1'de verilen maliyet fonksiyonlarında gösterildiği üzere ağda trafik yok iken ortaya konulan anahtar-tabanlı eşleştirme algoritması yaklaşımı, bölge-tabanlı eşleştirme algoritması yaklaşımına göre yaklaşık %19'luk bir gecikme iyileşmesine sahiptir. Son olarak Bölüm 6.2'de verilen senaryo testleri ile GALT ve eşleştirme algoritmaları bir arada kullanıldığında, yayın/abone oranı düşük iken küçük parçalı (İng. Fine-grained) uygulamalarda %53'e varan bir iyileşme farklı ağ trafik yükleri için gözlenmiştir. Bölüm 5.3'deki kaynak tüketimi sonuçlarına göre GALT algoritması için kullanılan kaynağın geri kalan CS kaynağına oranı %10 olarak bulunmuştur. Maliyet karşılaştırmalarında bu sonuç kullanılmıştır. PE işlem yükü anahtar-tabanlı yaklaşımda yok iken diğer yaklaşımlarda vardır. PE yükü GALT ve

eşleştirme algoritmalarının düğümlerde çalışması sonucu ortaya çıkmaktadır. DSD eşleştirme algoritmasında dinamik senaryolar için yapılan iyileştirmeyi gösterir. Anahtar-tabanlı yaklaşımda abone mesajları PE'lere çıkarılmadığı için dinamik senaryolarda bir iyileşme sağlanmaktadır. GMD, GALT algoritmasının iyimser eş zamanlama uygulamalarında oluşan geçiş mesajları için bir çözümün varlığını işaret eder. Anahtar-tabanlı yaklaşımda geçiş mesajları anahtar mimarisinde yapılan değişiklik ile ortadan kaldırılmıştır. Dolayısıyla GMD anahtar-tabanlı yaklaşımda var iken diğer yaklaşımlarda yoktur. Aşağıda bu bilgiler ışığında GALT ve eşleştirme algoritmaları için sırası ile Çizelge 1.1 ve Çizelge 1.2'de karşılaştırmalar yapılmıştır. En başarılı durum 5 adet '+' sembolü ile gösterilmiştir. Başarı oranı daha düşük çözümler en başarılı durum ile yukardaki veriler yardımıyla göreceli olarak değerlendirilmiştir.

Çizelge 1.1. GALT Algoritması için Herkese Yayın, Şişman Ağaç ve Anahtar-Tabanlı Yaklaşımların Değerlendirilmesi.

Yaklaşım	Ölçeklenebilirlik	Gecikme	Maliyet	PE Yükü	GMD
Herkese Yayın	+	+	+++++	Var	Yok
Şişman Ağaç	++++	++	+++++	Var	Yok
Anahtar	+++++	+++++	++++	Yok	Var

Çizelge 1.2. Eşleştirme Algoritması için Bölge ve Anahtar-Tabanlı Yaklaşımların Değerlendirilmesi.

Yaklaşım	Gecikme	PE Yükü	DSD
Bölge	++++	Var	Yok
Anahtar	+++++	Yok	Var

Bu çalışma sonunda ortaya konulan IS paralel arık olay benzetimlerinin küçük parçalı uygulamaları için şişman ağaç ağ topolojisinde CS'ye göre; daha iyi ölçeklenebilmekte, daha düşük gecikme ile çalışmakta, PE'ler üzerindeki işlem yükünü azaltmakta ve dinamik senaryo ile geçiş mesajı desteği sağlamaktadır. Bu iyileşmeler karşısında anahtarda %10'luk bir maliyet artışı ortaya çıkmaktadır.

1.3 Tezin Yapısı

Bölüm 2'de HLA temelleri ve HLA ile ilgili yapılan referans çalışmalara değinilmiştir. Bölüm 3'de GALT ve eşleştirme algoritmalarının anahtar üzerinde çalışması anlatılmıştır. Bölüm 4'de GALT ve eşleştirme algoritmalarının maliyet fonksiyonları farklı yaklaşımlar için ağda trafik olmadığı durumda incelenmiştir. Bölüm 5'de GALT ve eşleştirme algoritmalarının performansını artırmak için ortaya konulan IS ve test sisteminin detay tasarımı verilmiştir. Bölüm 6'da IS ve CS için PDES sonuçları ile maliyet fonksiyonları ağda trafik olmadığı durumda karşılaştırılmıştır. Ayrıca ağda trafik olduğu durumda IS ve CS'nin performansını değerlendirmek için yapılan çeşitli senaryo testlerinin sonuçları paylaşılmıştır. Son olarak Bölüm 7'de elde edilen sonuçlar değerlendirilmiş ve yorumlar yapılmıştır.

2. YÜKSEK DÜZEYLİ MİMARİ

Amerikan Savunma Bakanlığının (İng. Department of Defense, DoD) modelleme ve benzetim ana planı altı ana bölümden oluşur. Birinci bölüm modelleme ve benzetim için genel teknik bir uygulama çatısı geliştirilmesidir. HLA birinci bölümün altındaki bileşenlerdendir. HLA, modelleme ve benzetim topluluğunda standartlaşmayı teşvik etmek için tasarlanmıştır. Bu standartlaşma kapsamında DoD, HLA kullanımını zorunlu tutmaktadır [5].

HLA, birden çok bağımsız benzetimin dağıtılmış bir ortamda tek benzetim olarak çalışmasını sağlayan yazılım mimarisidir. IEEE 1516 ile standartlaşmıştır. Tekrar kullanılabilirliği ve birlikte çalışabilirliği desteklemektedir. Tekrar kullanılabilirlik sayesinde yeni geliştirilen bir uçağa ait benzetimdeki değişmeyen alt bileşenler, yeni uçak benzetiminde tekrar kullanılabilmektedir. Böylece benzetim maliyetleri düşürülebilmektedir. Birlikte çalışabilirlik sayesinde ise farklı programlama dilleri ve farklı işletim sistemlerinde geliştirilen benzetimler bir arada çalışabilmektedir [6].

Ayrık olay benzetimi, bir sistemin çalışmasını ayrık sıralı olaylar olarak modeller. Her olay belirli bir zamanda olur ve sistemin durum değişikliğine neden olur. Paralel (koşut) ayrık olay benzetimi ayrık olay benzetiminde, benzetimlerin çok işlemcili düğümlerde çalıştırılmasıdır. Dağıtılmış benzetim birbirlerine ağ üzerinden bağlanmış, fiziksel konumları dağıtılmış bilgisayarlar üzerinde benzetimlerin çalıştırılmasıdır [7].

HLA'da federe, HLA uyumlu benzetimi temsil etmektedir. RTI üzerinden bir araya gelen federeler ise federasyonu oluşturmaktadır. RTI aslında benzetim için çalışan dağıtık bir işletim sitemi gibi görev yapmaktadır. Federeler arası işlemler RTI üzerinden gerçekleştirilir. Federeler arasında gönderilen birbiri ile ilişkili veri topluluğuna obje denilir. Objedeki veri alanlarına ise özellik denilir. Federeler arasında gönderilen olaylara etkileşim denilir. Etkileşimdeki veri alanlarına ise parametre olarak adlandırılır. Obje sınıfları süreklilik içerirken etkileşimler içermez.

HLA temel olarak 3 tanımdan oluşmaktadır. Bunlar HLA obje model şablonları (İng. Object Model Template, OMT), HLA kuralları ve HLA arayüz tanımlarıdır. HLA OMT, HLA'nın benzetim ve federasyon objeleri için genel bir sunum formatı sağlar. Her bir

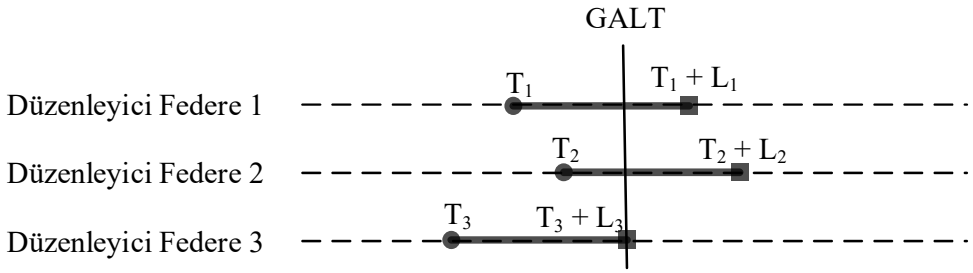
federe ve federasyon standart OMT kullanarak kendi obje modeli şablonunu oluşturur. Bu şablonda paylaşılan obje, özelik ve etkileşim setleri tanımlanır. Bu şablonlar yardımıyla bir federenin bir federasyonun parçası olma durumu daha kolay değerlendirilir [8]. HLA kuralları, federasyondaki bileşenlerin etkileşimlerinin doğru bir biçimde gerçekleştirilebilmesi için arlarındaki sorumlulukları ve ilişkileri belirtmektedir. HLA arayüz tanımları, federelerin RTI üzerinden federasyonla ve birbirleriyle hangi servisleri kullanarak etkileşime gireceklerini tanımlar [9].

HLA arayüz tanımları altı sınıfa ayrılmaktadır. Federasyon yönetim servisi; federasyonun yaratılması, katılma, ayrılma ve yönetiminden sorumludur. Deklarasyon yönetim servisi federelerin yayın yapacakları ve abone olacakları obje/etkileşim sınıflarını tanımlar. Obje yönetim servisi; federelerin obje/etkileşim sınıflarını nasıl kayıt ettireceğini, keşfedeceğini, özelik güncellemesini nasıl oluşturacağını, güncellemelerin ve etkileşimlerin nasıl alınacağını tanımlar. Sahiplik yönetim servisi federasyonun çalışması esnasında obje sahipliğinin dinamik transferini sağlar. TM servisi zaman yönetim politikaları ve zaman ilerletmede anlaşmayı uygulamak için gerekli mekanizmalardan sorumludur. DDM servisi yayıncı verilerin abonelere en verimli şekilde iletilmesini sağlar [10]. Bu servislerden TM ve DDM bu tez çalışmasında iyileştirme yapılan servislerdir.

TM, farklı işlemci hızlarına sahip bilgisayarlar üzerinde kosan dağıtılmış benzetim bileşenlerinin ortak bir zaman kavramı oluşturmasını sağlar. TM her bir federenin, federasyon zaman ekseninde ilerleme mekanizması ile ilgilenir. Bir federenin zaman ilerleyişi diğer bir federenin zaman ilerleyişine bağlı tutulabilir. Zaman ilerlemesini kontrol eden federeye düzenleyici, zaman ilerlemesi zaman düzenleyici federeye bağlı olan federeye ise kısıtlı federe denilir. Zaman düzenleyici federeler bir obje örneğinin özelik değerini güncelleme veya etkileşim parametresi gönderme gibi bir takım aktivitelerini kendi federasyon zamanları ile ilişkilendirebilirler. Bu gibi olayların (aktivitelerin) zaman etiketleri vardır ve zaman izi sıralı (İng. Time Stamp Order, TSO) olarak alınırlar. Federenin yerel zamanına mantıksal zaman denir. Zaman düzenleyen federe etkinlikleri yerel zamanı ve ileri bakma değeri toplamından büyük olacak şekilde TSO olarak gönderirken, zaman kısıtlı federe bu TSO iletilerini alıp sıralı olarak kullanır. Ayrıca kısıtlanmış federelerin kendileri ile ilişkilendirilmiş alt sınır zaman izleri (İng.

Lower Bound Time Stamp, LBTS) vardır. LBTS, federenin alabileceği mümkün olan en erken TSO etkinliğin zamanını belirtir. Düzenleyici federe zamanı ilerledikçe, kısıtlanmış federenin LBTS değeri de artar. Zaman ilerletildikten sonra gelen mesajlara ait zaman mevcut zamandan geride ise geçiş mesajı oluşmuş olur ve zaman geri alma işlemi gerçekleştirilir [11].

Düzenleyici federenin, mantıksal zaman ve ileri bakma değerinin toplamını verip zaman ilerleme isteğinde bulunması durumunda ağa zaman ilerleme isteği (İng. Time Advance Grant, TAR) mesajı gönderilir. Bu mesaj ile birlikte GALT algoritması çalışmaya başlar. Bu algoritmada hesaba katılan düğümlerin mantıksal zaman (T_i) ve ileri bakma (L_i) değerlerinin toplamının en küçüğü bulunacaktır. Eğer bir federenin ilerlemek istediği zaman bu değerden küçük ise federeye zaman ilerleme onayı (İng. Time Advance Grant, TAG) verilir. Şekil 2.1'deki örnekte Federe 2'nin ilerlemek istediği değer GALT değerinden küçük ve eşit olduğu için zamanını ilerletebileceklerdir. Fakat Federe 1 ve 3 zamanlarını ilerletemeyeceklerdir. Yeni GALT değeri ilerlemek istedikleri zaman olana kadar beklemek zorundadırlar.

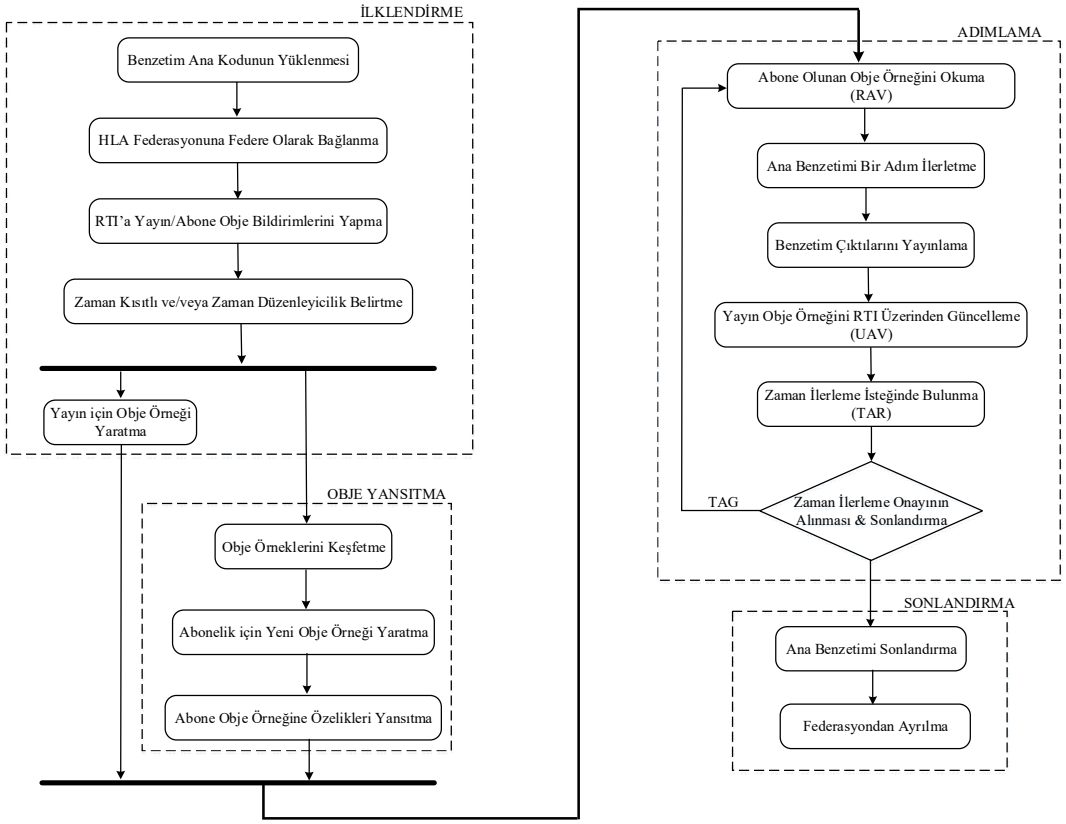


Şekil 2.1 Zaman İlerlemesi.

RTI ağdaki verileri dağıtmak için federasyona katılan federelerin yayın ve abone isteklerini DDM ile kontrol eder. RTI bu sistemde akıllı bir ağ anahtarı görevini yapar. Veri alan tarafın ilgisine göre veriyi yönlendirir. DDM'de federasyon yönlendirme uzayı (İng. Routing Space, RS) tanımlanır. RS boyutlardan oluşmaktadır. Boyutlar ise ölçüler setinden oluşan bölgeleri tanımlamak için kullanılır. Ölçüler ise çok boyutlu RS ile sınırlı uzunluklardır. Her bir yayıncı ve abone federe kendine ait bölgeyi tanımlar ve bölgelerin kesişimi boş küme değil ise RTI yayıncı ile abone arasında haberleşmenin gerçekleşmesini sağlar. RTI'da tanımlı metotlar sayesinde belirli bir obje örneğinin özelikleri belirlenmiş bir bölgeye bağlanabilir [11].

DDM yayın verisini ilgili abonelere göndermek için eşleştirme algoritmasını kullanır. Eşleştirme algoritmasında ilk olarak abone düğüm obje sınıfının özelliğine abone olma isteği (İng. Subscribe Object Class Attributes, SOCA) mesajını gönderir. Bu mesajın içeriğinde abone olunmak istenilen sınıfa ait kimlik bilgisi, ilgilenilen özelikler ve abone bölge bilgileri vardır. İlgili yayıncılar bu bilgileri kayıt ederler. Yayın yapacak düğüm bir sınıf özelliği güncellediğinde (İng. Update Attribute Value, UAV) mesaj gönderir. Yayın mesajı içeriğinde yayının yapılacağı sınıfın kimlik bilgileri, yayınlanan objenin özelikleri ve bölge bilgileri bulunmaktadır. DDM servisi eşleştirme algoritması; sınıf kimliklerinin aynı olmasını, yayın yapılan obje özelikleri ile abone obje özeliklerinden en az birinin örtüşmesini ve yayın ile abone bölgelerinin kesişmesini kontrol eder. Eğer bu şartlar sağlanıyor ise eşleşme bulunmuş olur. Yayın yapan düğüm UAV mesajını eşleşme bulunan düğümlere çoklu yayın grubu oluşturup gönderir.

HLA'da TM ve DDM servislerinin oluşturduğu ağ trafiği [1] çalışması üzerinden incelenebilmektedir. Bu çalışmada bağımsız tasarlanan bir benzetim bir federasyona tanıtılmıştır. Tanıtılan bu federe benzetim modeli federesi (BMF) olarak adlandırılmıştır. Şekil 2.2'de BMF'nin bir HLA benzetiminde çalışması esnasında oluşan akışın diyagramı verilmiştir. BMF federasyon obje model şablonunu okuyup ayrıştırır. Böylece her bir obje sınıfının özelik ve veri tipi yapısına ulaşır. Daha sonra yoksa federasyonu yaratır ve federasyona katılır. Benzetim obje model şablon dosyasını okuyarak RTI'ya hangi obje sınıfının hangi özeliklerinde yayın yapacağını veya abone olacağını bildirir. Obje örneği yaratılması, obje sınıflarının deklarasyonundan sonra yapılır. Bu obje örneği benzetim modelinin yayınları için kullanılmaktadır. BMF kendisini kısıtlı ve düzenleyici federe olarak tanıtır. Obje yansıtma aşamasında abone olunan obje sınıfı keşfedilir ve değeri yansıtılır. Obje sınıfı keşfedildiğinde ise (yeni obje örneği aboneliği) BMF keşfedilen obje sınıfının örneğini yaratır. Adımlama benzetimin ana kısmıdır. HLA'dan abone olunan obje örneği için veri alınır. Daha sonra benzetim modeli bir adım çalıştırılır. Elde edilen sonuçlar ile yayınlanan obje örneği güncellenerek RTI üzerinden gönderilir. Adımlamada son olarak RTI'dan zaman ilerleme isteğinde bulunulur. Benzetim devam ediyorsa RTI'dan zaman ilerleme onay mesajı bekler. Onay alındıktan sonra adımlama tekrar çalıştırılır. Adımlama aşamasında benzetimin koşması hariç bahsedilen bu işlemler TM ve DDM servisi aracılığı ile ağda trafik oluşturmaktadır. Benzetim sonlandırıldığında ise federasyondan ayrılma işlemi gerçekleştirilir ve ilgili hafızalar temizlenir.



Şekil 2.2 BMF'nin Akış Diyagramı [1].

Sonraki bölümlerde HLA ile ilgili referans çalışmalara değinilmiştir. İlk bölümde HLA'nın kullanım alanlarına değinilmiştir. İkinci bölümde HLA ile yapılan benzetimlerin performansını artırmak için uygulanan yöntemlerden olan yük dengeleme yöntemi ile ilgili yapılan çalışmalara değinilmiştir. Daha sonraki bölümlerde sırası ile tutucu eş zamanlama ve iyimser eş zamanlama uygulamalarındaki GALT algoritması ile eşleştirme algoritmalarında kullanılan yöntemlerle ile ilgili referans çalışmalar incelenmiştir. Son olarak HLA benzetimlerinde kullanılan ağ topolojilerine değinilmiştir.

2.1 HLA'nın Kullanım Alanları

Günümüzde yaygınlaşan HLA benzetim altyapıları birçok uygulamada tercih edilmektedir. Örneğin HLA ile yapılan bir benzetimde birçok farklı parçadan oluşan farklı motorların üretildiği ve bütünlendiği bir fabrika için ihtiyaca cevap verecek yeni bir üretim hattının kurulup kurulamayacağı tespit edilebilmektedir [12]. Hassas zamanlama ihtiyacı olan küresel konumlama ve uydu sisteminin (İng. Global Navigation and Satellite System, GNSS) L1 ve L2 uydu sinyalleri ile alıcının hareketini modelleyen

ve seyrüsefer mesajını oluşturan benzetimi HLA ile yapılabilmektedir [13]. RF, analog ve sayısal tasarımlardan oluşan yonga üzeri sistemler (İng. System On-Chip, SoC) farklı ortamlarda tasarlanmaktadır. Örneğin bir Buck (alçaltıcı) tip çevirgeç; kontrolcü içeren sayısal devre, pasif bileşenler ile yükten oluşan analog devre ve analog ile sayısal devrelerin bir arada bulunduğu analog-sayısal dönüştürücü (İng. Analog to Digital Converter, ADC) devre alt bloklarından oluşabilir. Bu alt bloklar SPICE veya Verilog ile tasarlanmış olabilir. Fakat HLA'da federasyon obje modeli yardımıyla bu bloklar arasında veri alışverişi yapma imkânı sağlanmaktadır. Böylece özel yöntemler (İng. Ad hoc) ile sağlanamayan farklı modellerin birlikte çalışabileceği bir ortam HLA tarafından SoC'lara için sağlanmış olur [14]. Hibrit sistem uygulamalarında HLA kullanmak mümkündür. Hibrit sistem, bir otomobilin vites değişimleri gibi ayrık olaylar ile Matlab üzerinde tasarlanmış motor modeli gibi sürekli sistemlerden oluşmaktadır. HLA yardımıyla hibrit sistemlerde her bir sistem kendi geliştirme ortamında tasarlanıp birlikte kullanılabilecektir [15]. Yine benzer şekilde siber fiziksel enerji sistemlerinin analizinde, sürekli zamanda çalışan güç sistemleri ile ayrık olay tabanlı çalışan bilgi ve haberleşme teknolojisi (İng. Information and Communication Technology, ICT) ağlarının benzetimi HLA ile yapılmaktadır. Bir örnek çalışmada [16] akıllı kafes (İng. Smart grid) uygulamaları için güç sistemi ve ICT'nin beraber analizi yapılmıştır. Güç sisteminde oluşan aşırı yüklenme durumunda ağ üzerinden kontrol algoritmaları ile müdahale olmaması durumu, ağda trafiğin az ve yoğun olması gibi durumlar incelenmiştir. Sonuç olarak HLA yardımıyla, beraber analizin önemi ortaya konmuştur. Güç sistemi ile ICT'nin birlikte çalışmasında kullanılan yaklaşımların analizi içinde ayrıca çalışma yapılmış ve gerçek zamanlı uygulamalarda kullanılan döngüde sistem (İng. System in The Loop, SITL) ile HLA karşılaştırılmıştır [17]. SITL ve HLA farklı haberleşme mimari ve teknolojilerinin güç sistemi uygulamalarındaki etkisini test etmek için uygun araçlardır. SITL gerçek zamanlı uygulamalarda döngüde donanım testleri ile gerçek bileşenler ile test yapmayı desteklese de HLA farklı teknolojileri, modelleri ve algoritmaları daha kolay desteklediği için kullanıcının benzetim seçeneklerini artırmaktadır. Ayrıca SITL'da ağ modelinin genişleyebilmesi kullanılan bilgisayara bağlı iken HLA'da böyle bir kısıtlama yoktur.

2.2 HLA'da Yük Dengeleme Çalışmaları

HLA federeler arasındaki fiziksel mesafeden kaynaklanan gecikmelerin iyileştirilmesi ile ilgili bir çözüm sağlamaz. Haberleşme yüklerinin federelerin uzaklığına göre statik dağıtımı bu gecikmeyi iyileştirmek için uygulanan çözümlerden bir tanesidir. Alternatif çözüm olarak dinamik dengeleme yöntemi de kullanılmaktadır [18]. Yapılan çalışmalarda yük dağıtımının merkezi ve dağıtık yapılması durumu [19], ekstra filtreleme ile yük dağıtımı yapılması [20] ve yük tahmini ile yük dağıtımı yapılması [21] durumları için mesaj gecikmelerindeki azalmalar ile elde edilen performans iyileşmeleri incelenmiştir. Yük dengeleme ile birbirileri ile yoğun haberleşen düğümleri birbirlerine yakın yerleştirilerek haberleşme gecikmesini azaltmak için federelerde; yerel yönetim ajanları, haberleşme yük monitörü ve göç yöneticisi servislerinin fazladan çalışması gerekmektedir. Bu yöntemde herhangi bir HLA servisinin iş yükü azaltılmadığı gibi eklenen servisler yüzünden haberleşme maliyeti artmaktadır. Fakat yük dengesizliği durumunda uygulama zamanının %45 kadarı boşa harcanabilmektedir [22]. Yük dengeleme için Intel Tofino gibi ağda programlama özelliği olan Ethernet anahtarlarını [23] ya da ağda hesaplama yöntemiyle (İng. In-network computing) çalışan Ethernet anahtarlarını [22] kullanmak mümkündür. Bu sayede bahsedilen servislerin düğümlerden kaldırılması sağlanabilecektir. Dolayısıyla da uygulamalarda boşa harcanan zaman engellenebilecektir.

2.3 Tutucu Eş Zamanlamada GALT Algoritması Çalışmaları

HLA'da işlem yapan düğümlerdeki mantıksal süreçleri (İng. Logical Process, LP) eş zamanlamak için iki farklı yöntem izlenmektedir. Bunların ilki tutucu eş zamanlamadır. Tutucu eş zamanlamada mesajların sıralı işlenmesi amaçlanmaktadır. Mesajları zaman sıralı oluşturur ve ağ bu mesajları sıralı olarak iletmeyi garantiler. Gelen mesaj yığını boş ise LP bekler. Fakat bu durum ağda kitlenmeye neden olur. CMB algoritmasında boşta beklenirken "null" mesajı gönderilerek bu durumun önüne geçilir. Bir ileri bakma değeri tanımlanır ve her benzetim adımında benzetim zamanın üzerine bu ileri bakma değeri eklenerek "null" mesajları gönderilir. Fakat ilerlenmek istenilen benzetim zamanı ile o anki benzetim zamanın arası fazla ve ileri bakma değeri küçük ise "lookahead creep" durumu ile ağda trafik oluşur. YAWNS algoritmasında ise global bir bariyer değeri oluşturulup LP'ler bu noktada eş zamanlanır ve zaman ilerlenebilecek güvenli minimum nokta bulunur [24].

TM servisinin performansının artırılmasını için temelde tutucu eş zamanlama yönteminde kullanılan GALT algoritmasının gecikmesinin iyileştirilmesi gerekmektedir. Bunun için ölçeklenebilir bir sunucu (multiple-Fedserv) mimarisi çalışmasında 256 federe bulunan bir federasyon kurulmuştur. Federeler 1Gb Ethernet ile iki seviyeli bir ağ topolojisinde birbirlerine bağlanıp, 1KB mesaj uzunluğuna kadar testler yapılmıştır. Zaman ilerleme kıyaslamalarında eklenen sunucular ile ölçeklenebilirlik sağlanmış ve TM servisinin performansı artırılmıştır. Fakat abone/yayın işlemlerinde performans bazı testlerde iyileşmiş ancak bazı testlerde Savunma Modelleme ve Benzetim Dairesi'nin (İng. Defense Modeling and Simulation Office, DMSO) kıyaslamalarına göre düşüş gözlenmiştir [25].

TM servisinin GALT algoritması kolektif haberleşme operasyonları kullanılarak da gerçekleştirilebilir. Bunun için düğümler ilk olarak bariyer işlemine girip birbirlerine eş zamanlanmaktadır. Daha sonra ise hepsini azaltma (İng. Reduce all) işlemi ile algoritmaya katılan düğümlerden zaman bilgilerinin toplanmasıyla GALT değeri hesaplanıp düğümlere aktarılabilecektir [26]. GALT algoritmasını bu şekilde çalıştırmak için ise alanda programlanabilir kapı dizileri (İng. Field Programmable Gate Array, FPGA) kullanarak kolektif haberleşme algoritmalarının uygulandığı çalışmalar mevcuttur [27], [28]. Bu çalışmada hepsini azaltma operasyonu uygulamasında yaklaşık 10 katlık bir iyileşme sağlanmıştır. Ayrıca Wilson fermiyonları ve Schrödinger fonksiyonları da yaklaşık %40 hızlandırılmıştır. Kolektif haberleşme operasyonlarını FPGA'ya aktarma yerine Mellanox SwitchIB-2 gibi uygulamaya özel tümleşik devreye (İng. Application Specific Integrated Circuit, ASIC) aktarmak da mümkündür. ASIC ile bariyer ve hepsini azaltma kolektif haberleşme operasyonları sırası ile 7,1 ile 10 kat arasında iyileştirilmiştir. Testler ise şişman ağaç ağ topolojisinde 5120 düğüme ve 32KB mesaj boyutuna kadar yapılmıştır [29], [30]. Ayrıca Mellanox'un MVAPICH2-X uygulamasında 7861 düğüm için ağdaki gecikmelerde hepsini azaltmada 5,1 kat ve bariyerde 7,1 katlık bir iyileşme sağlanmıştır [31]. Ağda hesaplama yöntemi ile de hepsini azaltma kolektif haberleşme algoritması gerçekleştirilebilmektedir [32]. Bu çalışmanın çok iyi ölçeklenebilmesi sayesinde yapılan benzetimlerde 16K düğüm için 1us gecikme ölçülmüş ve yazılımda gerçekleştirilen hepsini azaltma işlemine göre yaklaşık 40 katlık bir hızlanma elde edilmiştir. Benzer şekilde yapay zekâ öğrenme algoritmalarında kullanılan hepsini azaltma işleminin performansını da anahtar tasarımı ile iyileştirmek

mümkündür [33]. Bu çalışmada birbirlerine 10Gb Ethernet arayüzü ile bağlı 8 adet sunucu, Xilinx Ultrascale FPGA üzerinde uygulanan anahtar ile birlikte çalıştırılmıştır. Görüntü eğitme uygulamasında ortaya konulan ölçeklenebilir ve bant genişliği kayıpsız bu anahtar ile 1,67 kat hızlanma sağlanmıştır. Ağ anahtarı tasarımının değiştirerek kolektif haberleşme operasyonlarının performansını artırmanın haricinde hepsini azaltmada kullanılan haberleşme algoritmaları değiştirilerek de ağdaki gecikmenin azaltılması mümkündür. [34] çalışmasında önerdikleri PAARD algoritması ile Dragonfly ağ topolojisinde Halving-doubling algoritmasına göre %75,73 ve Ring algoritmasına göre %98,63'lük bir iyileşme elde edilmiştir. Kolektif haberleşme algoritmalarının performansını artırmak için uygulanan bir diğer yöntem ise anahtarsız Kautz ağ topolojisi kullanımıdır [35]. Bu topolojide FGPA'in dış dünya ile olan yüksek bant genişlikli optik arayüzleri desteklemesi ve optik arayüzlerden gelen verinin doğrudan FPGA'in yüksek bant genişlikli hafıza (İng. High Bandwidth Memory, HBM) kanalına yazılması sonucu, kolektif haberleşme algoritmaları Dragonfly bir ağa göre 7 kat düşük gecikme ile gerçekleştirilmiştir. Kolektif haberleşme tabanlı yöntemlerde ağda GALT değeri saklanmadığı için her GALT hesabı isteğinde tüm ilgili düğümlerin birbirine eş zamanlanıp, GALT işlemine yine bu düğümlerin katılması gerekmektedir. Günümüzde gittikçe popülerlik kazanan ağda programlama özelliği olan Ethernet anahtarlarında anahtar üzerinde veri saklamak mümkündür [36].

2.4 İyimser Eş Zamanlamada GALT Algoritması Çalışmaları

Eş zamanlama yönteminin diğeri ise iyimser eş zamanlamadır. Bu yöntemde LP'lere mesajların sıralı iletilmesine gerek yoktur. Bunun yerine sıralı olmayan mesajlar alındığında geri alma mekanizması çalışır. LP'ler durdurulmaz ve zaman sıralı çalışmanın bozulma riski alınır. Zaman sıralama bozulduğunda durum değerleri değişebilir veya başka LP'ye mesaj gönderilebilir. Bu koşullardan herhangi birisi gerçekleşir ise zaman atlama (İng. Time Warp, TW) mekanizması ile geri alma işlemi gerçekleştirilir. TW mekanizmasında iki farklı kontrol yöntemi vardır. Lokal kontrol yönteminde LP'ler mesajları işlemeden LP'lerin durumları saklanır. Geri alma mekanizması bu durumları tekrar yükler ve benzetim zaman ilerleme öncesi adımdan devam edebilir. Diğer bir uygulamada yapılan işlemler tersten yapılmasıdır. Hafıza ihtiyacı daha azdır. Fakat her zaman uygulanamayabilir. Diğer LP'lere gönderilen mesajı geri almak için ise ilgili LP'ye anti mesaj gönderilir. Fakat bu durumda etkilenen LP ve onun etkilediklerine anti

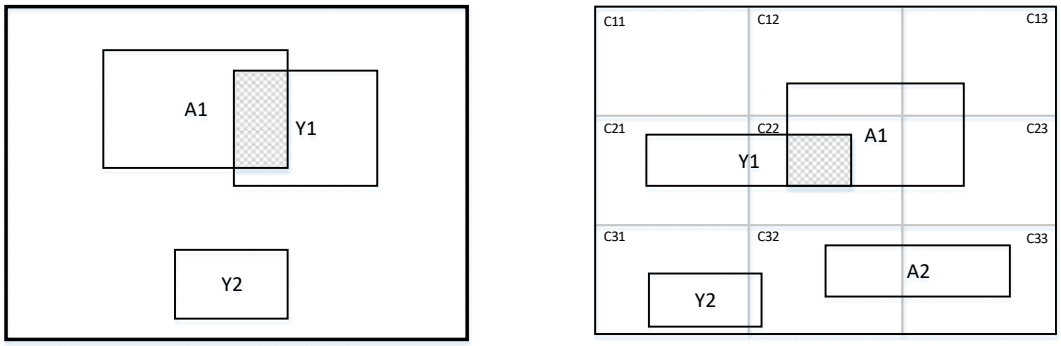
mesaj gönderilerek arka arkaya anti mesaj trafiğinin oluşma riski vardır. Global kontrol yöntemlerinde ise olabilecek bir geri alma işlemi için bir zaman etiketi alt sınırı belirlenir. Bu değer global sanal zamandır (İng. Global Virtual Time, GVT) [24]. GVT, GALT algoritmasına benzer şekilde çalışır. GVT ile mesajlar ve anti mesajlar için geri alınabilecek en küçük zaman etiketi hesaplanır. GVT uygulamasında problemli durumların birisi geçiş mesajı alınmasıdır. Herhangi bir düğüme zaman ilerleme isteği geldikten sonra zaman ilerlemesi onaylanmaktadır. Sonradan gelen özelik değeri güncelleme mesajının zamanı ise o anki benzetim zamanından geride ise geçiş mesajı oluşmuş olur. Bu durumda benzetim zamanı geri alınır ve gelen mesaj işlendikten sonra benzetim zamanı tekrar ilerletilir. Asenkron haberleşme ile GVT uygulamasında geçiş mesajları için performans iyileşmesi yakalanmıştır [37]. Ayrıca geri alma sayısını azaltılması için TW mekanizmalarının iyileştirilmesi yönünde çalışmalar yapılmaktadır [38]. Uzun benzetim çalışmalarında kullanılan farklı TW geri alma mekanizmaları sonucunda ortaya çıkan geri alma işlem sayıları nedeniyle benzetim süreleri yaklaşık 2 kat uzayabilmektedir [39].

2.5 Eşleştirme Algoritması Çalışmaları

Dağıtık interaktif benzetimlerde (İng. Distributed Interactive Simulation, DIS) düğümler birbirleri ile haberleşirken veri dağıtımı için herkese-yayın kolektif haberleşme operasyonunu kullanmaktadır. Fakat büyük ölçekli dağıtık benzetimlerde herkese-yayın kolektif operasyonu ağırlıklı haberleşmek ağda gereksiz trafik oluşmasına neden olacaktır. HLA ise doğrudan herkese-yayın yapmak yerine haberleşme ihtiyacına göre çoklu yayın grupları oluşturabileceği DDM servisini tercih etmektedir. DDM servisi yayın/abone isteklerine filtreleyip eşleştirebileceği temel beş farklı yaklaşımı desteklemektedir. Bu yaklaşımlar sınıf-tabanlı, bölge-tabanlı, kafes-tabanlı, hibrid ve sıralama-tabanlı olarak isimlendirilirler [40]. Sınıf-tabanlı yaklaşımda obje grupları sınıf olarak tanımlanır. Federe objenin sınıf özelliği değerine abone olabilir. Bu durumda o sınıftaki tüm objelerin seçilen özelik değerinde güncelleme alınır. Bu yöntemde sınıfın özeliklerinden ihtiyaç duyulmayanları filtreleme ihtiyacı yoktur. Çünkü seçilirken en başta sınıf özeliği seçilmektedir. Sınıf-tabanlı yaklaşımda ortak hafızalı çok çekirdekli makine ile yapılan paralel uygulamada mesaj gecikme süreleri iyileştirilebilmektedir [41]. Fakat bu uygulama performansı federe sayısına, bölge-tabanlı yaklaşıma göre daha hassastır. Ayrıca sınıf-tabanlı yaklaşımda aynı sınıf özeliklerine sahip olan tüm objelerin bilgisi aboneye iletildiği için, bu yaklaşım içerisinde birkaç obje bulunan küçük

federasyonlar için daha uygundur. Bölge-tabanlı yaklaşımda yayıncı ve abone federeler RS'de bölge tanımı yapmak zorundadır. Herhangi bir yayın mesajının bölge tanımı ile abone mesajının bölge tanımları kesişecek olursa eşleşme bulunmuş olur. Eşleşme durumunda abone federe, yayıncı federenin kesişim olan obje ve etkileşimlerinin güncellenen özelik ve parametre bilgilerine ait mesajları ağdan alır. Yayıncı düğümün mesaj dağıtmadan önce tüm abone düğümler ile bölge keşişimi hesaplaması ihtiyacından dolayı ortaya çıkan yüksek işlem gücü gereksinimi nedeniyle bölge-tabanlı yaklaşım kaba kuvvet (İng. Brute force) yaklaşım olarak da adlandırılır.

Şekil 2.3 (a)'da bölge-tabanlı yaklaşım ile çalışan eşleştirme algoritması ile ilgili örnek yer almaktadır. Yayın yapan düğümler Y1 ve Y2, abone düğüm ise A1'dir. Y1 ve A1 kesişmektedir. Bunun sonucu olarak da Y1 ile ilintili obje tarafından yapılan özelik güncellemeleri, A1'i oluşturan federeye yönlendirilmektedir. Y2 ile A1 kesişimi olmadığından Y2 için bu durum geçerli değildir. Fakat bölge-tabanlı filtrelemede A1, Y1'deki tüm özelik güncellemeleri alır. Bunun önüne geçmek için kafes-tabanlı yaklaşımda RS kafes hücrelerine bölünmektedir.



(a) (b)

Şekil 2.3 DDM servisi uygulamasında (a) Bölge-tabanlı (b) Kafes-tabanlı yaklaşım.

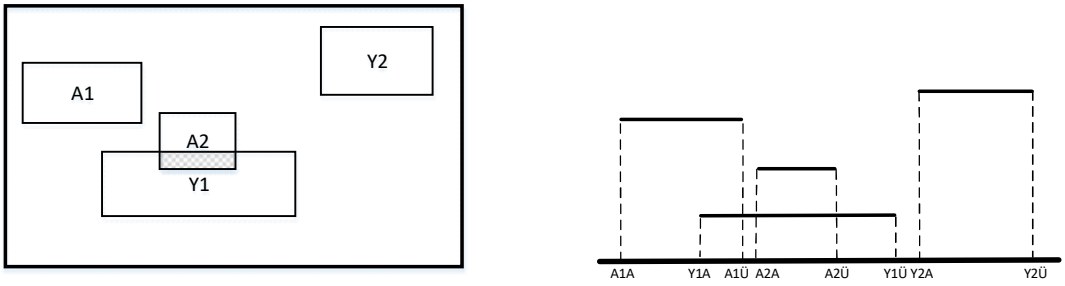
Kafes-tabanlı yaklaşımda yayıncı ve abone düğümler hücreler ile eşleştirilirler. Eğer yayıncı ve abone en az 1 hücrede kesişime sahip iseler eşleşeme bulunmuş olur. Şekil 2.3 (b)'de kafes-tabanlı yaklaşım ile çalışan eşleştirme algoritması ile ilgili örnek vardır. Örnekte A1 ve Y1, C22 kafes hücresinde kesiştikleri için Y1, A1'i abone olarak görür. Fakat Y2'de A2'yi abone olarak görmektedir.

Kafes-tabanlı yaklaşımda sabit, dinamik ve optimize dinamik olmak üzere üç farklı uygulama metodu vardır. Sabit uygulama metodunda RS'de kafesler açılış esnasında belirlenir ve çoklu yayın grupları bu kafesler ile ilintili oluşturulur. Her federe, yayın/abone bölgelerinin hücreler ile eşleştirmesini kendisi yapar. Böylece kafes hücrelerinin boyutu ve sayısı ilk açılışta belirlenir ve benzetim boyunca sabit kalır. Eğer federenin yayın/abone bölgesi hücre ile kesişiyor ise ilgili hücrenin çoklu yayın grubuna katılır. Dinamik uygulama metodunda çoklu yayın grupları o anki yayın ve abone bölgelerine bağlı olarak dinamik değişir. Çoklu yayın grupları hücrede en az bir yayıncı ve bir abone olduğunda tanımlanır. Bu sayede gereksiz veri gönderimi azaltılır. Her ev sahibinin yerel RTI bileşeninde (İng. Local RTI Component, LRC) kesişim algılama ve tetikleme mekanizması çalışır. Bu bir çeşit dağıtık DDM koordinatörlerin uygulaması olarak değerlendirilebilir. Her bir koordinatör belirli bir hücre setini izler. Böylece merkezi bir yapı olmadığı için ölçeklenebilme daha kolay olur. Kafes-tabanlı yapıda aynı hücrede olmalarına rağmen kesişmeyen hücreler de çoklu yayın grubuna dâhil olur ve gereksiz mesajlara neden olurlar. Ayrıca kesişen hücrelerin, kesişmeyen yerleri olduğu için yine gereksiz veri iletimi olacaktır. Bu gereksiz veri iletimlerini engellemek için kafes-tabanlı yaklaşımda optimize dinamik uygulama metodu geliştirilmiştir. Bu durumda alıcı ve verici tarafında filtreleme uygulanabilir. Eğer yayıncı federe, abonenin ilgilenmediği güncellemeler olmadığını biliyorsa bu verileri filtreleyip gereksiz veri gönderimini engelleyebilir. Buna gönderici tarafı filtreleme olarak adlandırılır. Eğer yayıncı federe abonenin güncelleme ihtiyacını bilmiyor ise mesajı gönderir. Alıcı federenin LRC'si federeye verinin alınıp alınmayacağına karar verir. Bu ise alıcı tarafında filtreleme olarak adlandırılır. Gönderici tarafta filtreleme daha çok istenir. Fakat hesap maliyeti yüksektir. Gönderici tarafında filtreleme yapılırsa gereksiz mesaj gönderilmesi ile alınması ve alıcıda filtreleme yapılması engellenir [42]. Kafes-tabanlı mimaride en uygun hücre boyutu seçiminin performans üzerinde önemli etkileri vardır ve birçok çalışmaya da konu olmuştur [43] [44]. Eğer CPU çalışma frekansı yüksek ağ hızı düşük ise hücre boyutları küçük, aksi durumda büyük hücre boyutu seçilmelidir. Başka bir ifade ile haberleşeme maliyetinin hesaplama maliyetine oranı ile hücre boyutunun artışı doğrudan ilişkilidir. Ayrıca en uygun hücre boyutu değeri benzetim boyunca değişen dinamik bir değer de olabilir [45]. Kafes-tabanlı yapı ile ilgili yapılan çalışmalar da performans gözetimi, benzetim gecikme değeri yerine mesaj yoğunluğundaki iyileştirme karşılaştırması ile yapılmıştır.

Kafes-tabanlı yaklaşımda DDM yöneticisi isimli bir alt model vardır. DDM yöneticisi veri filtrelemekten sorumludur. İçerisinde yerel DDM yöneticisi ve DDM koordinatörü bulunur. Yerel DDM yöneticisi tüm federelerde bulunur. Çoklu yayın grubu DDM koordinatörü tarafından hesaplanır ve yayıncının yerel DDM yöneticisine gönderilir. Yayıncının yerel DDM yöneticisi abonenin yerel DDM yöneticisine bağlanır ve veriyi doğrudan gönderir. DDM koordinatörü tüm yönlendirme uzayını hücrelere böler ve her hücre için abone ve yayıncı listesi tutar. Yayıncı ve abone bölgeleri arasında direk bağlantı olamasa bile yayıncı ve abone isteklerini ilgili hücrenin listesine yazarak iletişimlerini gerçekleştirirler. Bu yapıda yayıncı veya abone bölge değiştirdikçe ilgili hücrenin listelenmesi gerekir ve buna "liste güncelleme maliyeti" denir. Bu maliyet CPU hızına ve hafızasına bağlı olarak DDM koordinatörü için sınırlandırıcı olabilir. DDM filtreleme dağıtık olduğu için DDM yöneticideki mesaj yapısından dolayı ayrıca bir "mesaj maliyeti" vardır. Yayın/abone bölgesi değiştiğinde bu bilgi federe modelinden DDM koordinatörüne iletilir. Buradaki maliyet federe sayısı ve yayın sıklığına bağlı değişir. Eğer güncelleme frekansı yüksek ise çok mesaj olacaktır. DDM koordinatörü her yayıncı için çoklu yayın grubu oluşturduğunda, bunu yayıncılara iletmesi gerekmektedir. Yayıncı DDM koordinatöründen çoklu yayın grubu bilgisi aldığında, obje özeliklerini çoklu yayın grubundaki üyelere iletmek zorundadır. Bu aşamanın maliyeti çoklu yayın grubunun haberleşme maliyeti ile ilintilidir. Kafes-tabanlı yaklaşımda güncelleme bölgelerinin, abone bölgeleri ile karşılaştırma ihtiyacı yoktur. DDM koordinatörü bölgeleri RS'deki hücrelere eşleştirir. Fakat bölge-tabanlıda DDM koordinatörü her güncelleme bölgesi için çoklu yayın grubu oluşturmak için güncelleme bölgelerini abone bölgeleri ile RS'de tek-tek eşleştirmek zorundadır. Bu yüzden bölge-tabanlı yaklaşımda obje sayısı çok olursa eşleştirme maliyeti kafes-tabanlı yaklaşıma göre yüksek olmaktadır. Hibrid yaklaşımda ise bölge-tabanlı ve kafes-tabanlı yaklaşımlar birlikte kullanılmaktadır. Hibrid yaklaşım, bölge-tabanlı yaklaşımdaki eşleştirme maliyetini düşürmeyi hedeflerken, kafes-tabanlı yaklaşımdaki liste güncelleme ile mesaj maliyetini düşürmeye çalışmaktadır. Hibrid yaklaşım ilk olarak sabit uygulama metodu kullanan kafes-tabanlı yaklaşım ile kabaca kesişim bulurken, daha sonra bölge-tabanlı yaklaşım ile daha hassas kesişim bulacak şekilde çalışmaktadır. Bu sayede kafes-tabanlı yapıda gereksiz dolaşan mesajlar azaltılırken, bölge tabanlıda yapılacak eşleşme sayıları azaltılabilecektir. Hücre sayısı azaldıkça kafes-tabanlı yaklaşımdan gelen gereksiz mesaj sayısı artarken bölge-tabanlı yaklaşımdan gelen filtreleme özeliği ile bu gereksiz mesaj

sayıları azaltılır. Bu yüzden haberleşeme maliyeti hesaplama maliyetinden yüksek olan uygulamalarda az hücre sayılı hibrid yaklaşım kafes-tabanlı yaklaşımdan daha düşük gecikmeye sahip olacaktır [46].

Sıralama-tabanlı yaklaşımda abone ve yayın bölgelerinin içinde bulundukları işlem uzayı koordinat sisteminin eksenlerine izdüşümleri alınarak listeler oluşturulur. Kesişimler boyutlar bazında incelenmektedir. Liste üzerinde arama yapılarak bölge kesişimi bulunur. Şekil 2.4.a'da gösterilen iki boyutlu düzlemdeki yayıncı ve abonelerin, bir eksen için iz düşümleri Şekil 2.4.b'de gösterilmiştir. Kesişim bulunurken iki farklı tipte kesişim oluşabilmektedir. Bu kesişim tiplerinin ilki bir bölgenin diğer bölge ile iç içe geçmesidir. Şekil 2.4.b'de A1'in üst sınırı (A1Ü), Y1'in alt sınırından (Y1A) büyük olduğu için ve A1'in alt sınırı (A1A), Y1'in üst sınırından (Y1Ü) küçük olduğu için izdüşümü alınan eksende A1 ve Y1 iç içe geçerek eşleşmektedir. Diğer tip kesişim ise bir bölgenin diğer bölgeyi kapsamasıdır. Bunun örneği ise A2 ile Y1 arasında bulunmaktadır. Bu şekilde bulunan bölge kesişimi ayrı bir matriste saklanır [47]. Sıralama-tabanlı yaklaşım sıralama algoritmaları sayesinde diğer yaklaşımlara göre birçok gereksiz kontrolü ortadan kaldırmaktadır. Ayrıca sıralama-tabanlı yaklaşımda kullanılan algoritmalar optimize edilmeye uygunlardır [48]. Sıralama-tabanlı Raczy algoritmasının gecikme değeri, maksimum adım değerinin, tüm bölgelerdeki tüm boyutlarda her bir boyutun üst sınırına oranı artıkça azalmaktadır. Düşük oranlarda ise gecikme değerini iyileştirmek için dinamik büyük uzaysal ortamlara özel tasarlanan sıralama-tabanlı yaklaşım ile geliştirilmiş farklı algoritmalar mevcuttur. Razcy sıralama-tabanlı yaklaşım ile büyük uzaysal ortamlara özel tasarlanan sıralama-tabanlı yaklaşımın bir arada kullanıldığı çalışmada bölge-tabanlı yaklaşıma göre yaklaşık 9-kat iyileşme sağlanmıştır [49]. Fakat birlikte çalışırken hangi algoritmanın seçileceğinin kararının verilmesi işlemi benzetimi karmaşıklaştırmaktadır.



Şekil 2.4 Sıralama-Tabanlı Yaklaşımda (a) Yayın/Abone Bölgeleri (b) Bir eksen için İzdüşümüler.

Bahsedilen yaklaşımların haricinde ajan-tabanlı gibi daha az yaygın yaklaşımlar da mevcuttur. Bu yaklaşımda bir abone isteği olduğunda ilgili abone isteği için akıllı mobil ajanlar yayıncı içine ilgili özelik ve etkileşimleri takip etmek amacıyla yerleşir. Yayıncı, obje özeliği veya etkileşim parametresi güncellediğinde ilgili ajanlar veriyi yakalar. Veri üzerinde filtreleme yapar ve sadece ilgili veri aboneye gönderir. Abone tarafından bölge güncellemesi yapıldığında ajanın filtreleme parametreleri güncellenir. Ajan-tabanlı yaklaşımda güncelleme bölgesi ihtiyacı yoktur. Çünkü ajan kendi federesi için tüm veri güncelleme ve dağıtımı ile ilgilenmektedir. Ajan-tabanlı yaklaşımın performansı, abone bölgesinin değişimi dinamik ve yayıncı-abone kesişimi yüksek ise iyidir. Fakat ajanların federelere açılış esnasında yüklenmesi açılış zamanını artırmaktadır. Ayrıca ajan-tabanlı yaklaşım kullanılan büyük ölçekli benzetimlerde yüzlerce ajan olmaya başladığında bölge-tabanlı ve kafes-tabanlı yaklaşıma göre gecikme süreleri daha uzun olmaktadır [50].

DDM servisinin performansını iyileştirmek için ortaya konulan bu yöntemlerin haricinde mevcut yaklaşımların uygulama metotlarının değiştirilmesi ile de performans iyileşmesi sağlamak mümkündür. Bölge-tabanlı ve kafes-tabanlı yaklaşımların performansını artırmak için CUDA tabanlı grafik işlemci ünitesine (İng. Graphics Processing Unit, GPU) [51] veya ağ işlemcisine [52] eşleştirme algoritmalarının aktarılması üzerine çalışmalar yapılmıştır. Bu çalışmalarda eğer bölgeler tekdüze dağıtılmış ise elde edilen iyileştirme artmaktadır. Bu iyileştirmeler hesaplama maliyeti üzerinden incelenmiştir.

2.6 HLA Benzetimlerinde Ağ Topolojileri

HLA'de ölçeklenebilir ağ topoloji yaratma üzerine yapılan [25] çalışmasında sunucu tabanlı bir ağ kurulup sunucu sayısı artırılarak ölçeklenebilirlik sağlanmıştır. Fakat TM servisinin performansı iyileştirilirken, DDM servisinin performansı bazı testlerde kötüleşmiştir. TM ve DDM servislerinin performansı Torus ağ topolojisi üzerinde çakışma olmayan haberleşme algoritmaları geliştirilerek birlikte de iyileştirilebilir. Torus ağ üzerindeki haberleşme rotaları değiştirilerek ve haberleşme adım sayısı düşürülerek haberleşme maliyet fonksiyonu düşürülebilmektedir [53].

Torus ve hasır doku gibi topolojiler ile ölçeklenebilir, çakışmayan ve düşük gecikmeli çözümler önerilse de [54], bu topolojilerde iki düğüm arası maksimum mesafe ağ

boyutuna bağlı olarak çok artmaktadır. Bu durumda ise kullanılan anahtar için paketlerin anahtar üzerinde tamponlanması sonucu oluşacak gecikme ve dolayısıyla da haberleşeme gecikmesi artmaktadır. Bu yüzden üç seviyeli şişman ağaç (İng. Three-tiered fat tree) ağ topolojisi seçilmiştir. Eğer 48-kapılı bir Ethernet anahtarı üç seviyeli şişman ağaç topolojisinde kullanılırsa 27648 düğüm ağa bağlanabilmektedir [55]. Şişman ağaç topolojisinde Ethernet anahtarının kapı sayısı artırılarak ağ kolayca ölçeklenebilmektedir.

Ölçeklenebilirlik ile ilgili yapılan referans çalışmada ölçeklenebilirliği sağlarken DDM servisinin performansının etkilendiği belirtilmiştir [25]. Fakat bu tez çalışmasında elde edilen ölçeklenebilirlik DDM servisi servisinin performansını etkilememektedir. Ayrıca anahtar-tabanlı yaklaşım ile DDM koordinatörünün çoklu yayın grubu oluşturma zorunluluğu ortadan kalkmaktadır. Çoklu yayın grubu oluşturmak için bölge-tabanlı yaklaşımdaki gibi merkezi DDM koordinatörü kullanan yaklaşımlarda ise ölçeklenebilme ile ilgili sıkıntılar mevcuttur. IS şişman ağaç mimarisinden dolayı kolayca ölçeklenebilmektedir. GALT algoritmasının referans çalışmalarda anlatılan yaklaşımlarında düğümler önce bariyer işlemine girip sonra hepsinin azaltma işlemi ile GALT hesabını yapmaktadır [27], [28], [29], [30], [31], [32], [33]. Fakat bu tez çalışmasında geliştirilen yaklaşım ile asenkron olarak GALT hesabı tek bir adımda yapılabilmektedir. Ayrıca iyimser eş zamanlamada GALT hesabı esnasında ortaya çıkan geçiş mesajları bu tez çalışması ile ortadan kaldırılmıştır. Referans çalışmalar ise geçiş mesajı oluşması durumunda uygulanan geri alma işlemlerinin sayısını azaltma yönündedir [37], [38], [39]. Eşleştirme algoritması için incelene referans çalışmalarda benzetim süresi yerine aşırı yüklenme, mesaj yoğunluğu gibi farklı metrikler üzerinden karşılaştırmalar yapılmıştır [43], [44], [46]. Alınan benzetim süresi ölçümlerinde ise haberleşme maliyetini dâhil edilmeden sadece hesaplama maliyeti için sonuçlar paylaşılmıştır [51], [52]. Bu yüzden incelenen referans çalışmalar ile bu tez çalışmasında yapılan çalışmaları aynı değerlendirme metrikleri üzerinden karşılaştırmak mümkün değildir. Literatüre yapılan katkıyı daha net ortaya koymak için standart bir CS kullanılarak IS geliştirmiştir. Bölüm 6'da verilen maliyet fonksiyonları ve senaryo testleri ile belirlenen metrikler doğrultusunda, CS ve IS karşılaştırılmıştır.

3. GALT VE EŞLEŞTİRME ALGORİTMALARI

HLA'nın TM ve DDM servisleri altında kullanılan GALT ve eşleştirme algoritmaları ile bu algoritmaların şişman ağaç ağda anahtar üzerinde nasıl çalıştığı bu bölümde anlatılmaktadır. İlk olarak anahtarın kullanılacağı şişman ağaç ağ topolojisi incelenmiştir. Daha sonra şişman ağaç ağ topolojisi üzerinde GALT ve eşleştirme algoritmalarının ortaya konulan anahtar-tabanlı yaklaşımda nasıl çalıştığı anlatılıp, algoritmaların sözde kodları (İng. Pseudo code) verilmiştir.

3.1 Şişman Ağaç Ağ Topolojisi

Üç seviyeli şişman ağaç topolojisi Şekil 3.1'de k yollu ağaç olarak gösterilmiştir. Bu örnekte k değeri dörttür. Bu değerin anlamı ağaçta her kapsülde çekirdek anahtar altında en fazla 4 yolun bulunabileceğidir. Bu durumda 4 yollu şişman ağacın kurulmasında yapılan hesaplar aşağıdaki gibidir [55]:

Kapsül sayısı: k=4 (I)

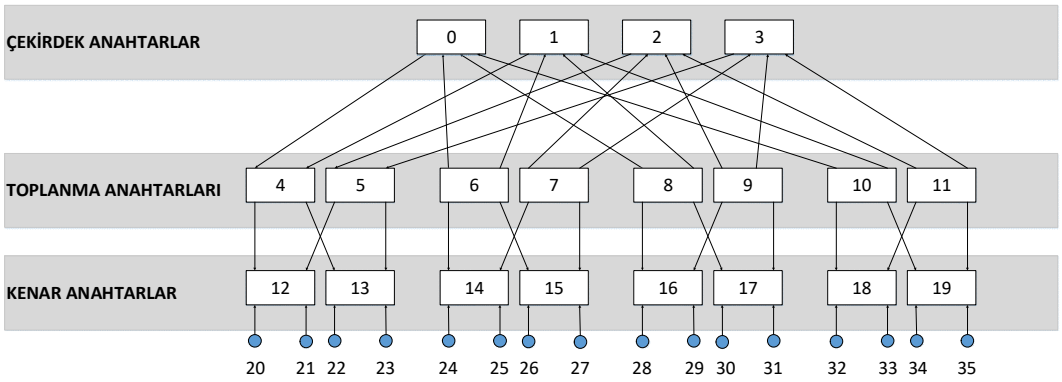
Her kapsüldeki toplanma ve kenar anahtar sayısı: k/2=2 (II)

Kenar düğümlerde bulunan anahtarların her birine bağlanan düğüm sayısı: k/2=2 (III)

Toplamda k=4 kapılı çekirdek anahtar sayısı: $(k/2)^2=4$ (IV)

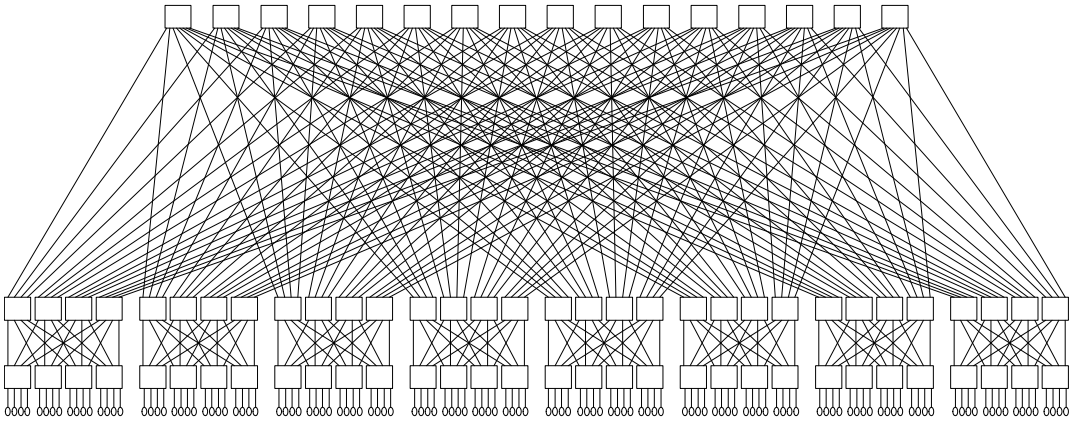
Desteklenen düğüm sayısı: $k^3/4=16$ (V)

Çekirdek anahtarların her bir portu k (4) kapsüllerin birisine bağlıdır. Bu tasarımdaki tüm anahtarlar özdeştir.



Şekil 3.1. Düğüm Sayısı 16 için Üç Seviyeli Şişman Ağaç.

Üç seviyeli şişman ağaçta ağ boyutu anahtar kapı sayısı artırılarak büyütülebilmektedir. Şekil 3.2'de 8 kapılı anahtarlar kullanılarak oluşturulan 128 düğümlü ağ gösterilmiştir.



Şekil 3.2. Düğüm Sayısı 128 için Üç Seviyeli Şişman Ağaç.

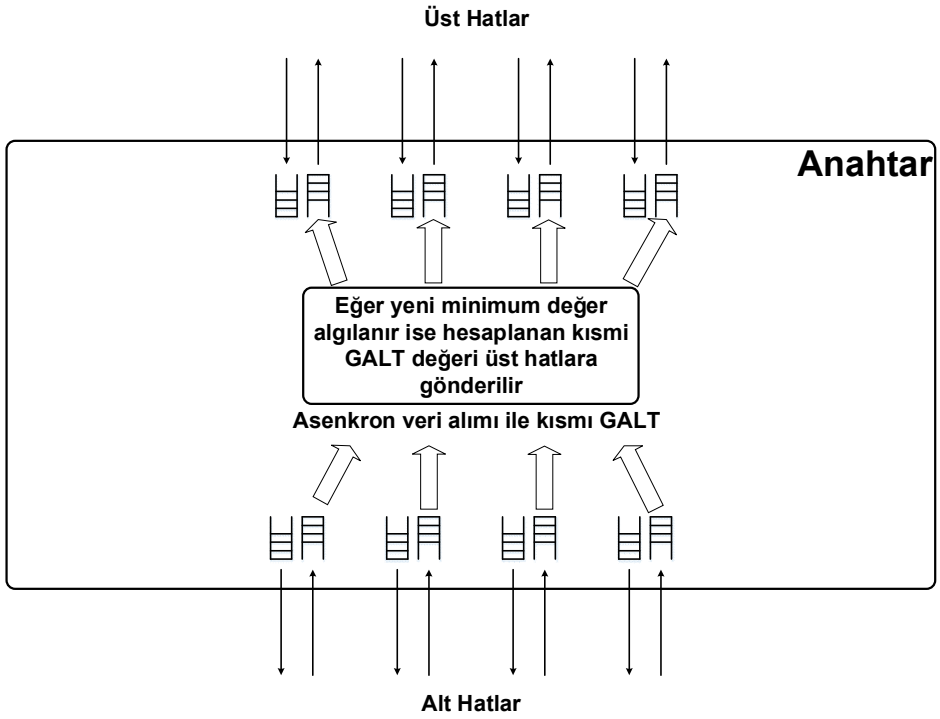
3.2 GALT Algoritması Uygulaması

GALT algoritması uygulama yöntemleri kullanılan anahtara, ağın yapısına veya düğümlerin işlem gücüne göre değiştirilebilmektedir. CS kullanıldığı durumda kullanılabilecek GALT hesabı yöntemlerinden bir tanesi herkese yayın-tabanlı yöntemdir. Bu yöntemde düzenleyici düğümlerden en az bir tanesi zaman ilerletmek için ağdaki düzenleyici tüm düğümlere TAR mesajı gönderir. TAR mesajı tüm düğümlere gönderildiği için gönderilen mesaja herkese yayın-tabanlı zaman ilerletme isteği (İng. Broadcast-Based Time Advance Request, BTAR) mesajı denilir. Bu yöntemde en kötü durum tüm düğümlerin tüm düğümlere BTAR isteğinde bulunduğu yani tüm düğümlerin hem düzenleyici hem de kısıtlı federe olduğu durumdur. Bu durumda bir herkesin-herkese yayını durumu oluşur. BTAR mesajının RTI'a gönderilmesi ile başlayan GALT hesabı RTI'dan TAG mesajı alınması ile bitirilir.

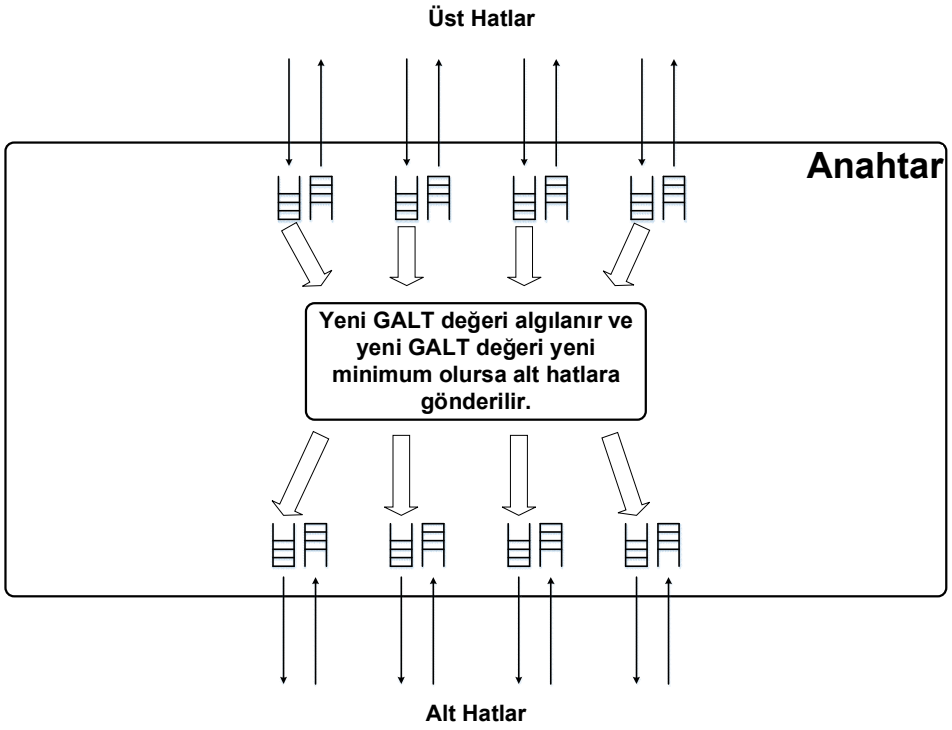
CS kullanıldığı durumda kullanılabilecek GALT hesabı yöntemlerinden bir diğeri ise şişman ağaç-tabanlı yöntemdir. Bu yöntemde ilk olarak her kenar anahtara bağlı düzenleyici düğümlerden her kenar anahtar için bir düzenleyici düğüm seçilir. Seçilen düzenleyici düğümlere aynı kenar anahtara bağlı diğer düzenleyici düğümlerden zaman ilerletme isteği gelir. Seçilen düzenleyici düğümlerde bağlı oldukları kenar anahtar için GALT hesabı yapılır. Daha sonra aynı işlem kapsül seviyesinde gerçekleşir. Her kapsül için seçilen toplayıcı anahtara bağlı kenar anahtar seviyesi seçili düzenleyici düğümler

içinden seçilen bir tanesinde, kenar anahtar seviyesinde seçilen düzenleyici düğümlerde hesaplanan GALT değerleri toplanır. Son olarak yine aynı işlem kapsüller seviyesinde gerçekleştirilir. Kapsüller seviyesinde yapılan işlem sonunda tüm ağ için GALT hesabı tamamlanmış olur. En son seçili kapsülün, seçili toplanma anahtarının seçili kenar anahtarına bağlı düğümde bulunan sonuç tüm ağa herkese-yayın olarak dağıtılır. GALT hesabı yapılırken şişman ağaç yapısına uygun yapıldığı için bu hesabı başlatmak için gönderilen mesaja şişman ağaç-tabanlı zaman ilerletme isteği (İng. Fat Tree-Based Time Advance Request, FTAR) mesajı denilir.

Bu çalışma kapsamında tasarlanan IS kullanıldığında ise GALT algoritması hesabının işlem yükü PE'lerden alınıp Ethernet anahtarına aktarılmaktadır. Sonuç olarak GALT hesabı başlatan her düzenleyici federe ağa TAR mesajı gönderir. Mesajı alan anahtar eğer gelen mesaj alt bağlantılardan geliyor ise daha önce kapılara kayıt edilen kısmi minimum zamanlar (İng. Partial GALT, PG) ile bir karşılaştırma yapar. Eğer yeni minimum değer daha büyük ise yeni minimum değeri PG değeri üzerine yazar. Yeni PG değerini alan çekirdek anahtar değil ise mesajı şişman ağaçta üst bağlantılara iletir. Bu durum Şekil 3.3.a'da gösterilmiştir. Yeni PG değerini alan çekirdek anahtar ise PG değerini GALT hesabı sonucu olarak GALT isimli saklayıcıya kayıt edip tüm alt bağlantılara dolayısıyla da tüm ağa bu değeri dağıtır. Bu değeri üst seviyesinden alan çekirdek anahtar seviyesinin altındaki anahtarlar ise gelen değeri ilgili kapıya ait GALTS isimli saklayıcıya kayıt eder. Eğer tüm üst bağlantı kapılarından gelen GALTS değerlerinin minimumu anahtarın GALT saklayıcısındaki değere eşit değil ise bu durum GALT değerinin güncellendiğini ve güncellenen değerin tüm üst bağlantılardan geldiğini gösterir. Bu durumda ise yeni GALT değeri IS üzerinde saklanır ve Şekil 3.3.b'de gösterildiği gibi tüm alt bağlantılara yeni GALT değeri gönderilir. Tüm üst bağlantılardan gelen değerler güncellenmedikçe alt bağlantılara GALT değeri gönderilmeyeceği için UAV mesajlarının TAR mesajlarından sonra gelmesi engellenir. Sonuç olarak ağda geçiş mesajı kalması sorunu ortadan kaldırılır. Son olarak aşağıda Algoritma 1'de GALT algoritmasının IS üzerinde uygulamasını anlatan sözde kod verilmiştir. TM servisi için geliştirilen bu algoritma, GALT algoritmasının anahtar-tabanlı yaklaşımı olarak adlandırılmıştır.



(a)



(b)

Şekil 3.3. GALT Algoritmasının (a) Alt ve (b) Üst Hatlardan Gelen Mesaj için Anahtar Lojiği.

Algoritma 1. GALT Algoritması Sözde Kodu.

Input: Partial GALT value or the GALT value in an incoming network packet.
Output: Partial GALT value or the GALT value in an outgoing network packet.

```
1:  Set pg ← 0.
2:  Set vals[ ] ← 0.
3:  Set galt ← 0.
4:  Set galts[ ] ← 0.
5:  while forever do
6:     Receive a network packet from a link
7:     if (the packet content is a partial GALT value OR the GALT value) then
8:        switch (the link type)
9:        case downlink:
10:          Set vals[i] with the received value such that i is the incoming link index.
11:          if (pg < min(vals[ ]) ) then
12:             Set pg ← min(vals[ ]).
13:             if (core switch) then
14:                Set galt ← pg.
15:                Send galt to every downlink.
16:             else
17:                Send pg to every uplink.
18:             end if
19:          end if
20:          break;
21:       case uplink:
22:          Set galts[i] with the received value such that i is the incoming link index.
23:          if (galt ≠ min(galts[ ]) ) then
24:             Set galt ← min(galts[ ]).
25:             Send galt to every downlink.
26:          end if
27:          break;
28:       end switch
29:    else
30:       // …
31:       // Handling the other network packet types
32:       // …
33:    end if
34: end while
```

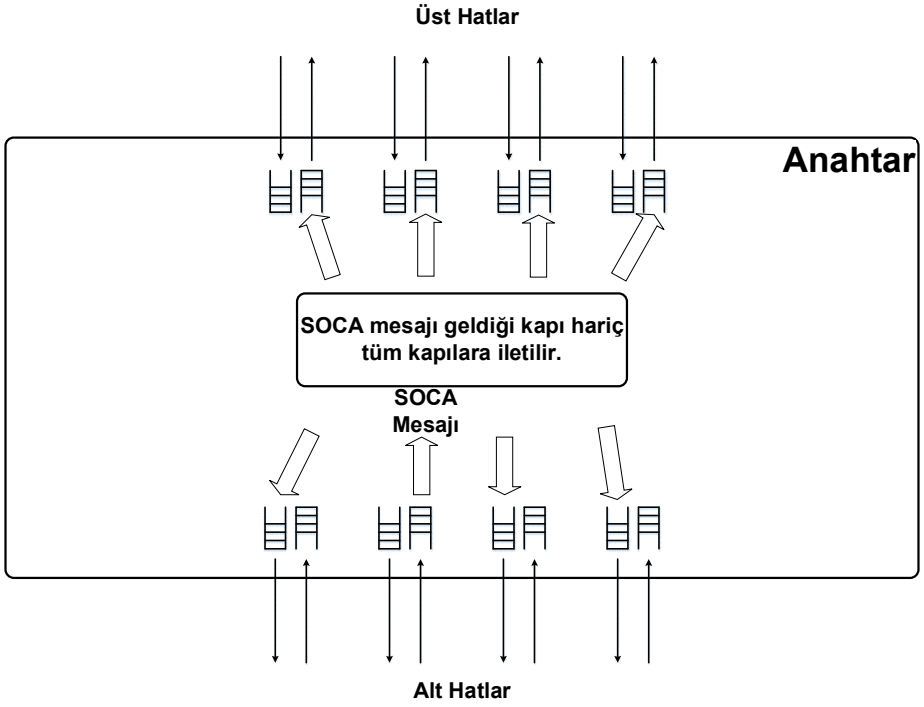
3.3 Eşleştirme Algoritması Uygulaması

Bu çalışmada DDM servisinin bölge-tabanlı yaklaşımında kullanılan eşleştirme algoritması IS'ye hızlandırıcı olarak eklenmiştir. CS kullanıldığı durumda tüm abone federeler, abonelik isteklerini herkese-yayın mesajı olarak ağa iletir. Herkese-yayın olarak ilerleyen SOCA mesajı içerisinde; sınıf kimliği, özelik vektörü, her eksen için başlangıç ve bitiş noktalarını içeren bölge bilgileri bulunmaktadır. Bu mesajı alan yayıncı düğümler abone bilgilerini kayıt ederler. Yayın yapacak düğüm yayın yapmadan önce yayın mesajı ile bölge mesajını eşleştirir. Eğer ilgili abone mesajları ile yayın mesajı arasında eşleşme var ise mesaj doğrudan kesişim olan abonelere gönderilir. Böylece yayın yapan düğümün gönderdiği UAV mesajı ağda ilgili düğümlere doğru ilerler.

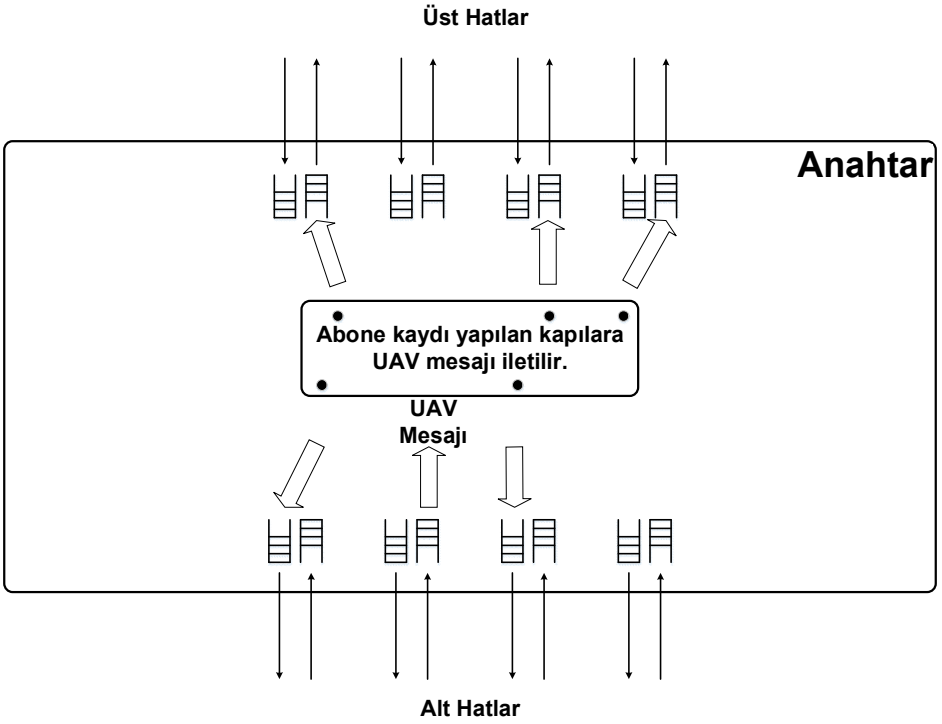
Abone mesajı hem CS hem de IS'de Şekil 3.4.a'da gösterildiği gibi herkese-yayın mesajı olarak ilerlese de IS'de mesaj içeriği anahtar üzerindeki saklayıcılara kayıt edilir. Bu sayede mesajın alıcı düğümlere iletilmesine ihtiyaç kalmamaktadır. Yayın mesajlarında IS kullanıldığı durumda, PE'lerde koşan eşleştirme algoritması anahtar üzerine aktarıldığı için bir iyileşme elde edilmiştir. IS eşleşme bulunan kapılara, yani ilgili abonelere Şekil 3.4.b'de gösterildiği gibi UAV mesajlarını ulaştırır.

Eşleştirme algoritması IS'de çalışması esnasında ilk olarak mesajın tipini kontrol eder. Eğer gelen mesaj SOCA mesajı ise, mesajın aşağı ve yukarı bağlantıdan gelme durumu kontrol edilir. Yukarı bağlantıdan gelen mesajlar tüm aşağı kapılara iletilirken abone mesajı içeriği (sınıf/etkileşim kimliği, özelik vektörü ve bölge koordinatları) IS üzerindeki saklayıcılara kayıt edilir. Aşağıdan gelen mesajlarda ise yine IS üzerindeki saklayıcılara kayıt yapılırken mesajın geldiği kapı hariç tüm alt ve üst kapılara mesaj iletilir.

Eşleştirme algoritması mesaj tipini UAV olarak algılar ise yukarı bağlantılardan gelen mesajlar için aşağı bağlantılar ile bölge kesişimi bakılır ve kesişim durumunda kesişim olan kapıya mesaj iletilir. Bu şekilde tüm aşağı bağlantılar taranır. Eğer UAV mesajı bir aşağı bağlantıdan geliyor ise mesajın geldiği bağlantı hariç eşleşme bulunan alt bağlantılara yayın mesajı iletilir. Çekirdek anahtar hariç anahtarlarda yukardan gelen SOCA mesajları yığınında eşleşme bulunursa, rastgele olarak ilk üst bağlantı kapısından UAV mesajı gönderilir. Aşağıda Algoritma 2'de eşleştirme algoritmasının IS üzerinde uygulamasını anlatan sözde kod verilmiştir. DDM servisi için gerçekleştirilen bu algoritma, eşleştirme algoritmasının anahtar-tabanlı yaklaşımı olarak adlandırılmıştır.



(a)



(b)

Şekil 3.4. Eşleştirme Algoritmasının (a) SOCA ve (b) UAV Mesajları için Anahtar Lojiği.

Algoritma 2. Eşleştirme Algoritması Sözde Kodu.

Input: Class/interaction id (c), attribute vector (v), and region coordinate (r) values in an incoming network packet.

Output: Class/interaction id, attribute vector, and region coordinate values in an outgoing network packet.

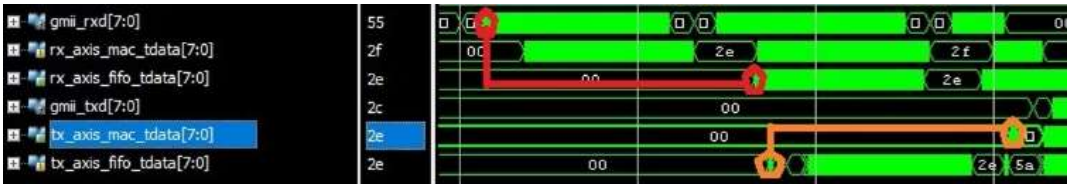
```
1:  Set ul_list ← Ø.
2:  Set dl_list[ ] ← Ø.
3:  while forever do
4:     Receive a network packet from a link
5:     if (the packet content "subscribe object class attributes with region" message) then
6:        if (message from uplink) then
7:           ul_list.add_unique((c, v, r))
8:           Send received message to every downlink
9:        else
10:          dl_list[i].add_unique((c, v, r))
11:          Send received message to every downlink except for downlink i
12:          if (not core switch ) then          // The core switches have no uplinks.
13:             Send received message to every uplink
14:          endif
15:       endif
16:    else
17:       if (the packet content "update attribute value" message) then
18:          if (message from uplink) then
19:             for every downlink i do
20:                if(dl_list[i].find_match((c, v, r))) then
21:                   Send received message to downlink i
22:                end if
23:             end for
24:          else
25:             if (not core switch ) then          // The core switches have no uplinks.
26:                if (ul_list.find_match((c, v, r))) then
27:                   Send received message to an arbitrary uplink
28:                end if
29:             end if
30:             for every downlink i do
31:                if (i is NOT incoming port AND dl_list[i].find_match((c, v, r))) then
32:                   Send received message to downlink i
33:                end if
34:             end for
35:          end if
36:       else
37:          // …
38:          // Handling the other network packet types
39:          // …
40:       end if
41:    end if
42: end while
43:    //  list.add(3-tuple)              Add the tuple to the list
44:    //  list.add_unique(3-tuple)       Add the tuple to the list if the tuple is not already there
45:    //  list.find_match(3-tuple)       Search the tuple in the list if exist return true
```

4. MALİYET FONKSİYONU ANALİZLERİ

Maliyet fonksiyonları hesaplama ve haberleşme olmak üzere iki kısımdan oluşmaktadır. Hesaplama maliyeti GALT ve eşleştirme algoritmalarının tanımladığı işlemleri gerçekleştirmek için harcana süre iken, haberleşme maliyeti ağ üzerinde mesajların aktarılması esnasında harcanan süredir. Bu bölümde anahtar-tabanlı yaklaşım ile geliştirilen TM servisinin GALT ve DDM servisinin eşleştirme algoritmalarının IS uygulamalarındaki hesaplama ve haberleşme maliyet fonksiyonları ağda trafik olmadığı durum için çıkarılmıştır. Ayrıca bu çalışma sonunda paralel ayrık olay benzetimlerinde elde edilecek olan performans iyileştirmesini değerlendirebilmek için CS kullanılarak uygulanan GALT ve eşleştirme algoritmalarının da maliyet fonksiyonları çıkartılmıştır.

Bu maliyet fonksiyonun çıkarılabilmesi için ilk olarak IS ve şişman ağaçtaki gecikmelerin analiz edilmesi gerekmektedir. Bunun için ilk olarak Xilinx'in Trimode ortam erişim kontrolcüsü (İng. Media Access Controller, MAC) tasarımının Vivado programında benzetimi yapılmıştır. Şekil 4.1'de benzetim sonuçları verilmiştir. Benzetim sonuçlarına göre RX tarafında MAC IP'si yardımı ile gigabit ortam bağımsız arayüz (İng. Gigabit Media Independent Interface, GMII) arayüzünden alınan veri (gmii_rxd), MAC'in çıkışına (rx_axis_mac_tdata) veri alınmasından hemen sonra yazılmaktadır. Fakat Gelişmiş genişleyebilir akış arayüzü (İng. Advanced eXtensible Interface Stream, AXIS) arayüzünden alınan verinin (rx_axis_fifo_tdata) anahtarlama için verilmesi RX ilk giren ilk çıkar (İng. First In First Out, FIFO) hafızasının MAC tarafından tüm paketin alınmasından sonra gerçekleşmektedir. Tüm paket alındıktan sonra alınan pakette bir hata bulunursa (taşma gibi) paket düşürülür. RX FIFO'da taşma durumu, alıcı saat frekansı gönderici saat frekansından fazla ise ya da alıcı tarafında beklenen çerçeveler arası boşluk gönderme tarafından büyükse olur. Özetle alıcı ve verici tarafında tüm paket FIFO'da saklanıp hata ayıklandıktan sonra PE'ye iletilir ya da PE'den MAC'a iletilir. Yani alan tarafında önce GMII arayüzünden tüm veri alınıp sonra PE'ye aktarılır. Ethernet anahtarı mimarisinde depola ve gönder mimarisi seçildiği için taşma veya hatalı çerçeve durumunda çerçeve düşürülebilmekte ve işlemciye hiç aktarılmamaktadır. Bu şekilde hatalı çerçeveler ile PE yalıtılmış ve güvenilir haberleşme sağlanmış olur. Şekil 4.1'deki gecikme bilgileri maliyet fonksiyonları hesaplanırken kullanılacaktır. Örneğin bir paketin ağ anahtarına MAC tarafından girdiği durumda: Paket RX FIFO'nun MAC tarafında yazılması $L\tau$ kadar zaman alacaktır. Burada L mesaj boyutu ve $1/\tau$ hat bant genişliğidir.

Daha sonra bu paketin RX FIFO'nun PE tarafından ağ anahtarı çekirdeğine girip daha sonra ağ anahtarı çekirdeğinden ilgili çıkış kapısına yönlendirilmesi ise anahtarlama gecikmesidir ve trafik olmadığı durumda paketin hafızaya yazılmasına göre çok kısa sürmektedir. Yine Şekil 4.1'de gösterildiği üzere TX FIFO'dan MAC tarafına iletilen paketin FIFO'nun MAC tarafını terk edip GMII arayüzünden gönderilmesi Lτ kadar zaman alacaktır. Burada yarım çift yönlü (İng. Half duplex) haberleşme durumunda defalarca çakışma (İng. Collision) olsa bile veri TX FIFO'dan tekrar tekrar çekilip gönderilebilmektedir. Aslında Ethernet anahtarının yaygın kullanımında tam çift yönlü (İng. Full duplex) haberleşme tercih edildiği ve alıcı tarafındaki hafızalar küçük parçalara bölünerek aynı anda okumayı desteklediği için TX tarafında ayrı bir tamponlama ihtiyacı yoktur. O durumda TX FIFO gecikmesi olmayacaktır. Fakat bu çalışmadaki mimaride çıkış tarafında böyle bir tamponlama olduğu için Lτ kadar olan bu gecikme anahtarlama gecikmesinin parçası olarak tüm analizlerde kullanılmıştır.

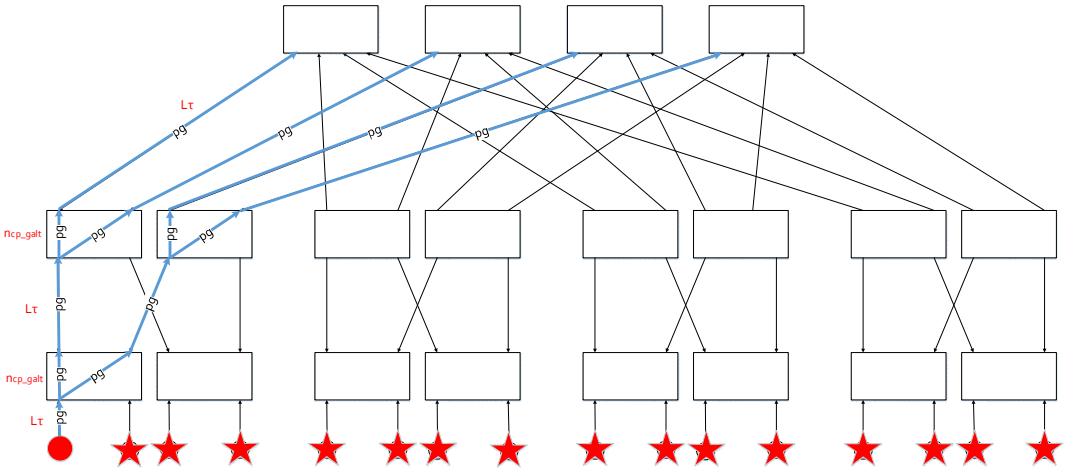


Şekil 4.1. GMII Arayüzü ile MAC üzerinden Paketin Alınıp-Gönderilmesi.

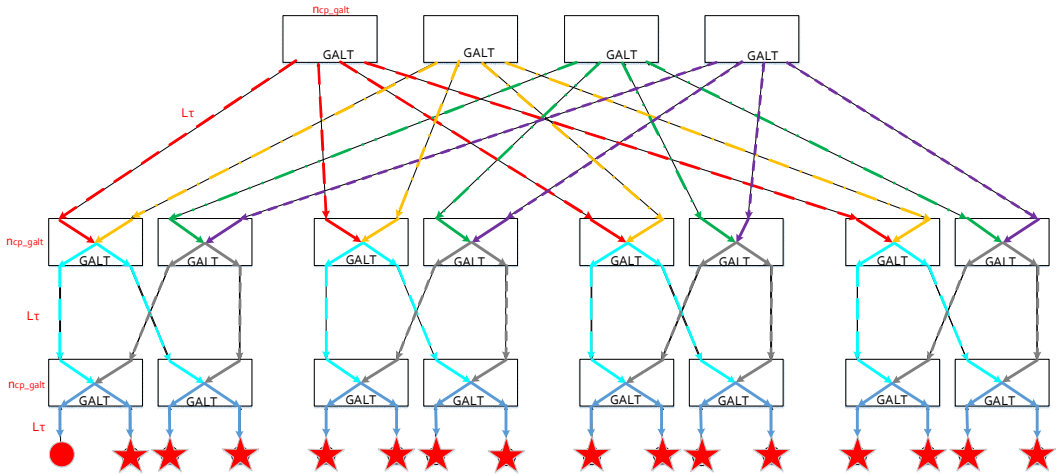
Maliyet fonksiyonda kullanılan bir diğer parametre de mesaj ilklendirme gecikmesidir ve bu çalışmada üçe ayrılmıştır. İlki mesajın PE tarafından gönderilmesi için geçen süre toplamıdır (α_1). İkincisi PE'ler mesaj ile işlem (GALT veya eşleştirme algoritması) yaparken geçen süredir (α_2). Üçüncüsü ise mesajın PE tarafından alınması için geçen süredir (α_3). Yayın/abone sisteminde IS'de abone mesajının düğüme çıkarılma ihtiyacı olmadığı için abone mesajı için mesaj alınma zamanı yoktur (α_3=0). GALT ve eşleştirme algoritması için anahtar gecikmesi daha önce belirtildiği gibi ayrı ayrı ölçülmüştür. Saat periyodu cinsinden bir değerindedir. GALT algoritması için 25 ve eşleştirme algoritması için 118 saat periyodu ölçülmüştür. Bu değerler kapı sayısından bağımsızdır. Anahtarlama gecikmesine ayrıca yukarda bahsedilen TX FIFO gecikmesinden dolayı Lτ kadar bir gecikme eklenmektedir. Tasarlanan ağ daha önce bahsedildiği gibi 3 seviyeli şişman ağaç topolojisindedir. Ağdaki düğüm sayısı N ile gösterilmiştir. N değeri ayrıca $k^3/4$ olarak da gösterilir. Burada k değeri daha önce belirtildiği gibi kullanılan Ethernet anahtarının kapı sayısıdır. Ethernet anahtarı mesajın tamamını paket olarak almakta ve paket olarak iletmektedir. Ağda trafik olmadığında bir paketin anahtara girip çıkması Lτ artı anahtarlama süresi kadar zaman almaktadır.

4.1 Anahtar-Tabanlı GALT Algoritması

Bölüm 3.2'de sözde kodu verilen anahtar-tabanlı GALT algoritmasında PG değerinin hesaplanması Şekil 4.2'de gösterilmiştir. Burada daire sembolü ile gösterilen düğüm GALT değerini güncellemektedir. İşlemler TAR mesajı ile başlar. Güncel GALT ağda çekirdek anahtara doğru PG değeri olarak ilerlemektedir. Şekil 4.3'de çekirdek anahtarlara ulaşan PG değerinin yeni GALT değeri olduğunda tüm düğümlere iletilmesi gösterilmiştir. Çekirdek anahtardan tüm düğümlere iletilen GALT değeri toplanma ve kenar anahtarlarda ilerlerken tüm üst hatlardan GALT değerinin gelmesi beklenir. Böylece ağda kalan geçiş mesajları engellenmiş olur.



Şekil 4.2. TAR için PG Değeri Hesabı.



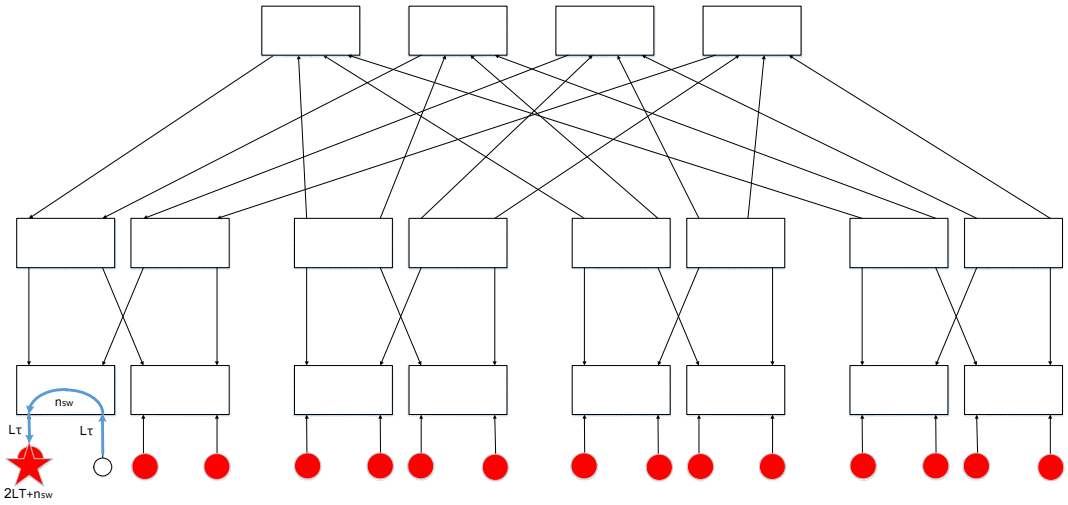
Şekil 4.3. TAR için GALT Değerinin Düğümlere İletilmesi.

IS'de tasarlanan GALT algoritmasında ilk PE'den paket alındığında RX FIFO'nun MAC tarafında Lτ süresi geçecektir. Paketin anahtar üzerinden işleme uğrayıp tekrar ilgili çıkış tamponuna yazılması içinde anahtarlama süresi (n_{cp_galt}) geçecektir. Daha sonra ilk seviyedeki anahtardan bir üst seviyedeki anahtara paket gönderilirken bir üst seviyedeki anahtarın RX FIFO'suna MAC tarafına paket yazılırken Lτ kadar daha zaman geçecektir. Sonra diğer anahtarlarda yine aynı şekilde ilerleyecektir. Her hatta 1 Lτ ve her anahtarda anahtarlama süresi kadar gecikme olmaktadır. Toplamda 6 hat ve 5 anahtar olduğu için 6Lτ ile $5n_{cp\_galt}$ toplamı kadar gecikme olacaktır. Anahtar gecikmesi ise aslında hesaplama ve TX FIFO'ya veri aktarma gecikmesidir. Her anahtardaki anahtar gecikmesinde bulunan hesaplama gecikmesi $25t_{clk}$ olur. Saat periyodu 8ns ve toplam 5 anahtar için 1us olarak hesaplanır. Anahtar gecikmesi bu gecikmeye ilave 5Lτ veri aktarma gecikmesi oluşmaktadır. Mesajlar ağda ilerlerken anahtarlardaki tamponlama gecikmesi hariç başka bir haberleşme gecikmesine uğramayacaktır. Bu durumda IS ile kullanılan anahtar-tabanlı GALT algoritması hesabı için maliyet fonksiyonu 1.1'de verilmiştir.

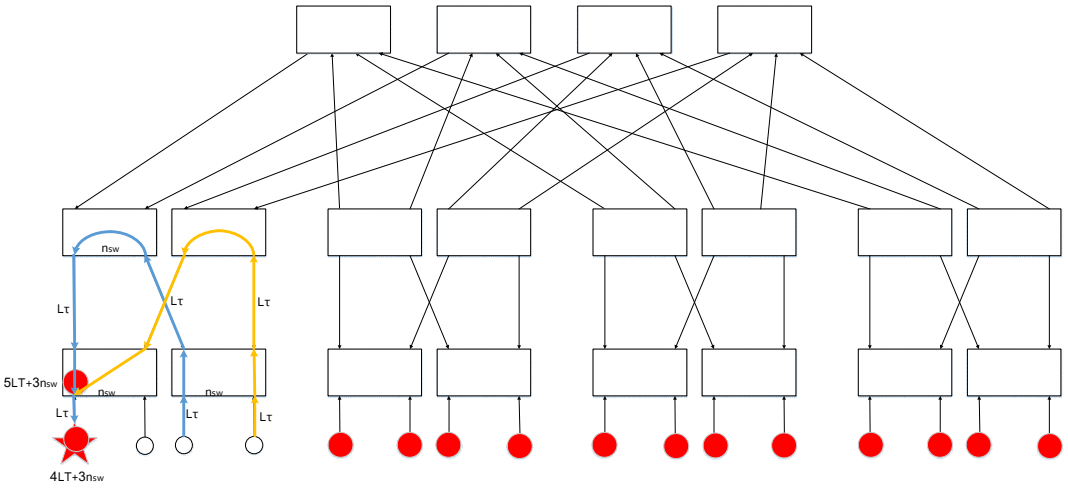
$$T = \alpha_1 + \alpha_3 + 6L\tau + 5n_{cp\_galt} \tag{1.1}$$

4.2 Herkese Yayın-Tabanlı GALT Algoritması

IS kullanılmadığı yani CS kullanıldığı durumda GALT hesabında iki yöntem izlenebilecektir. İlk yöntemde tüm düzenleyici federeler tüm kısıtlı federelere zaman bilgisini gönderip GALT hesabının her kısıtlı federenin kendisi tarafından hesaplandığı durumdur. Bu yöntem daha önce bahsedilen herkese-yayın tabanlı hesaplama yöntemidir. İşlemler BTAR mesajı ile başlar. Bu yöntemde tüm düğümler birbirlerine GALT değerlerini gönderirler. Şekil 4.4'de bu yöntemde gerçekleşen ilk adım gösterilmiştir. Yıldız sembolü ile gösterilen düğüm daire sembolü ile gösterilen düğümlerden GALT değeri alan düğümdür. İlk GALT değerini ilk adımda aynı kenar anahtara bağlı olduğu düğümden almıştır. Çizimde gösterilen n_{sw} değeri anahtarlama gecikmesidir. CS kullanıldığı durumda değeri 1 Lτ'dir. Şekil 4.5'de ikinci adımda aynı kapsüldeki düğümlerden toplanma anahtarı üzerinden mesaj gelmektedir. Son olarak Şekil 4.6'da ilk komşu kapsüldeki düğümlerden gelen mesaj gösterilmiştir. Diğer komşu kapsüllerin mesajları ise şekiller üzerinde gösterilmemiş olup, Şekil 4.6'ya benzer şekilde davranacaktır. Alıcı düğüme bağlı kenar anahtarın tamponunda tüm mesajlar sıraya girecektir.

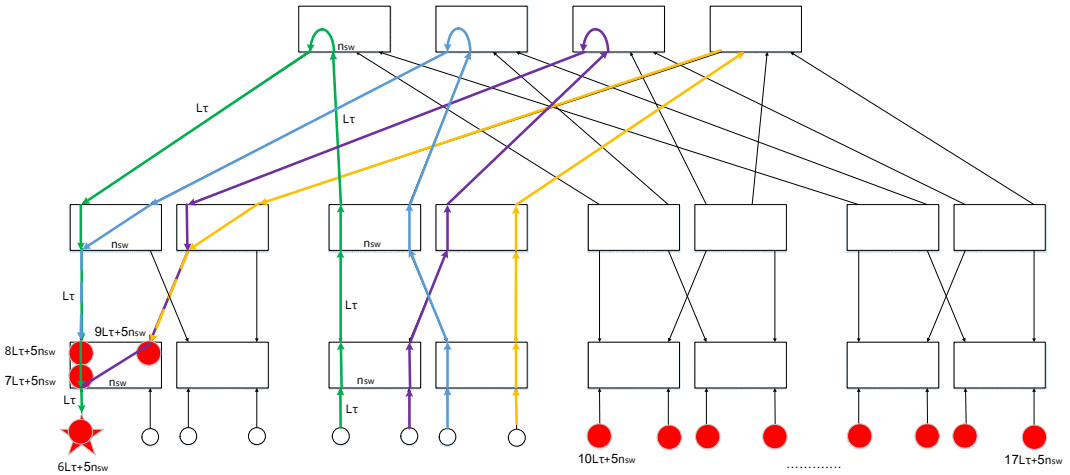


Şekil 4.4. BTAR için Aynı Kenar Anahtara Bağlı Düğümlerden Mesaj Gelmesi.



Şekil 4.5. BTAR için Aynı Kapsüle Bağlı Düğümlerden Mesaj Gelmesi.

Ağ topolojisi olarak 3 seviyeli şişman ağaç kullanıldığı için aynı kenar anahtara bağlı düğümlerden seçili bir tanesine diğer düğümden gelen mesajın gelme süresi en erken 3Lτ'dir. Bunun 2Lτ'si hat 1Lτ'si ise anahtarlama gecikmesidir. Aynı toplanma anahtarına bağlı düğümlerden gelen mesajların seçili düğüme gelme süresi en erken 7Lτ'dir. Diğer kapsüllerden gelen mesajların seçili düğüme gelme süresi ise en erken 11Lτ'dir. Herkesin-herkese yayınında aynı kenar anahtardaki düğümlerin sayısı dörtten büyük olursa kenar anahtar gecikmesi daha baskın olup 3Lτ'lik gecikmenin ardından diğer tüm düğümlerdeki mesajlar sıra ile gelecektir. Son mesaj ise 6Lτ'de gelecektir. Aynı kapsülden gelen ilk mesaj 7Lτ'de geleceği için arada boşta geçen zaman olmayacaktır.



Şekil 4.6. BTAR için Komşu Kapsüle Bağlı Düğümlerden Mesaj Gelmesi.

Herkese-yayın tabanlı GALT hesabında herkesin-herkese yayını durumu oluştuğu için ilk başta mesaj veri işleme hattının (İng. pipeline) dolması gerekmektedir. 8 kapılı bir anahtarda her kenar anahtara 4 düğüm bağlı olacağı için ilk mesajın 3Lτ'lik gecikmesinden sonra sırası ile 2. ve 3. mesajlar 4Lτ, 5Lτ sonra gelecek, yani ilgili düğüm tarafından alımı tamamlanacaktır. Aynı toplanma anahtarına bağlı bir sonraki paket ise 7Lτ'de gelecektir. Böylece boşa geçen zaman sadece 1Lτ olacaktır. Aynı kapsüldeki zaman 11Lτ'yi geçeceği için diğer kapsülden gelen mesajlar veri işleme hattında sıraya girecek ve kayıpsız olarak iletilebilecektir. Bu yüzden anahtar kapı sayısı 8'den büyük olan anahtarlarda 1 Lτ gecikmede olmayacağı için herkesin-herkese yayın maliyeti 3Lτ gecikme üzerine Lτ(N-2)'dir. Toplamda ise Lτ(N+1) olacaktır. Sonuç olarak herkese-yayın tabanlı GALT hesabında maliyet fonksiyonu CS kullanıldığında kapı sayısı sekizden büyük için aşağıdaki 1.2'deki gibidir. Kapı sayısı sekiz ve sekizden küçük olursa veri hattını doldurmak için ortaya çıkan gecikmeler formülde ele alınmamıştır ve ağ boyutu büyüdükçe ihmal edilebilir bir değerdir. Ethernet anahtarı kapı sayısı sekiz için boşta geçen zaman 1 Lτ olarak yukardaki gibi anlatılmıştır. Kapı sayısı sekizden küçük olursa diğer anahtarlarda ise maliyet fonksiyonu şu şekilde bulunur. Kapı sayısı 4 için ikinci kapsülden ilk mesaj 11Lτ'de gelmektedir. İkinci kapsülden ilk düğümden itibaren tüm mesajların gelmesi ise kalan 11 düğümden dolayı 11Lτ olmaktadır. Toplamda ise 22Lτ'lik bir mesaj gecikmesi olacaktır. 1.2 formülüne göre gecikme hesaplandığında N=16 için 17Lτ çıkması gerekirken sonucun 22Lτ çıkmasının sebebi ise 3Lτ ile 6Lτ ve 7Lτ ile 11Lτ arasındaki boşlukta mesaj gelmemesidir. Benzer şekilde 6 kapılı anahtar için bu gecikme 55Lτ olması gerekirken 57Lτ'lik bir gecikme çıkmasının sebebi de aynı kenar

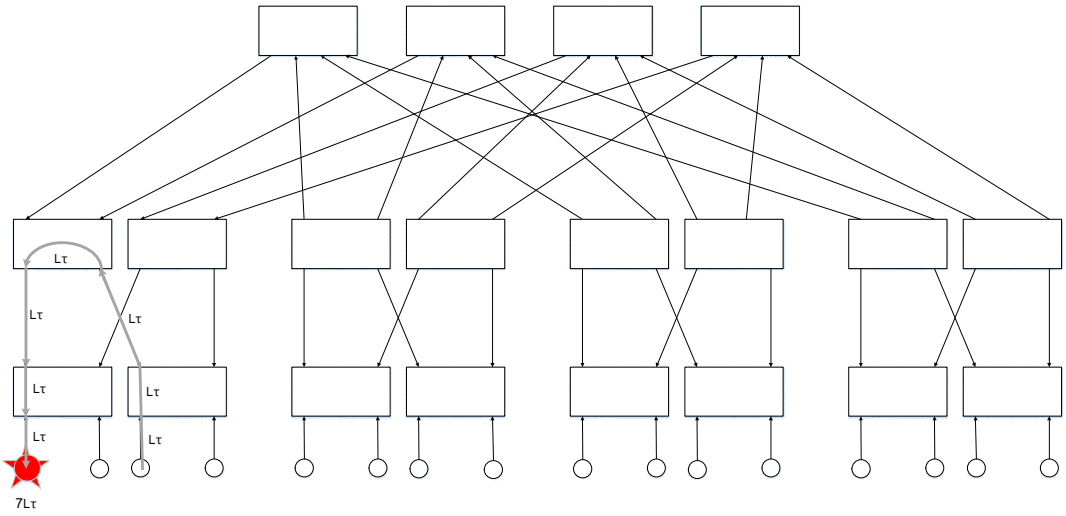
anahtara bağlı 3. düğümden gelen mesajın 4Lτ'de gelmesinden sonra diğer kenar anahtardan gelen mesajın 7Lτ'de gelmesi ile arada yaşanan 2Lτ'lik kaybın sonucudur.

$$T = \alpha_1 + \alpha_2 + \alpha_3 + L\tau(k^3/4+1) \quad (1.2)$$

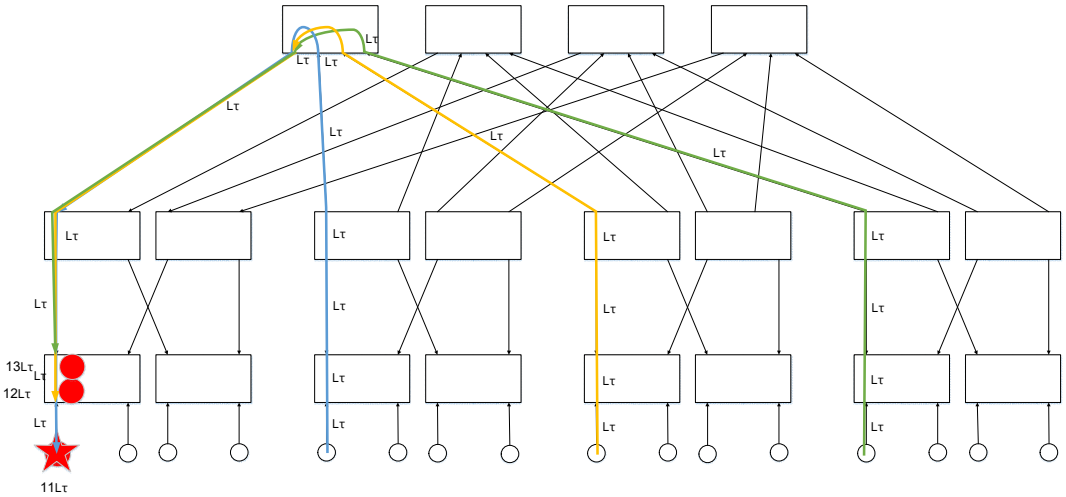
4.3 Şişman Ağaç-Tabanlı GALT Algoritması

CS kullanıldığı durumda GALT hesabında kullanılacak bir diğer yöntem ise yine daha önce bahsedilen şişman ağaç-tabanlı hesaplama yöntemidir. Bu yöntemde ilk olarak aynı kenar anahtara bağlı olan düzenleyici düğümler seçili düğüme FTAR mesajı ile GALT hesabı isteğinde bulunurlar. Daha sonra aynı kapsüldeki kenar anahtarların seçili düğümleri aynı kapsüldeki seçili toplanma anahtarı üzerinden, kapsül seviyesinde seçilen düğüme, hesapladıkları GALT değerini gönderir ve kapsüle ait GALT değeri bulunur. Sonra yine kapsüller seviyesinde seçilen kenar anahtara bağlı düğümler, ağ seviyesinde seçilen düğüm ile haberleşir ve ağın GALT değeri bulunur. Son olarak bulunan GALT değeri tüm ağa herkese-yayın olarak dağıtılır. Yine bu yöntemde ilk adım tüm kenar anahtarlarda Şekil 4.4'deki gibi olacaktır. Kenar anahtar seviyesi seçili düğüme 3Lτ süresinde ilk mesaj iletecek ve bu düğümlerde bir GALT hesabı yapılacaktır. Aynı şekilde komşu kenar anahtarlardan seçilen toplanma düğümü üzerinden gelen bulunmuş GALT değeri ise Şekil 4.7'de gösterildiği gibi her kapsülde seçili düğüme 7Lτ sonra ulaşacaktır. Yine GALT hesabı iki düğüm için yapılıp kapsül içinde GALT değeri bulunmuş olacaktır. Diğer kapsüllerin seçili düğümlerinden gönderilecek mesajlar Şekil 4.8'de gösterildiği gibi sırası ile 11Lτ, 12Lτ ve 13Lτ'de ulaşacaktır. Sonrasında ise bu 4 değerin minimumu bulunup Şekil 4.9'da gösterildiği gibi GALT değeri tüm ağa herkese-yayın olarak iletilecektir. Toplamda mesaj gecikmesi ise 3Lτ, 7Lτ, 13Lτ ve 11Lτ'nin toplamı olan 34Lτ'dir.

Mesaj hesaplama maliyeti ise ilk 2 adımda 2 düğüm (k/2) sonraki adımda 4 düğüm (k) için yapılmakta ve son adımda hiç yapılmamaktadır. Toplamda ise 2k eleman için yapılmaktadır. Diğer düğüm sayılarında ise farklı olarak kenar anahtara bağlı düğüm, kenar anahtar ve kapsül sayısına bağlı olarak mesaj gecikmesi artmaktadır. Mesaj ilkelendirme gecikmesi ise sabit olarak mesaj ilklendirme süresinin 4 katı alınmaktadır.



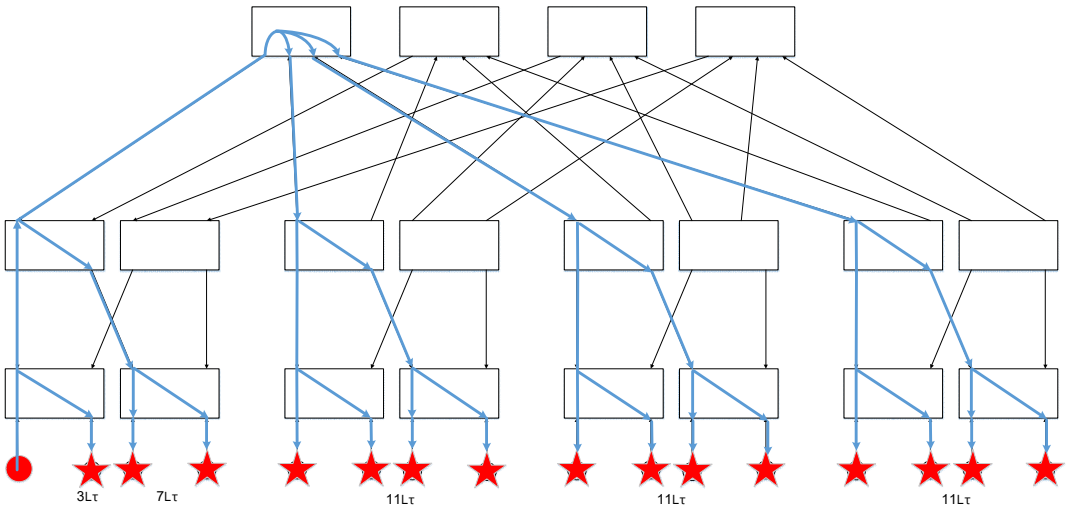
Şekil 4.7. FTAR için Komşu Kenar Anahtara Bağlı Düğümden Mesaj Gelmesi.



Şekil 4.8. FTAR için Komşu Kapsüllerden Mesaj Gelmesi.

Düğüm sayısı 54 olan bir ağda her kenar anahtara 3 düğüm bağlanmakta ve her kapsülde 3 kenar anahtar olmaktadır. Toplamda ise 6 kapsül olmaktadır. Bu durumda kenar anahtara bağlı düğüm sayısı 1 artığı için gecikme 3Lτ yerine 4Lτ olacaktır. Kenar anahtar sayısı da 1 arttığı için gecikme 7Lτ değil 8Lτ olacaktır. Aynı şekilde kapsül sayısı da 4 yerine 6 olduğu için son mesaj 13Lτ'de değil de 15Lτ'de gelecektir. Sonuçta en sonda yapılan herkese-yayın ile birlikte toplam mesaj gecikmesi 38Lτ (4+8+15+11) olacaktır. 128 düğümlü örnekte de sırası ile aynı kenar anahtara (4 düğüm), toplanma düğümü üzerinden komşu kenar anahtara (4 kenar anahtar) ve farklı kapsüle (8 kapsül) bağlı düğümlerin gecikmesi 5Lτ, 9Lτ ve 17Lτ olacaktır. Bu durumda ise toplamda 42Lτ'lik bir

gecikme olacaktır. Yine tüm düğüm sayıları için mesaj ilklendirme süresi, 1 mesaj ilklendirme süresinin 4 katı olacaktır.



Şekil 4.9. Şişman Ağaç-Tabanlı GALT Algoritması GALT değerinin Dağıtılması.

Bu yapıda tüm düğümlerden GALT değeri gelmesine gerek kalmadan her kenar anahtar kendi içinde hesap yapıp sonra bu işlem toplanma anahtarı ve kapsül seviyesinde şişman ağaç mimarisine uygun olarak tekrar ettiği için hesap yükü genişleyen ağlarda oldukça azalmaktadır. Formül 1.3'de şişman ağaç-tabanlı yaklaşım ile GALT hesabının maliyet fonksiyonu verilmiştir. Burada k daha önce bahsedildiği gibi kapsül sayısı olup 4, 6 ve 8 değerini almaktadır. Yine α_2 değeri 2k kadar PE için GALT algoritması süresi ölçülerek maliyet fonksiyonlarında kullanılabilmektedir.

$$T = 4(\alpha_1 + \alpha_3) + \alpha_2 + (3+(k/2-2))L\tau + (7+(k/2-2))L\tau + (11+(k-2))L\tau + 11L\tau$$

$$T = 4(\alpha_1 + \alpha_3) + \alpha_2 + (26+2k)L\tau \quad (1.3)$$

Sadece GALT algoritması için düğümlerde harcanan α_2 süresi ile IS'de harcanan süreyi karşılaştırmak mümkündür. ARM A9 kullandığımızda testlerde, 64B mesaj uzunluğu ve 16 düğümün GALT değeri yaklaşık 1.1µs'de hesaplanmıştır. IS kullanıldığında ise anahtarda GALT algoritması için harcanan süre $5n_{cp\_galt}$ olduğu için toplamda 1µs harcanmaktadır. Fakat düğüm sayısı 2000'e çıkacak olursa IS'de harcanan süre yine 1µs iken CS'de herkese yayın-tabanlı yaklaşım için 469.8µs'ye, şişman ağaç-tabanlı yaklaşım için ise 229µs'ye çıkmaktadır.

4.4 Anahtar-Tabanlı Eşleştirme Algoritması

Yayın/abone sisteminde abone mesajları ağda herkese-yayın olarak ilerlemektedir. IS üzerinde abone mesajlarının kayıt edildiği için abone mesajlarının PE'lere iletilme ihtiyacı yoktur. Sağlanan bu iyileşmenin haricinde abone mesajları ile ilgili başka bir iyileştirme çalışması yapılmadığı için abone mesajları ile ilgili bir maliyet fonksiyonu çıkarılmamıştır. Eşleştirme algoritmasının çalıştığı yayın mesajları ile ilgili bir maliyet fonksiyonları çıkarılmıştır. Eşleştirme algoritmasının maliyet fonksiyonunda en kötü durum ağdaki tüm düğümlerin hem yayıncı hem de aynı yayına abone olma ve abone istekleri ile yayın mesajlarının eşleştiği durumdur. Bu durum aslında bir herkesin-herkese yayın durumudur. Herkesin-herkese yayın durumunu Şekil 4.4, Şekil 4.5 ve Şekil 4.6'da gösterildiği şekilde gerçekleşecektir. Anahtar-tabanlı yaklaşımda n_{sw} değeri 1 $L\tau$ ile eşleştirme algoritmasının anahtarda uygulanması için geçen sürenin toplamı olacaktır. Bu bölümde çıkarılan maliyet fonksiyonunda n_{cp_match} olarak gösterilmiştir. Anahtar-tabanlı yaklaşımda eşleştirme algoritması çalışırken; ilk olarak tüm düğümler mesajlarını ağa iletecektir. En zorlayıcı durumda tüm düğümlerden gönderilen mesajlarda eşleşme olduğu için her gelen mesaj kenar düğümden tüm düğümlere gönderilir. Böylelikle bir düğümün ilk mesaj alma süresi $2L\tau$ ile anahtarlama süresi toplamı olacaktır. Bu esnada geriye alınacak mesaj sayısı ise N-2'dir. Sonuç olarak tüm mesajlar alındığında toplam geçen süre $NL\tau$ ile anahtarlama süresinin toplamı olacaktır. Kenar anahtara bağlı tüm düğümlerin gönderdiği mesajlar kenar anahtara bağlı düğümlere sıra ile gelecektir. Bu esnada tüm kenar anahtarların toplanma anahtarlarına mesajlar iletilecek ve gelen mesajlar yine sıra ile en üstteki çekirdek anahtarlara gönderilecektir. Çekirdek anahtarlarda biriken mesajlarda yine sıra ile toplanma ve kenar anahtarlara gönderilecektir. Kenar anahtarlar kendilerine bağlı düğümler ile iletişimi başlattığı için toplanma ve çekirdek anahtar üzerinden gelen mesajlar yeterli tampon olduğu durumda yine bu anahtarda sıraya girecektir. Sonuç olarak yeterli büyüklükteki ağda her düğüm mesaj alma ve anahtarlama gecikmesinden sonra bu veri işleme hattından sırayla diğer tüm düğümlerin mesajlarını alacaktır. Her anahtarda eşleştirme algoritması sonucuna göre mesajlar iletilecektir. Eşleştirme algoritması işlemi gecikmesi IS üzerinde 118 saat darbesi olarak ölçülmüştür. Eşleştirme algoritmasında anahtarlama gecikmesi n_{cp_match} ise, IS için donanım işlem gecikmesi olan 0.94us (118 x t_{clk}) üzerine gelen $1L\tau$'lik tamponlama gecikmesinden oluşmaktadır. Ayrıca herkesin-herkese yayın durumunda ilk komşu düğümden gelen mesajda bu anahtar gecikmesi yaşanırken daha sonra gelen

mesajlar veri işleme hattında sıraya gireceği için maliyet fonksiyonunda hesaba katılmamaktadır. Formül 1.4'de anahtar-tabanlı eşleştirme algoritmasının maliyet fonksiyonu verilmiştir.

$$T = \alpha_1 + \alpha_3 + (k^3/4)L\tau + n_{cp\_match} \quad (1.4)$$

4.5 Bölge-Tabanlı Eşleştirme Algoritması

Bölge tabanlı eşleştirme algoritmasında CS kullanıldığı durum yine Bölüm 4.2'deki gibi herkesin-herkese yayın durumu oluşacaktır. Maliyet fonksiyonu benzer şekilde Formül 1.5'de verildiği gibi bulunmaktadır. Bu durumda farklı olarak anahtar üzerinde algoritmanın koşma durumu yerine algoritma PE'lerde koşmaktadır. Anahtarlama gecikmesi sadece 1 Lτ'dir ve maliyet fonksiyonuna eklenmiştir.

$$T = \alpha_1 + \alpha_2 + \alpha_3 + L\tau(k^3/4+1) \quad (1.5)$$

Sadece eşleştirme algoritması için düğümlerde harcanan α_2 süresi ile IS'de harcanan süreyi karşılaştırmak mümkündür. ARM A9 kullandığımızda testlerde, 64B mesaj uzunluğu ve 16 düğümün eşleştirme işlemi yaklaşık 14μs'de hesaplanmıştır. IS kullanıldığında ise anahtarda eşleştirme algoritması için harcanan süre n_{cp_match} olduğu için toplamda 0.94μs harcanmaktadır. Fakat düğüm sayısı 2000'e çıkacak olursa IS'de harcanan süre yine 0.94μs iken CS'de 806μs'ye harcanmaktadır. Bu sonuçlara göre hesaplama gecikmesi IS'de, anahtar-tabanlı GALT algoritması ile herkese yayın-tabanlı algoritmaya göre 470 kat ve şişman ağaç-tabanlı algoritmaya göre 229 kat iyileştirilmiştir. Eşleştirme algoritmasında ise anahtar-tabanlı yaklaşım ile elde edilen iyileşme 857 kat olarak bulunmuştur.

$\alpha_1 + \alpha_3$ değeri paketin işlemcinin yardımıyla hafızadan okunup ağa gönderilmesi ve sonra tekrar paket ağdan alındığında hafızaya yazılması için geçen süre toplamıdır. [56] çalışmasında bu süre yaklaşık 7,4us alınmıştır. Maliyet fonksiyonlarının analizlerinde bu değer referans alınmıştır. Maliyet fonksiyonlarındaki τ değeri 1Gb Ethernet için 8ns alınmıştır. α_2 değeri Xilinx ZC702 geliştirme kartındaki ARM Cortex A9 CPU'da farklı düğüm sayıları ve mesaj boyutları için ölçülmüştür. A9 işlemci frekansı 667MHz ve SDRAM bant genişliği 4264MB/s'dir.

5. AKILLI ETHERNET ANAHTARI

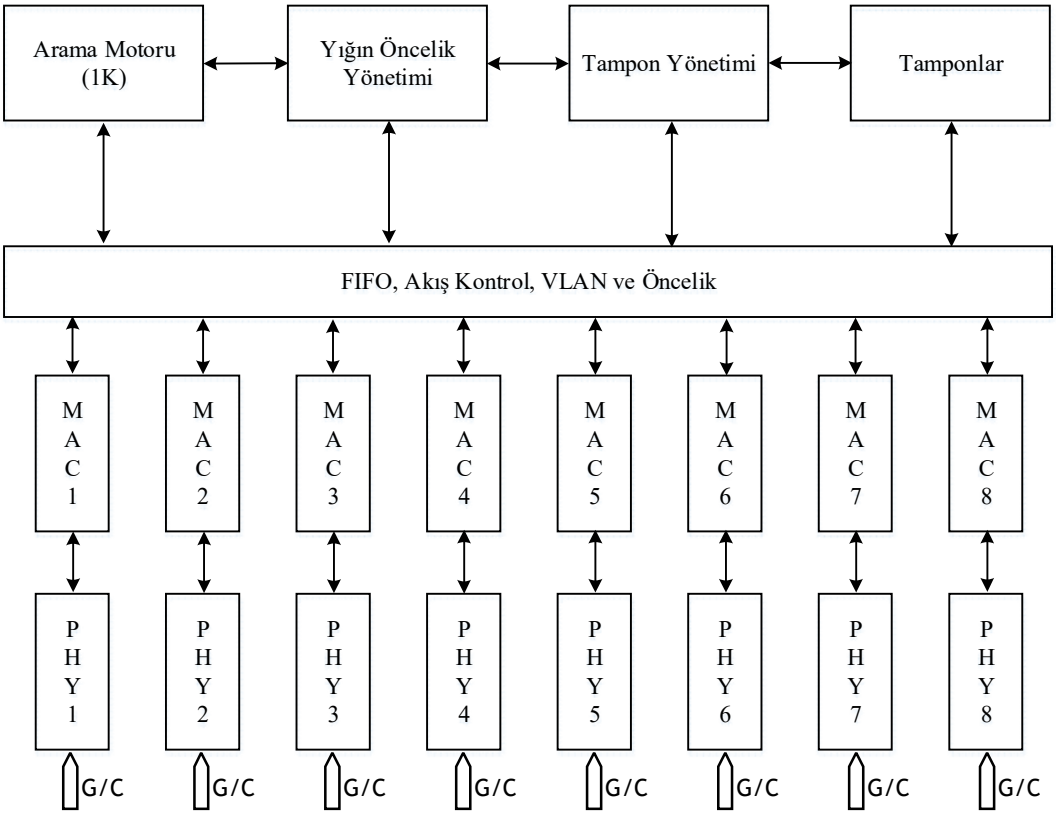
Bu bölümde bu tez kapsamında ortaya konulan akıllı Ethernet anahtarı (IS) tasarımı detaylı olarak anlatılmıştır. Tasarlanan Ethernet anahtarının CS'ye nasıl bir iyileştirme getirdiği gösterilmiştir. Ayrıca Ethernet anahtarını oluşturan temel alt birimler ve test etmekte kullanılan yardımcı birimler detaylandırılmıştır. Tasarlanan Ethernet anahtarının FPGA tasarımının blok şeması da yine bu bölümde verilmektedir. IS tasarımının çekirdeğinde ileriki kısımlarda detaylı anlatılan TM/DDM hızlandırıcı ünitesi (İng. TM/DDM Accelerator, TDA) yer almaktadır. Bu blok aslında bu tez çalışması kapsamında ortaya çıkan ağ anahtarının klasik ağ anahtarlarından temel farkıdır. TDA ile TM ve DDM servislerinin geçmişteki çalışmalara göre daha yüksek performans ile çalıştırılması ve ağda kalan geçiş mesajlarının ortadan kaldırılması hedeflenmiştir.

5.1 Ethernet Anahtarı Mimarisi

Ethernet anahtarları endüstride birçok alanda kullanılmaktadır. Örneğin OpenVPX standardında tasarlanan kart tipleri belirtilirken Ethernet anahtarları, 6 ana kart grubundan olan anahtar modülü kartlarında yer almaktadır [57]. Ethernet anahtarları OpenVPX standardında ağ topolojilerin de anlatılan merkezi anahtar mimarisinde kullanılmaktadır. Hatta değiştirilebilen cihazlarda (İng. Line Replaceable Unit, LRU) kullanılan Ethernet anahtarları cihaz içindeki farklı kartların Ethernet üzerinden haberleşmesini sağlar. Bu uygulamalarda yaygın olarak kullanılan Ethernet anahtar yongalarından bir tanesi Şekil 5.1'de mimarisi verilen KSZ8999'dur. KSZ8999'un temel özelikleri ise aşağıda verilmiştir [58]:

- 9 kapılı (8+1) 10/100Mbit bütünleşik anahtar, 8 adet fiziksel seviye gönderici-alıcı (İng. Physical Layer Transceiver, PHY) ve 1 adet ortam bağımsız arayüz (İng. Media Independent Interface, MII) veya seri ağ arayüzü (İng. Serial Network Interface, SNI).
- Paket tampon için 128KB SRAM.
- 2Gb SRAM bant genişliği.
- 1K MAC adresi için adres arama motoru (İng. Address look-up engine).
- Otomatik adres öğrenme (İng. Address learning), adres eskimesi (İng. Address aging), adres taşıma (İng. Address migration).
- 802.1p, öncelik (İng. Priority) ve kapı tabanlı öncelik desteği.

- Kapı tabanlı, sanal yerel ağ (İng. Virtual Local Area Network, VLAN) desteği.
- Donanım tabanlı tam/yarım çift yönlü, akış kontrol ve otomatik olarak anlaşma (İng. Auto-Negotiation, AN) desteği.
- Tam çift yönlü IEEE802.3x akış kontrolü desteği.
- Yarım çift yönlü geri baskılayan akış kontrolü desteği.
- Herkese-yayın fırtınası koruması (İng. Broadcast storm protection) desteği.



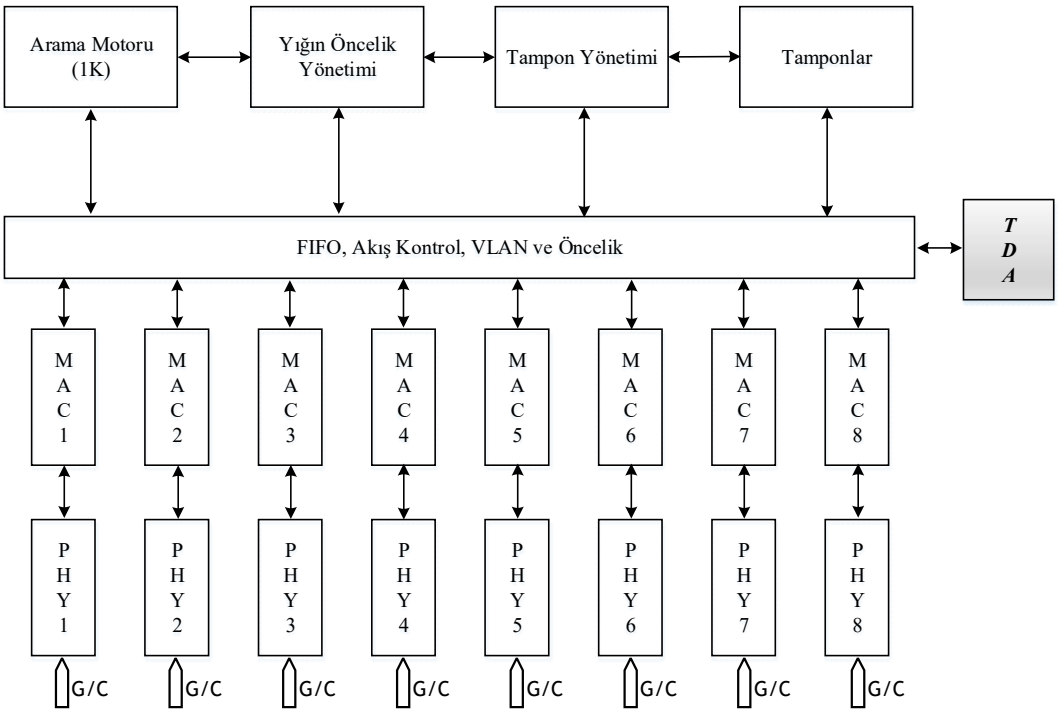
Şekil 5.1. KSZ8999 Temel Mimarisi [58].

Ethernet anahtarının 8 kapısı, doğrudan Ethernet haberleşme için kullanılabilir. 9. Kapı olan MII arayüzü ise harici Ethernet PHY'ye bağlanmak için veya PHY'siz iletişim için kullanılabilir. 128KB'lık SRAM'e 2Gb'de yazma okuma yapılabilmektedir. KSZ8999 kapılara ayrılan tampon boyutunu dinamik olarak değiştirebilmektedir. Yani trafiğin yoğunluğuna göre belirli bir kapıya ayırdığı tampon boyutunu artırıp-azaltabilmektedir. Dâhili adres arama motoru, dâhili adres arama tablosunda MAC adresleri ve MAC adres ile ilgili bilgileri saklamaktadır. 1K boyutlu içerik adreslenebilir hafızadan (İng. Content addressable memory) oluşmaktadır. Adres öğrenme sayesinde eğer gelen paketteki

kaynak adresi (İng. Source Address, SA) dâhili adres arama tablosunda yok ise eklenir. Ayrıca gelen pakette herhangi bir hata yok paket uzunluğu doğru ise dâhili adres arama motoru SA'yı, kapı numarası, zaman etiketi bilgileri ile birlikte adres arama tablosuna yazarlar. Tablo dolduğu zaman tablodaki son eleman silinip yerine yazılır. Dâhili adres arama motoru herhangi bir kapının adresinde değişiklik olup olmadığını sürekli olarak monitör eder. Eğer gelen pakette herhangi bir hata yok paket uzunluğu doğru, fakat gelen paketin SA'sı tabloda olmasına rağmen eşleştiği kapı bilgisi farklı ise adres taşıma gerçekleşir. Adres arama motoru SA'yı her gördüğünde ilgili kayıttaki zaman etiketini günceller. Zaman etiketi bilgisi adres eskime tayininde kullanılır. Adres arama motoru sürekli olarak bu zaman etiketlerini kontrol ederek adres eskime durumunu kontrol eder. Adres eskime süresi 300 saniyedir. Adres eskime durumunda ilgili kayıt tablodan silinir. Eğer hedef adres (İng. Destination Address, DA) arama tablosundaki adresler ile eşleşirse, KSZ8999 DA'yı paketin gideceği kapıyı bulmak için kullanır. Eğer DA arama tablosundaki adresler ile eşleşmez ise KSZ8999 gelen paketi, paketin geldiği kapı hariç tüm kapılara gönderir. Herkese-yayın durumunda benzer şekilde mesaj, paketin geldiği kapı hariç tüm kapılara gönderilir. Çok noktaya yayın (İng. Multicast) durumunda çoklu yayın grubundaki kapılardan mesajın geldiği kapı hariç hepsine mesaj paketi gönderilir. KSZ8999, paket çerçeve, boyut, çerçeve doğrulama kelimesi (FCS) hatası olan paketlere anahtarlama yapmaz. Ethernet anahtarlamayı sakla ve ilet (İng. Store & forward) şeklinde yapar. Her paket transferinden sonra bir önceki paketin transferi ile arada geçen zaman ölçülüp paketler arası boşluk (İng. Inter-frame gap) olarak saklanır. Yarım çift yönlü haberleşmede gönderme ve alma tek hat üzerinden gerçekleşir. Bu yüzden hatta çakışma olma riski vardır. Hatta çakışma olma durumunda KSZ8999, IEEE Std 802.3 ikili üstel geri çekilme algoritması (İng. Binary exponential back-off algorithm) uygulamaktadır. 16 çakışmadan sonra yonga yapılandırmasına göre paketler düşürülebilecektir. Ayrıca eğer gönderilen paket 512 bit zamanı boyunca çakışmaya maruz kalırsa paket düşürülür. KSZ8999, 64B'den büyük ve 1536B'den küçük paketleri kabul eder. Fakat dev çerçeve (İng. Jumbo frame) desteği yoktur. KSZ8999 paket gönderip ve alırken 802.3x akış kontrol çerçevelerini destekler. Çerçeve alma esnasında eğer KSZ8999 duraklatma kontrol çerçevesi alırsa, duraklatma kontrol çerçevesinde alınan sayaç değeri dolana kadar çerçeve göndermeyi durdurur. Eğer sayaç değeri dolmadan başka bir duraklatma kontrol çerçevesi gelirse sayaç son gelen çerçevedeki değer ile yüklenir. Akış kontrol esnasında KSZ8999'dan sadece akış kontrol paketleri gönderilir. KSZ8999 veri aldığı

kapının hafıza kaynakları tükendiğinde akış kontrolü başlatır. KSZ8999 IEEE standart 802.3x'de belirtilen maksimum bekletme süresi içeren akış kontrol çerçevesini (XOFF) gönderir. Eğer kapıya ait hafıza kaynakları tekrar uygun olursa KSZ8999 akış kontrolünü durdurmak için sıfır bekleme zamanı içeren akış kontrol çerçevesini (XON) gönderir. Ayrıca KSZ8999'un akış kontrol mekanizmasının birçok defa açılıp kapanmasını engellemek için histerezis özelliği sağlanmaktadır. Yarım çift yönlü haberleşmede ise akış kontrol çerçevelerine ek olarak KSZ8999 ağ üzerindeki diğer istasyonların veri gönderimini erteletmek için bir başlama eki (İng. Preamble) göndererek geri baskılama ile akış kontrolünü sağlar. KSZ8999 anahtarın çok fazla herkese-yayın paketi almasını engelleyecek herkese-yayın fırtınası koruması metodu mevcuttur. Herkese yayında gelen yayın yayını gönderen kapı hariç tüm kapılara gönderileceği için gerek bant genişliği gerekse hafıza olarak çok fazla kaynak tüketecektir. KSZ8999, yapılandırma hafızasına yazılan periyod sınırından (İng. Threshold) daha sık herkese-yayın paketi gelmesi durumunda gelen herkese-yayın paketlerini almayacaktır. KSZ8999 gelen paketlerin önceliğini algılayabilir. Kapı tabanlı öncelikte belirli bir giriş kapısına öncelik verilebilir. Bu durumda hedef yığınında bu kapı öncelikli olur. Buna örnek olarak internet telefon protokolü verilebilir. Böylelikle internet telefon protokolü trafiği tüm ağda önceliğe sahip olabilir. 802.1p metodunda ise kapıdan karışık veri (yüksek ve düşük öncelikli) gelirse kullanılır. Paketin geldiği kapıda gelen paketteki öncelik etiketi incelenerek paketin önceliğinin yüksek veya düşük olduğuna karar verilir.

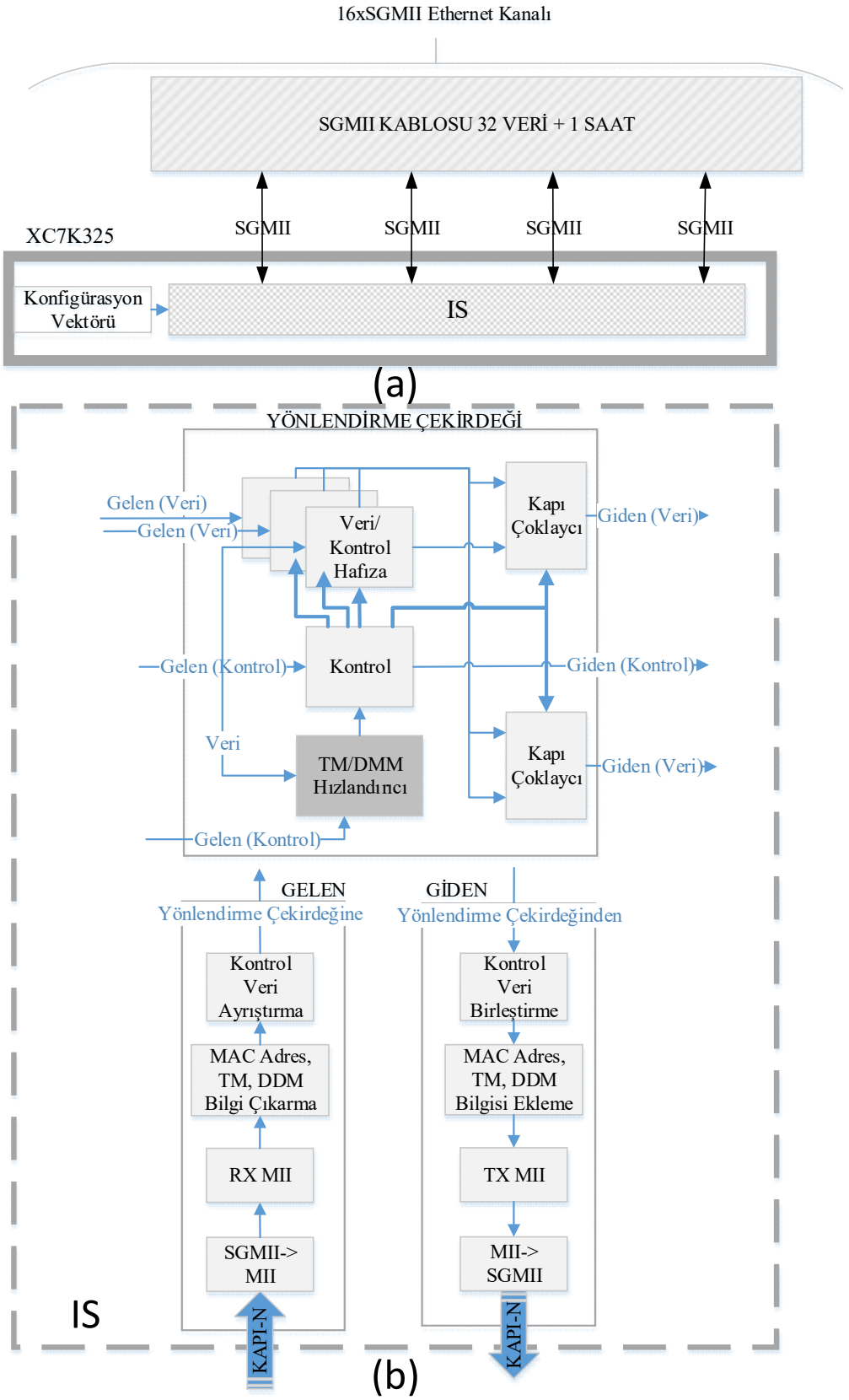
KSZ8999 sanal ağları da desteklemektedir. VLAN maskeleme saklayıcıları açılarak aktifleştirilir. VLAN ayarları giriş kapısından yapılır ve giriş kapısının anahtar üzerinde hangi çıkışları görebileceği belirlenir. VLAN'a katılan kapı, VLAN maskeleme saklayıcısına VLAN'da bulunan diğer kapıları girer. Böylelikle VLAN katılan kapı seçilen kapıları görebilecektir. Çok noktaya ve herkese-yayın çerçeveleri de VLAN ayarlarına uyarlar. Herhangi bir VLAN'daki kapıya birçok noktaya veya herkese-yayın paketi gelirse sadece VLAN'da kayıtlı kapılar bu mesajları alabilirler. Otomatik olarak anlaşma sayesinde hat üzerindeki çiftler en iyi ortak ayarları seçebilir ve birbirlerine yeteneklerini yayınlayabilirler. Otomatik olarak anlaşma yapılabilmesi için hat üzerindeki çiftlerin otomatik anlaşmayı desteklemesi gerekir. Aksi takdirde KSZ8999 otomatik olarak anlaşmayı doğrudan geçer [58]. Şekil 5.2'de bu tez kapsamında KSZ8999 Ethernet anahtarı gibi bir CS'ye eklenen donanım TDA olarak gösterilmiştir.



Şekil 5.2. Ethernet Ağ Anahtarı Mimarisinde Yapılan Ekleme.

Burada hedeflenen CS mimarisine GALT ve eşleştirme algoritmaları için bir donanımsal hızlandırıcı eklenmesidir. Aslında bu hızlandırmayı içerisinde işlemci bulunan VSC744x-02 [59] serisi modern Ethernet anahtarları ile gerçekleştirmekte mümkündür. VSC744x-02 serisinde 500MHz MIPS24KEc uyumlu mikroişlemci bulunmaktadır. Bu mikroişlemci yüksek bant genişlikli Ethernet DMA ve 1GB DDR3 arayüzünü destekleyecek DDR3 kontrolcü bulundurmaktadır. Fakat bu işlemleri mikroişlemci de yapmak küçük ağlar için avantajlı olsa da ağ büyüdükçe mikroişlemcinin işlemleri seri olarak gerçekleştirmesinden kaynaklı gecikme nedeni ile mikroişlemci kullanarak yapmak dezavantajlı olacaktır. Örneğin; 125MHz'de 1 Byte olarak gelen Gb Ethernet çerçevelerinden bu işlemci ile sınırlı sayıdaki kapı için işlenebilecektir. Yani kapı sayısı artacak olursa bu işlemci performans olarak önerilen mimarinin gerisine düşecektir.

Bu tez çalışmasında ortaya konulan IS tasarımı Şekil 5.3'de gösterilmiştir. Şekil 5.3.a'da IS'in KC705 deneme kartı (İng. Evaluation Board, EVB) üzerinde uygulaması, Şekil 5.3.b'de ise IS blok diyagramı verilmiştir. Burada IS olarak gösterilen, CS içine TDA bloğunun eklendiği Ethernet anahtarı tasarımıdır.



Şekil 5.3. IS Mimarisi ve KC705 Deneme Kartında Kullanımı.

Şekil 5.3'de verilen IS mimarisini Şekil 5.1'de verilen Ethernet anahtarı mimarisi ile karşılaştıracak olursak:

- Örnek Ethernet anahtarı mimarileri otomatik adres öğrenme, adres eskimesi, adres taşımayı desteklerken IS'de bu özelikler bulunmamaktadır. IS'de MAC adresleri her kapı için donanıma açılış esnasında yüklenir. IS'nin dinamik ortamlarda test edilmesi gerekmediği için bahsedilen özellikler uygulanmamıştır.
- Örnek Ethernet anahtarı mimarileri trafik yoğunluğuna göre tampon boyutunu dinamik olarak değiştirebilmektedir. Akış kontrol kullanıldığı durumda tampon belirli bir seviye doluluktayken karşı tarafa paket gönderme duraklatılabileceği için bu özelik çok kritik bir özelik değildir. Bu yüzden IS'de uygulanmamıştır.
- Örnek Ethernet anahtarı mimarilerinin desteklediği sanal ağ özeliği IS'de uygulanmamıştır. Çünkü IS'de test edilen algoritmalarda tüm kanallar tek bir ağda kullanılmaktadır. IS tasarımında kullanılan MAC ise VLAN özelliğini desteklemektedir.
- Örnek Ethernet anahtarı mimarilerinin desteklediği öncelik tabanlı paket özelliği IS'de uygulanmamıştır. Çünkü IS'de alınan paketlerin öncelikleri eşittir. IS tasarımında kullanılan MAC ise bu özelliği desteklemektedir.
- IS'de yarım çift yönlü haberleşme desteklenmemektedir. Testlerde yarım çift yönlü haberleşme olmadığı için hatta çakışma olmayacaktır. Bu yüzden örnek Ethernet anahtarı mimarilerin de kullanılan çakışma algoritmalarının kullanılmasına gerek yoktur. IS tasarımında kullanılan MAC ise yarım çift yönlü haberleşmeyi desteklemektedir. Fakat SGMII PHY IP'si (İng. Intellectual property) yarım çift yönlü haberleşmeyi desteklemediği için yarım çift yönlü haberleşme kullanımı bu mimaride mümkün değildir.
- Örnek Ethernet anahtarı mimarileri paketler arası zaman ölçebilmeyi desteklemektedir. IS'de ise paketler arası zaman ölçebilme özelliğini desteklememektedir. Çünkü adres eskime özelliği kullanılmadığı için bu şekilde bir zaman ölçümüne IS'de ihtiyaç yoktur.
- Gelişmiş örnek Ethernet anahtarında (VSC744x-02 serisi gibi) virüsten koruma için paket filtreleme gibi işlemler dâhili işlemciyi kullanarak yapılabilmektedir. Tasarlanan IS'de böyle bir ihtiyaç olmadığı için bir işlemci tasarıma eklenmemiştir.

- Gelişmiş örnek Ethernet anahtarı yayın/abone sistemlerinde kullanmak üzere internet grup yönetim protokolünü (İng. Internet Group Management Protocol, IGMP) destekler. Örneğin VSC744x-02 serisi ağ anahtarları IGMPv3'ü destekler. IGMPv3 sayesinde Ethernet anahtarı üzerinde çoklu yayın grupları oluşturulabilmektedir. Yayın yapan düğümler abone isteklerine göre çoklu yayın grubu oluşturabilir. TDA sayesinde TM ve DDM servisleri için IGMP ile çoklu yayın grubu oluşturmaya ihtiyaç kalmayacağı için bu özeliğin IS'ye eklenmesine ihtiyaç yoktur.
- IS'de, örnek Ethernet anahtarı mimarisinde bulunan kuyruklama (İng. Queueing) işlemleri yapılmamıştır. Kuyruklama işlemleri paralel ayrık olay benzetici (İng. Parallel Discrete Event Simulator, PDES) program üzerinde yapılmaktadır.
- IS içerisinde bulunan TDA bloğu, GALT ve eşleştirme algoritmalarını desteklemektedir. Örnek Ethernet anahtar mimarilerinde bulunmayan bu özelik sayesinde IS kullanarak kurulan paralel ayrık olay benzetimlerinde TM ve DDM servislerinin performansı artırılabilmektedir.

Şekil 5.3.b'de verilen IS blok tasarımında gösterilen "SGMII->MII" ve "MII->SGMII" blokları için Xilinx'in "1G/2.5G Ethernet PCS/PMA or SGMII 16.0" IP'si kullanılmıştır. "RX MII" ve "TX MII" blokları için ise yine Xilinx'in "Üç Modlu Ethernet MAC 9.0 (İng. Tri Mode Ethernet MAC, TEMAC)" IP'si kullanılmıştır. "MAC Adres, TM, DDM Bilgi Çıkarma/Ekleme" blokları ise MAC adres, TM ve DDM servilerinin çalışması için gerekli olan mesaj başlık ve verilerinin ayıklanması ve oluşturulması işlemlerini gerçekleştirmektedirler. Benzer şekilde bu bilgiler kullanılarak yönlendirme çekirdeğindeki kontrol ve TDA blokları için ilgili kontrol kelimeleri ile verilerinin oluşturulup okunması da "Kontrol Veri Ayrıştırma/Birleştirme" blokları ile sağlanmaktadır. Yönlendirme çekirdeği anahtarlama işlemlerinin yapıldığı ana bloktur.

5.1.1 Xilinx IP 1G/2.5G Ethernet PCS/PMA or SGMII 16.0

"1G/2.5G Ethernet PCS/PMA or SGMII 16.0" kısaca SGMII IP'si Xilinx tarafından geliştirilmiştir [60]. GMII formatında alınan 125MHz, 8-bit verinin, SGMII formatına çevrilmesini sağlar. Bir çift LVDS diferansiyel kapı ile (RX/TX), 1Gbit/s hızında Ethernet verisinin alınıp/gönderilmesini sağlar. Fiziksel kodlama alt katmanı (İng. Physical Coding Sublayer, PCS) kısmı GMII'dan gelen verinin 8B/10B kod

açımı/kodlama işlemini yapar. SGMII için link eşleri ile veri değişimini gerçekleştirip otomatik olarak anlaşma yapılmasını sağlar. Fiziksel ortam bağlantısı (İng. Physical Medium Attachment, PMA) kısmında veri gönderme ve alma işlemleri için serileştirme/seri veri çözme yapılır. 8B/10B kodlanmış veri için saat geri kazanımı yapılır. Yapılandırma ve gelişmiş yapılandırma vektör girişlerinden bu IP'ye ait ayarlar yapılır. Aşağıda giriş kapılarından verilen bu IP'ye ait yapılandırma ve gelişmiş yapılandırma vektörü ayarları gösterilmiş ve açıklanmıştır. Yapılandırma vektör ayarı yanındaki ikili değer (İng. Binary) IS tasarımda ayarlanan değerdir.

- Unidirectional Enable[0]=0: '1' olduğunda hat kurulma durumundan bağımsız olarak veri gönderimi gerçekleşebilir. AN olmadığında kullanılır.
- Loopback Control[1]=0: '1' olduğunda IP geri döngü moda alır.
- Power Down[2]=0: '1' olduğunda 7 serisi FPGA'lerde donanıma özgü gönderici-alıcılar güç kapalı moda geçer. İlk duruma getirme ile temizlenir.
- Isolate[3]=0: '1' olduğunda GMII arayüzünden gelen paketlere cevap verilmemekte ve paket gönderimi (elektriksel bağlantıda yüksek empedans) durdurulmaktadır. Böylece ortama erişim operasyonları durdurulmaktadır.
- Auto-Negotiation Enable[4]=1: '1' olduğunda IP'de AN arayüzü aktif ise AN özelliği açılır.

Gelişmiş yapılandırma vektörü ayarları ise aşağıdaki gibidir:

- Bit[0]=1: SGMII için '1' seçilmeli.
- Bit[5]=X: SGMII: Saklı değer.
- Bit[8:7]=X: SGMII: Saklı değer.
- Bit[11:10]=[10]: Hız 1000Mb/s.
- Bit[12]=1: Tam çift yönlü haberleşme aktif.
- Bit[15]=1: Hat aktif.

5.1.2 Xilinx IP Tri Mode Ethernet MAC 9.0

TEMAC IP'si Xilinx tarafından geliştirilmiş bir IP'dir [61]. AXIS arayüzünden alınan 125MHz, 8-bit verinin GMII formatına çevirerek, SGMII formatına çevrilmek üzere SGMII IP'sine aktarılmasını sağlar. 10/100/1000Mb Ethernet haberleşme hızlarını

destekler. Ethernet paketinde bulunan başlama eki, çerçeve başlangıç belirteci, doldurulmuş bölüm ve çerçeve doğrulama kelimesi alanları yine bu IP tarafından oluşturulur. Yarım çift yönlü haberleşmeyi ve öncelik tabanlı akış kontrolünü destekler. IP, yapılandırma girişinden saklayıcılarına yüklenecek yapılandırma değerini okur. Aşağıda bu IP'ye ait yapılandırma ayarları verilmiştir. Yapılandırma ayarının IP'ye verilen ikili sistemdeki değeri yanındaki eşittir operatörü ile gösterilmiştir.

- Transmitter/Receiver Pause Frame Source Address[47:0] = X"0605040302DA": Akış kontrolü için kullanılır.
- Transmitter/Receiver Max Frame Size[15:0]=0: Maksimum çerçeve uzunluğu standartta izin verilenin üzerinde ise girilir. 1518 ve üzerinde değer girilmelidir.
- Transmitter/Receiver Max Frame Enable[14]=0: Bu değer '1' ve Jumbo çerçeve '0' olursa standartta izin verilen çerçeve boyutunun üzerine çıkılabilir.
- Transmitter/Receiver Speed Configuration[13:12]=10: Bu bitler hız ayarlamak için kullanılır. Aktif olması için bir ilk duruma getirme sinyali verilmelidir. 1Gbit için '10' girilir.
- Transmitter Interframe Gap Adjust Enable[8]=0: Bu bit '1' olursa tx_ifg_delay_port kapısında verilen kadar gecikmeye izin verir. 0 verilirse legal minimum kadar iç çerçeve arası boşluğu yaratılır.
- Transmitter/Receiver Half-Duplex[6]=0: Bu bit '1' olursa yarım çift yönlü, '0' olursa tam çift yönlü çalışılır.
- Transmitter/Receiver Flow Control Enable[5]=1: Bu bit '1' olduğunda duraklatma isteği geldiğinde göndermeyi duraklatma kontrol çerçevesi (İng. Transmitter pause control frame) gönderilir. '0' ayarlanırsa duraklatma isteği sinyali dikkate alınmaz.
- Transmitter Jumbo Frame Enable[4]=1: Bu bit '1' set edilirse Jumbo çerçeve yani standartta verilen çerçeve uzunluğunun üzerine çıkılabilir. Fakat '0' set edilirse maksimum çerçeve uzunluğu ile izin verilen çerçeve veya maksimum çerçeve aktif değil ise standart tarafından izin verilen en büyük çerçeveye izin verilir.
- Transmitter/Receiver In-Band FCS Enable[3]=0: Bu bir '1' set edilirse kullanıcı gönderici çerçeve kontrol kelimesi (İng. Frame Check Sum, FCS) değerini hesaplayıp girmeli ayrıca veri 64-Byte altında ise veriyi 64-Byte tamamlamalıdır.

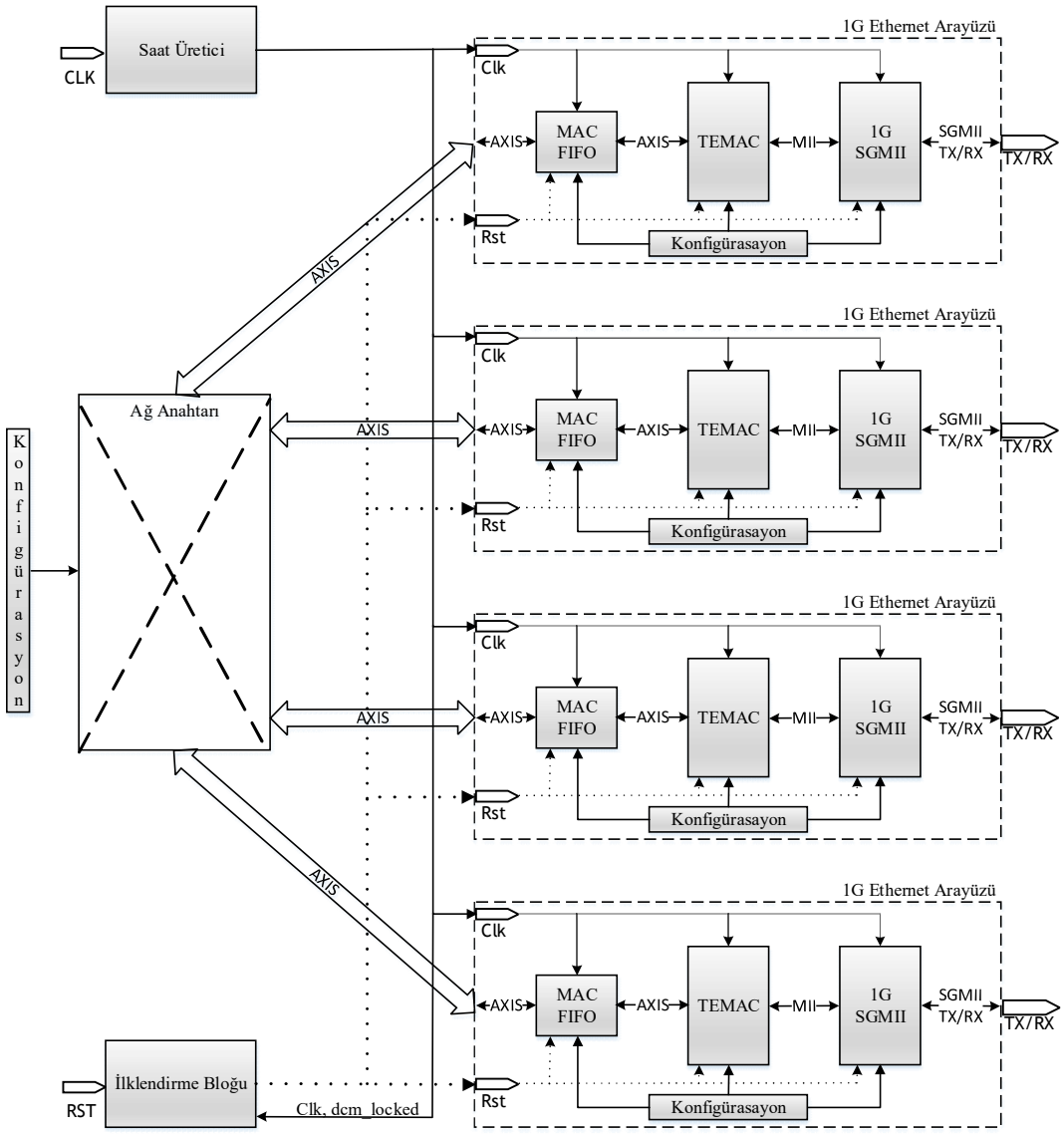
‘0’ set edilirse doldurma (İng. Paddding) ve FCS hesabını TEMAC yapıp verinin sonuna ekler.

- Transmitter/Receiver VLAN Enable[2]=0: Bu bit ‘1’ set edilirse gönderici VLAN etiketli çerçeveleri 1522-Byte'a kadar kabul eder.
- Transmitter/Receiver Enable[1]=1: Bu bit göndericiyi açıp kapamak için kullanılır. ‘1’ olduğunda açılır.
- Transmitter/Receiver Reset[0]=1->0: Bu sinyal göndericiyi ilk duruma getirir. Aktif durumda lojik ‘1’ olur. Çalışma esnasında önce lojik ‘1’ verilip sonra lojik ‘0’ verilmektedir.
- Receiver Promiscuous Mode[11]=0: Bu moda ‘1’ ayarlanırsa çerçeve filtre yapılmadan tüm veriler alınır.
- Receiver Control Frame Length Check Disable[9]=0: Bu bit ‘1’ set edilirse gelen çerçeveler minimum çerçeve uzunluğundan kısa mı diye kontrol edilmez. ‘0’ ise minimum çerçeve uzunluğundan kısa ise hatalı çerçeve diye işaretlenir.
- Receiver Length/Type Error Check Disable[8]=0: Bu bit ‘1’ olursa uzunluk/tip alanları kontrol yapılmaz. ‘0’ olursa kontrolü yapılır. Normalde akışta kontrol edilmelidir.

5.1.3 IS Tasarımı

TEMAC ve SGMII IP’leri ihtiyaç duyulan kapı sayısı kadar kullanılarak ağ anahtarı IP’sine bağlanır. Böylece IS oluşturulur. IS çekirdeğinde bulunan ağ anahtarı, tasarım modülerliği göz önünde bulunarak ayrı bir IP olarak geliştirilmiştir. Gerçekleştirilen IS tasarımının temel özelikleri aşağıda verilmiştir.

- IS çekirdeğindeki ağ anahtarı IP’si AXIS arayüzünde her bir kapıdan gelen 125MHzx8bit verinin ilgili kapıya AXIS arayüzünde gönderilmesini sağlar. Yani SGMII ve TEMAC IP’lerinden gelen Ethernet çerçevesi bu IP içinde işlenir.
- 4, 8, 16 kapı için GALT algoritması, tasarlanan IS ile KC705 deneme kartı üzerinde test edilmiştir. Eşleştirme algoritmasının 4 kapı için benzetimi yapılmıştır.
- Test sisteminde ZC702 kartından, Şekil 5.10’de belirtilen paket yapısına uygun gelen Ethernet çerçeveleri ile seçilen algoritmalar çalıştırılabilmektedir.

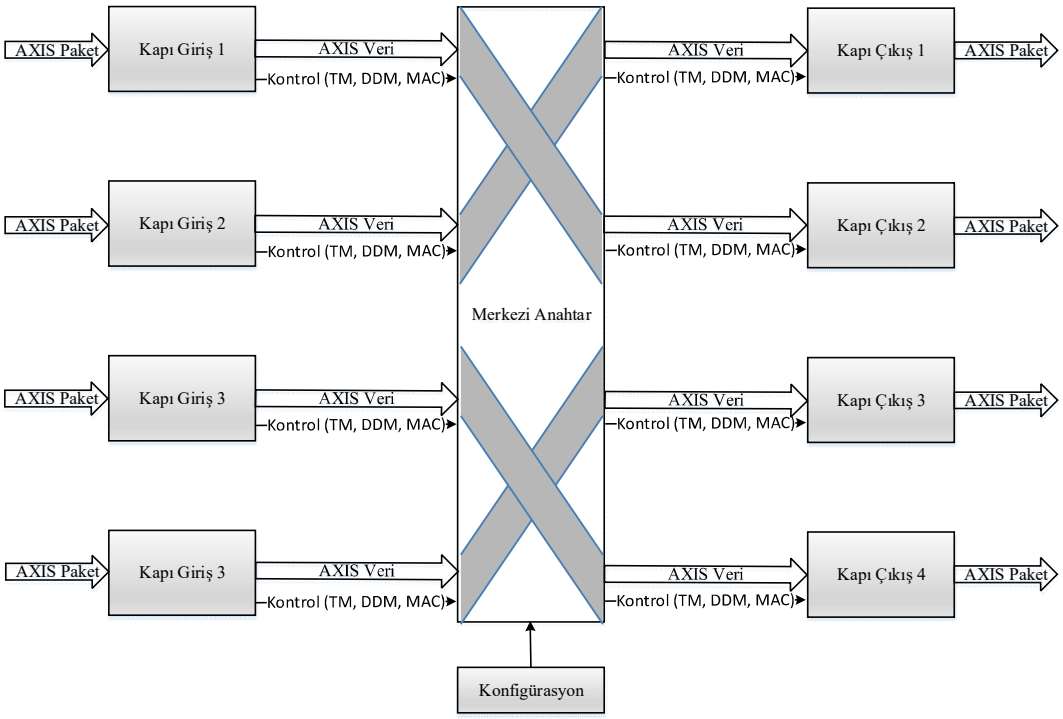


Şekil 5.4. IS'nin FPGA Tasarımı Blok Şeması.

Şekil 5.4'de IS'nin FPGA tasarımı blok şeması gösterilmiştir. Burada ağ anahtarı IP'sinin bağlandığı hiyerarşik 1G Ethernet arayüzü blokları içerisinde TEMAC, SGMII, MAC FIFO ve yapılandırma ayarlarının yüklendiği IP'ler bulunmaktadır. Her bir kapı için bu yapılardan bulunması gerekmektedir. Bu projede ayrıca FPGA üzerindeki saat girişlerinden alınan saat sinyalleri ile ilgili saatlerin oluşturulduğu faz kitlenmeli döngü (İng. Phase Locked Loop, PLL) IP'si bulunmaktadır. Bu IP tek saat girişinden alınan saat ile ihtiyaç duyulan tüm saatlerin (125MHz diferansiyel, 625MHz, 104MHz, 208MHz… gibi) üretilmesini sağlar. Ayrıca IP'lerin ilk duruma getirilmesi içinde ayrı bir IP bulunmaktadır. Bu IP'ler ile birlikte IS tasarımı tamamlanmaktadır.

5.2 Ağ Anahtarının IP Tasarımı

Bu bölümde tasarlanan ağ anahtarı IP'sinin detayları verilmiştir. Şekil 5.5'de tasarlanan ağ anahtarı IP'sinin 4 kapı için blok şeması gösterilmiştir. Tasarlanan 4 kapılı ağ anahtarı her kapı için MAC adres bilgisini konfigürasyon vektörü ile almaktadır. Ayrıca bir kontrol saklayıcısından anahtar kontrol değerlerini almaktadır. Kontrol saklayıcısından gelen değer seçili kapıyı kapatmak ve alt-üst yayın grubu oluşturmak için kullanılır. Alt-üst yayın grupları sayesinde alt bağlantılar (İng. Downlink) ve üst bağlantıların (İng. Uplink) ayrılması sağlanır.



Şekil 5.5. 4 Kapılı Ağ Anahtarları Blok Tasarımı.

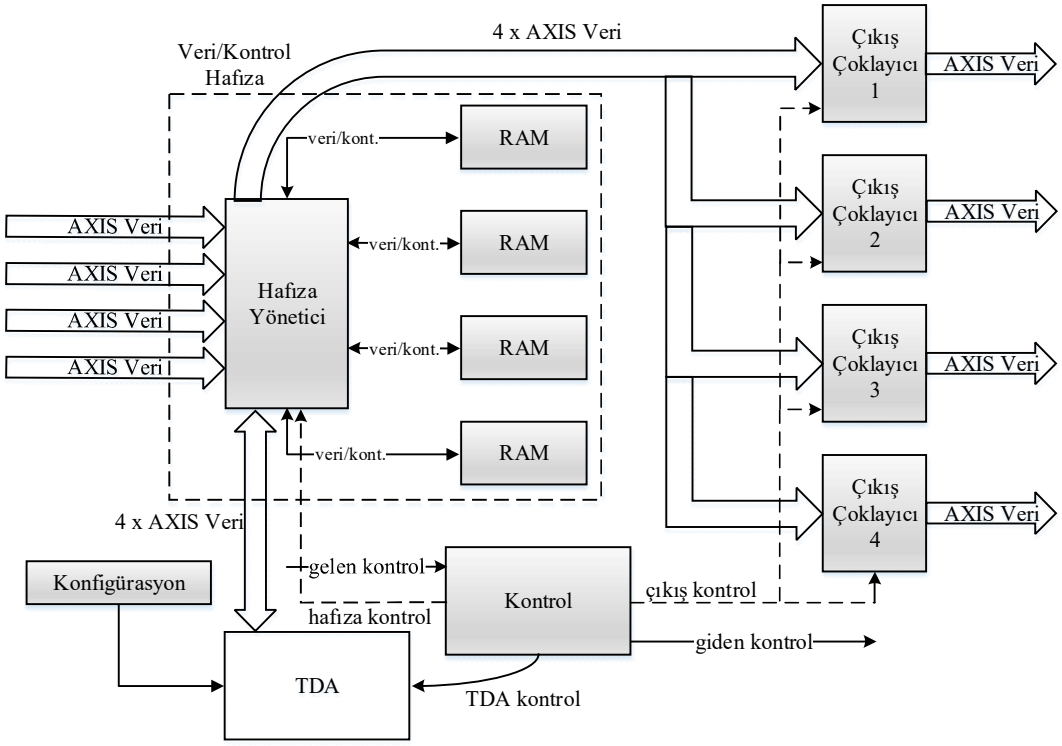
İlerleyen bölümlerde tasarlanan 4 kapılı ağ anahtarı IP'sinin alt blokları ve işlevleri anlatılmıştır. İlk olarak kapı giriş/çıkış bloğu anlatılmıştır. Kapı giriş bloğu Şekil 5.3'de verilen IS blok tasarımında MAC adres, TM, DDM bilgilerinin çıkarıldığı ve kontrol ile AXIS verilerinin oluşturulduğu bloktur. Kapı çıkış bloğu da çıkış tarafında tam tersi işlemleri yapmaktadır. Merkezi anahtar bloğu ise paket anahtarlama işleminin yapıldığı bloktur. Kapı giriş/çıkıştan gelen AXIS verilerin tamponlanması ve yine kapı giriş/çıkıştan gelen kontrol bilgilerine göre paketlerin anahtarlaması yapılır. İçinde bulunan TDA bloğu ise bu tez çalışması kapsamında geliştirilen bloktur.

5.2.1 Kapı Giriş/Çıkış Bloğu

Kapı giriş bloğu bir adet ilk giren ilk çıkar ve mesaj ayrıştırıcı ve birleştirici lojiğinden oluşmaktadır. Kapı giriş bloğunda gelen mesaj AXIS arayüzünden alınmaktadır. İçeride bir mesaj sayıcı ile mesaj baytları sayılmaktadır. Bu mesaj sayıcı yardımıyla gelen mesaj içerisindeki hedef/kaynak IP adresleri, mesaj boyutu, haberleşme (operasyon) tipi, özel operasyon kodu ve operasyon kodu kontrol verileri ayrıştırılır. Özel operasyon kodu minimum, maksimum ve lojik VE işlemleri için kullanılır. Operasyon kodu kontrol veri tipini kayan noktalı, işaretli ve işaretsiz tam sayı olarak seçmek için kullanılır. Ayrıca haberleşme tipi tek yöne yayın, herkese yayın, asenkron hepsini azaltma ve yayın/abone mesajları olarak ayrılır. Mesaj ayrıştırıcı ayrıca yayın/abone sisteminde ilgili mesaja ait koordinat bilgilerini de ayırmaktadır. FIFO'nun nerdeyse dolu sinyali yardımıyla FIFO belirlenen mesaj seviyesinde veri alınca giriş veri kanalı durdurulabilmektedir. Bu yapıya geriye doğru baskılama (İng. Backpressure) yapısı denilmektedir. AXIS arayüzünün bir özelliğidir. Bu blok ile bu bloğun dışındaki blokların gecikmesi nedeni ile oluşabilecek veri kayıpları engellenebilmektedir. Normal şartlar altında FIFO'ya yazılan geçerli veriler, verinin gönderileceği hafızanın uygun olması durumunda bu bloktan dışarıya gönderilirler. Ağ anahtarı örneği 4 kapılı olduğu kapı giriş bloğundan 4 adet bulunmaktadır. Ağ anahtarı girişinde AXIS arayüzünden veri alan kapı giriş bloğu gibi, çıkışında da aynı işlemlerin tersten yapıldığı, AXIS arayüzünden veri gönderilen kapı çıkış bloğu bulunmaktadır.

5.2.2 Merkezi Anahtar Bloğu

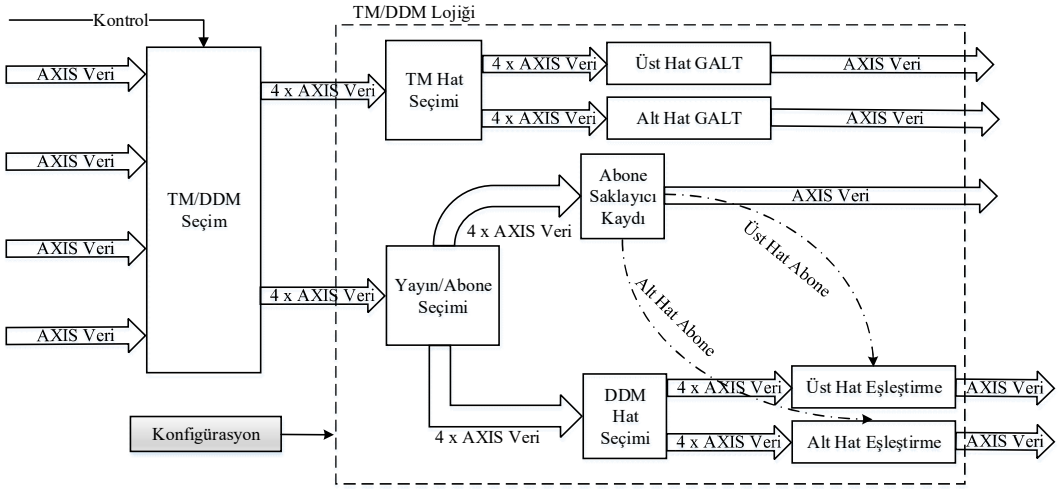
Şekil 5.6'de merkezi anahtar bloğu gösterilmiştir. Merkezi anahtar bloğunda 1 adet veri/kontrol hafıza, 4 adet çıkış çoklayıcı (İng. Multiplexer), 1 adet kontrol lojiği ve TDA bloğu bulunmaktadır. Bu bahsedilen lojik devreler TDA hariç standart bir ağ anahtarında bulunanlardır. Fakat bu tez kapsamında tasarlanan ağ anahtarında fazladan bir de bu tez çalışmasında geliştirilen TDA bloğu bulunmaktadır. Giriş kapıları ile çıkış kapıları arasındaki ilişki her kapı için gelen kontrol bilgileri kullanılarak merkezi anahtar lojiğinin kontrol bloğu üzerinde kurulur. Kontrol bloğu ayrıca gönderilen paketin gittiği kapının kontrol bilgisini de oluşturup gönderir. TDA bloğunu da kontrol eder. Hafıza yönetici üzerinden çıkış kapılarına gönderilecek veriyi kontrol eder. Ayrıca verinin gideceği kapının çoklayıcısını da kontrol eder.



Şekil 5.6. Merkezi Anahtar Blok Tasarımı.

5.2.3 TDA Bloğu

TDA bloğu standart Ethernet anahtarına bu tez çalışması kapsamında eklenen bloktur. Bu blok içinde GALT ve eşleştirme algoritmaları çalıştırılmaktadır. GALT ve eşleştirme algoritması için Şekil 5.7'de tasarlanmış lojik devre kullanılmaktadır. TDA'da alt hatlardan gelen bağlantılar ve üst hatlardan gelen bağlantılar için ayrı işlem yapılmaktadır. Modüle GALT algoritması için üzerinde çalışacağı veri ile birlikte özel operasyon kodu gibi kontrol bilgiler gelir. Modülden ise GALT algoritmasının sonucu çıkar. Bu sonuç daha sonra kapı çıkış seçim lojiğinden dışarı aktarılmak üzere ilgili hafıza alanına yazılır. Bu yapıda GALT hesabına katılan kapıların verisi, kontrol bloğu tarafından gönderilen bilgilerinin hafıza yöneticiye ulaşması ile hafızadan çekilir. Hafızadan çekilen veriler kontrol bloğu tarafından gönderilen ayarlara uygun olarak işlenip bu lojikte tasarlanan veri işlem hattından yeni değer olarak çıkar. GALT hesabı için kayan noktalı sayı ile minimum işlemi ayar olarak seçilmektedir. Mesaj, giriş tamponundan çekildikten toplam 25 saat darbesi sonra GALT hesabının sonucu bu bloktan çıkar ve hafıza yöneticiye iletilir. Hafıza yönetici üzerinde gerekli tamponlamalar yapılır. Kontrol bloğu ilgili çıkış kapılarının uygun olması durumunda hafıza bloğunda bulunan yeni GALT değerini içeren paketi çıkış çoklayıcılarına gönderir.



Şekil 5.7. TDA Blok Tasarımı.

Eşleştirme algoritmasında her geçerli abone mesajı alındığında ilgili veriler saklayıcılara kayıt edilir ve ilgili çıkış kapılarına gönderilir. Yayın mesajı geldiğinde ise abone bilgeleri saklayıcılardan çekilip gelen paket üzerindeki eşleştirme algoritması koşar. Mesaj giriş tamponundan çekildikten toplam 118 saat darbesi sonra eşleştirme algoritmasının sonucu bu bloktan çıkar. Buradan elde edilen eşleştirme algoritması sonuçları kontrol ve hafıza bloğu aracılığı ile ilgili çıkış kapılarına gönderilir.

5.3 FPGA Kaynak Tüketimi

Bu çalışmada geliştirilen anahtar için Kintex 7 tabanlı XC7K325T FPGA üzerinde uygulama (İng. Implementation) yapılmıştır. Çizelge 5.1'de ilk satırda IS'nin, ikinci satırda CS'nin, üçüncü satırda ise GALT algoritmasının lojik devre kaynak tüketimleri ve toplam FPGA kaynağının kullanım oranları gösterilmiştir. Kullanılan FPGA'in 203800 LUT'u, 407600 saklayıcısı ve 445 adet blok RAM kaynağı vardır.

Çizelge 5.1. 4 Kapılı Ethernet Anahtarının IS, CS ve GALT Uygulamalarında XC7K325T FPGA için Kaynak Tüketimleri.

FPGA Blok İsmi	Kaynak Kullanımı			Kullanım Oranı(%)		
	LUT	Reg	RAM	LUT	Reg	RAM
IS	38962	26265	12	19,1	6,4	2,7
CS	35455	25299	12	17,4	6,2	2,7
GALT	3507	966	0	1,7	0,2	0

Bu sonuçlar IS ile CS'nin maliyet karşılaştırmalarında kullanılmıştır. CS'ye eklenen GALT algoritması bloğu Ethernet anahtarının kapı maliyetini yaklaşık %10 oranında artırmaktadır.

5.4 Test Sistemi

Bu bölümde ilk olarak test sisteminin temelinde bulunan deneme kartlarının seçimi anlatılmıştır. Sonrasında ise seçilen deneme kartları ile birlikte kurulan test sistemi detaylı olarak verilmiştir.

5.4.1 Deneme Kartı Seçimi

Test sisteminde birisi test örüntü üreteci diğeri ise IS'nin uygulandığı olmak üzere toplam 2 deneme kartı seçimi yapılmıştır. Kaynak tüketimi ve tasarım zorluğu açısından ilk olarak IS'nin uygulanacağı kart seçilmiştir. Deneme kartı seçiminde belirleyici gereksinim Ethernet kapı sayısı olmuştur. Düğüm sayısı 16, 54 ve 128 için testler yapılacağı için 128 Ethernet kapısının olduğu bir çözüme ihtiyaç duyulmuştur. Fakat üzerinde FPGA bulunan mevcut deneme kartları ile 128 Ethernet kapısı bulunan bir çözüme ulaşılamamıştır. Genelde FPGA deneme kartları ile çok düğümlü ağlar kurulmak istendiğinde farklı FPGA kartları birbirlerine 100G gibi yüksek bant genişlikli optik arayüzler ile bağlanmaktadır. Fakat bu çözümde araya giren optik bağlantı tasarlanan IS mimarisini çok karmaşıklaştıracağı için tercih edilmemiştir. Diğer bir çözüm ise OpenVPX gibi standart kartlar kullanmaktır. Fakat standart bir kart kullanıldığında o kartı taşıyacak bir sırtlık ve konektör bağlantılarının bulunacağı bir arayüz kartı tasarımı gerekmektedir. Bu yüzden OpenVPX hazır çözümleri tercih edilmemiştir. Tüm düğümler ile uygulamaya özel bir kart tasarlanmadan çalışmak mümkün olmadığı için donanım üzerinde tüm testlerin yapılacağı bir deneme kartı kullanmak yerine IS'nin anahtar mimarisinin doğrulanabileceği bir çözüme karar kılınmıştır. Ethernet anahtar mimarisini doğrulamak için 16 kapılı deneme kartları test çalışmaları için makul bulunmuştur. İkinci gereksinim ise FPGA kaynağıdır. Xilinx FPGA'ler saklayıcı olarak zengin oldukları için ve IS tasarımına kuyruklama gibi yoğun hafıza gereksinimi olan işlemler yapılmadığı için FPGA kaynaklarından sadece LUT miktarına bakılmıştır. FPGA kaynak tüketimi değerlendirmesinde tasarım ilk anda tamamlanmadığı için çeşitli firmaların Ethernet anahtarı kaynak tüketimlerine bakılmıştır. Üç farklı firmanın Ethernet anahtarları

incelenmiş ve 4 kapılı bir anahtarın ortalamada 40K LUT tükettiği gözlenmiştir. 16 kapılı anahtar için LUT ihtiyacının 160K'yı bulabileceği öngörülmüştür. GMII arayüzünden gelen verinin tamponlamaya uğramadan çalışabilmesi için incelenen tüm IP'lerin 125MHz'de seçili FPGA teknolojisinde uygulanabildiği ayrıca doğrulanmıştır. Test düzeneği basitliği değerlendirilirken ise IS için seçilen kartın test örüntü üretici kart ile bağlandıktan sonra ortaya çıkan düzenekte; kolay hata ayıklanması, test örüntülerinin kontrol edilmesi, sonuçların monitör edilmesi ve kablo karmaşası gibi alt metriklere bakılmıştır. Deneme kartlarının üzerine takılan hazır kartlar ile 18 kapıya kadar BASE-T Ethernet arayüzünde çıkılabildiği gözlenmiştir. Fakat BASE-T arayüzünde kablo karmaşası ortaya çıkacağı için bu yöntem tercih edilmemiş yerine SGMII üzerinden haberleştirilebilecek kartlar incelenmiştir. Sonuç olarak 32 kapıya kadar SGMII arayüzünden haberleşmeyi destekleyen ve 160K LUT gereksinimini karşılayan; Altera FPGA kullanılan "Arria 10 SoC" geliştirme kiti ve Xilinx FPGA kullanılan KC705 ile Inrevium TB-OK-D21 ürünleri incelenmiştir. Kintex 7 serisi, Arria 10 ve Ultrascale serilerine göre teknolojik olgunluğu daha fazla olduğu için bu çalışmada tercih edilmiştir. Ayrıca daha basit test kodlarının çalışacağı FPGA kapasitesi daha düşük ve SGMII üzerinden haberleşmeyi destekleyen ZC702 Kartı ise test örüntü üretici olarak seçilmiştir. ZC702 kartı ile test örüntüsü üretilmiş ve KC705 kartı ile de bu test örüntüleri alınıp anahtar-tabanlı GALT ve eşleştirme algoritmalarının uygulanması hedeflenmiştir. Aşağıda Şekil 5.8'de test sisteminin resmi gösterilmiştir. Burada soldaki kart Xilinx'in KC705 sağdaki kart ise ZC702 kartıdır. Aradaki özel kablo ile 16 diferansiyel hat, sinyal bütünlüğü bozulmadan SGMII arayüzünde taşınabilmektedir.

5.4.2 Test Sistemi Tasarımı

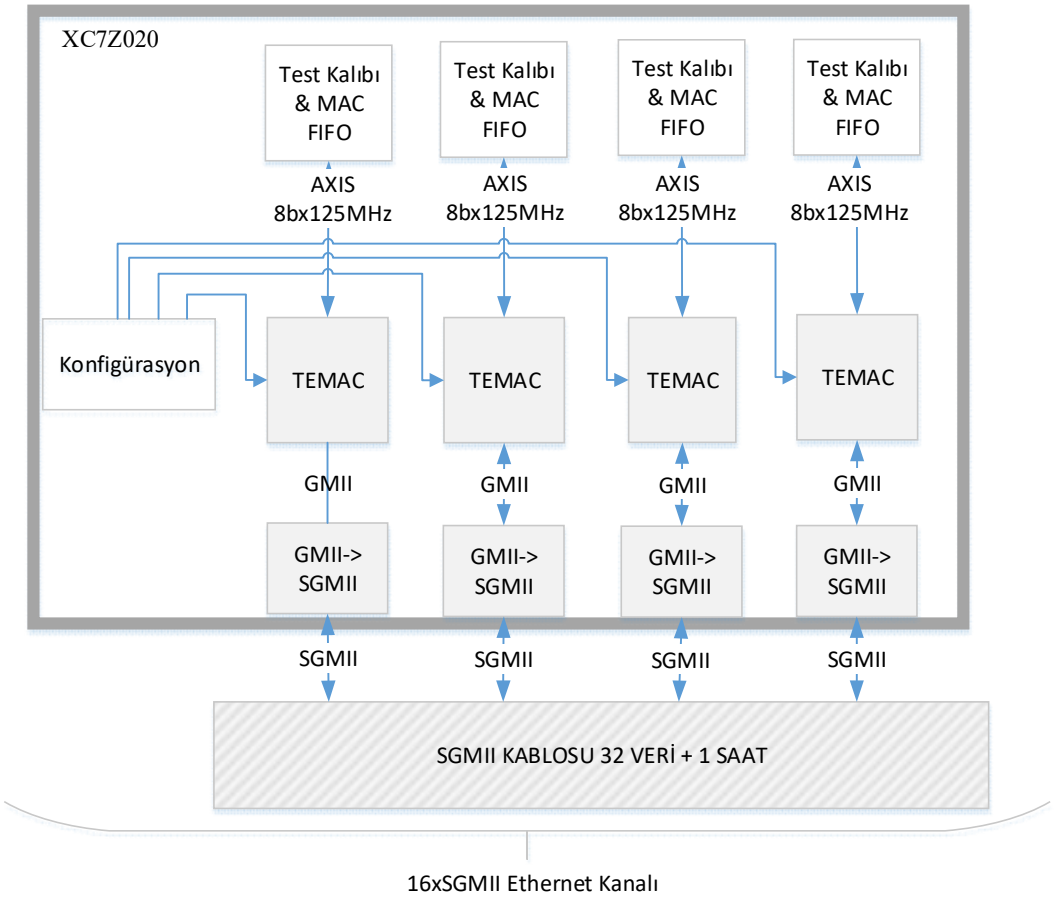
Testlerde ilk olarak fiziksel arayüzlerini test etmek için 4, 8 ve 16 kanaldan ZC702 kartında üretilen test örüntüleri KC705 kartına gönderilmiştir. KC705 kartında tasarlanan ağ anahtarından geçen test verileri geri gönderimi yapılarak tekrar ZC702 kartına gönderilmiştir. Alınan veriler kontrol edilerek SGMII arayüzlerinin çalıştığı doğrulanmıştır. Daha sonra tek bir MAC kontrolcü ile üretilen Ethernet paketi kişisel bilgisayara bağlanılmıştır. Böylelikle MAC kontrolcü ve test örüntü üreteci bağımsız bir cihaz ile test edilmiştir. Son olarak benzetimleri ve uygulaması yapılan ağ anahtarı IP'sinin 4, 8 ve 16 kapılık sürümleri MAC, SGMII gibi diğer IP'lerin olduğu KC705 için

tasarlanan FPGA projelerine bağlanarak geri gönderim testleri yapılmış ve IS tasarımının iç testleri tamamlanmıştır.

Şekil 5.8. Test Sistemi Fotoğrafı.

Daha önce Şekil 5.3.a'da verilen blok şema, KC705 FPGA projesinin de blok şemasıdır. Şekil 5.9'de ise IS'nin performans testlerinde kullanılan ZC702 FPGA projesinin blok şeması verilmiştir. Bu gösterimlerde saat ve ilklendirme blokları verilmemiştir. ZC702 FPGA projesi yardımıyla KC705 üzerindeki IS'nin anahtarlama gecikmesi, tamponlama gecikmesi gibi kritik performans parametreleri ölçülmüştür. Test örüntüsü üreteci IP'si çok yüksek hızlı tümleşik devreler (İng. Very High Speed Integrated Circuit, VHSIC) için donanım tanımlama dilinde (İng. VHSIC Hardware Description Language, VHDL) yazılmıştır. Kaynak ve Hedef MAC adres bilgilerinin blok tasarım aşamasında parametrik olarak girildiği ve haberleşme tipi, özel operasyon kodu, operasyon kodu kontrol ve operasyon kodu verisinin giriş kapılarından okunduğu bir IP tasarlanmıştır. IP bu parametrelere göre test örüntüsü üretmektedir.

Test örüntüsü üretici IP'si tarafından seçilen yapılandırma ayarlarına göre üretilen Ethernet paketinin yapısı aşağıda Şekil 5.10'de gösterilmiştir. Tasarlanan IP yine VHDL dilinde yazılan bir test kodu ile test edilmektedir.

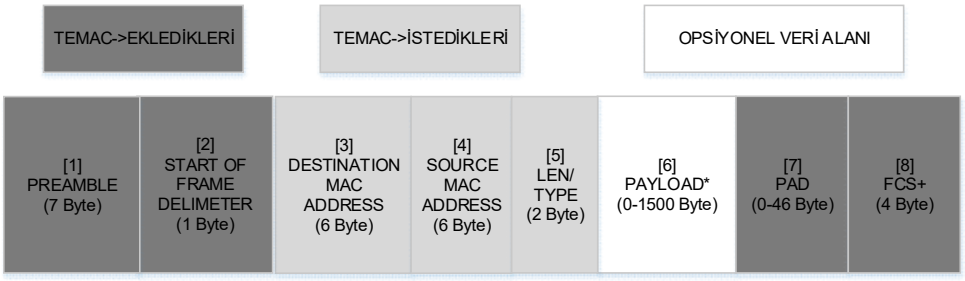


Şekil 5.9. ZC702 FPGA Tasarımı Blok Şeması.

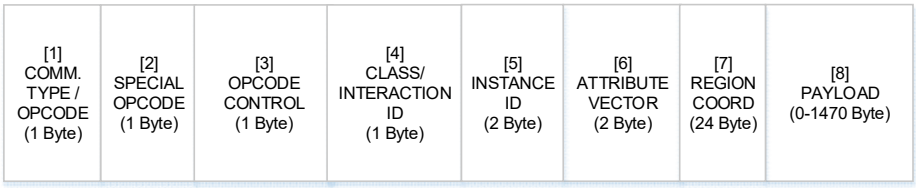
Şekil 5.10'de verilen paket yapısını daha detaylı inceleyecek olursak:

- TEMAC Eklentileri: Başlama eki 7-Byte 0x55'den oluşmaktadır. Başlangıç belirteci 1-Byte 0xD5'den oluşmaktadır. Doldurulmuş bölüm gönderilen çerçevenin en az 64-Byte olduğundan emin olmak için kullanılır. Çerçeve doğrulama kelimesi gönderilen mesajın fiziksel ortamdan doğru geçtiğini test etmek için kullanılır. Bu alanlar seçilen yapılandırma ayarlarına göre TEMAC tarafından doldurulmaktadır.
- Hedef MAC Adresi: Tek noktaya gönderimde Ethernet paketi içindeki verinin gönderileceği kapının MAC adresini belirtir. TEMAC'e gelen Ethernet çerçevesinde bulunması gerekmektedir.
- Kaynak MAC Adresi: Ethernet paketinin geldiği MAC adresini belirtir. TEMAC'e gelen Ethernet çerçevesinde bulunması gerekmektedir.

- Uzunluk/Tip: Kapıdan giren paketin uzunluğunu veya tipini belirtmek için kullanılır. TEMAC'e gelen Ethernet çerçevesinde bulunması gerekmektedir.
- Haberleşme Tipi: Tasarlanan ağ anahtarı IP'sinin desteklediği haberleşme tiplerinden birisi girilir. Ağ anahtarı IP'si tek noktaya yayın, herkese yayın, asenkron hepsini azaltma ve yayın/abone sistemini desteklemektedir.
- Özel Operasyon Kodu: Kolektif haberleşme operasyonu seçildiği zaman hangi özel operasyonun yapılacağı seçilir. Ağ anahtarı IP'si minimum, maksimum ve lojik VE özel operasyonlarını destekler.
- Operasyon Kodu Kontrol: Seçilen özel operasyon kodu için kullanılacak veri tipinin seçilmesini sağlar. İşaretli işaretsiz tamsayı ve kayan noktalı sayılar desteklenmektedir.
- Sınıf/Etkileşim Kimliği: Abone/yayın mesajında gönderilen sınıfın/etkileşimin kimlik bilgisidir.
- Örnek Kimliği: Abone/yayın mesajında gönderilen örneğin kimlik bilgisidir.
- Özelik Vektörü: Abone/yayın mesajında gönderilen sınıfın özelik vektörüdür.
- Bölge Koordinat: Abone/yayın sisteminde X,Y,Z koordinatlarını belirtir.
- Yük: Seçilen özel operasyon kodu için kullanılacak veri dizisidir.



*TEMAC jumbo ethernet çerçeveleri ile 1518B'dan büyük veri alabilir.
+FCS toplam Byte uzunluğuna bakar. Minimum Byte uzunluğu 18 iken maksimum doldurma 46B olur.
- Ethernet çerçevesi = PRE + SFD + DA + SA + LEN/TYPE + Opsiyonel veri + PAD+FCS
[1] PRE: 7B-> 0x55 (pg051, page 83)
[2] SFD: 1B-> 0xD5 (pg051, page 83)
[3] DA: 6B (pg051, page 83)
[4] SA: 6B (pg051, page 83)
[5] LEN/TYPE: 2B (pg051, page 83)
[6] PAYLOAD: Opsiyonel veri; maksimum 1492B normal çerçeve, jumbo çerçevede limit yoktur (pg051, page 83)
[7] PAD: Maksimum 46B (pg051, page 83)
[8] FCS: Çerçeve kontrol kelimesi 4B (pg051, page 83)



[1] Haberleşeme Tipi (Opcode): Unicast (UCA), Broadcast (BRO), Asynchronous Reduce All (ARA), Subscribe Object Class Attribute Message (SOCA), Update Attribute Value Message (UAV)
[2] Özel Operasyon Kodu: Minimum, Maximum, Logic AND
[3] Operasyon Kodu Kontrol: Unsigned integer, signed integer, single-precision floating-point
[4] Sınıf/Etkileşim Kimliği: En değerli bit sınıf veya etkleşim durumunu gösterir. Diğer bitler kimliği gösterir.
[5] Örnek Kimliği: Örneğin kimlik bilgisini gösterir.
[6]Özelik Vektörü: Özelik vektörü sınıf ile ilgili özelikleri göstermek için kullanılır. Her bit bir özelliğe karşılık gelmektedir.
[7] Bölge Koordinatları: (X1, X2) (Y1, Y2) (Z1, Z2) Noktaları yayın veya abone bölgelerini tanımlamak için kullanılır.
[8] Yük: Operasyon kontrol veya Ethernet çerçeve verisi.

Şekil 5.10. Ethernet Paket Yapısı.

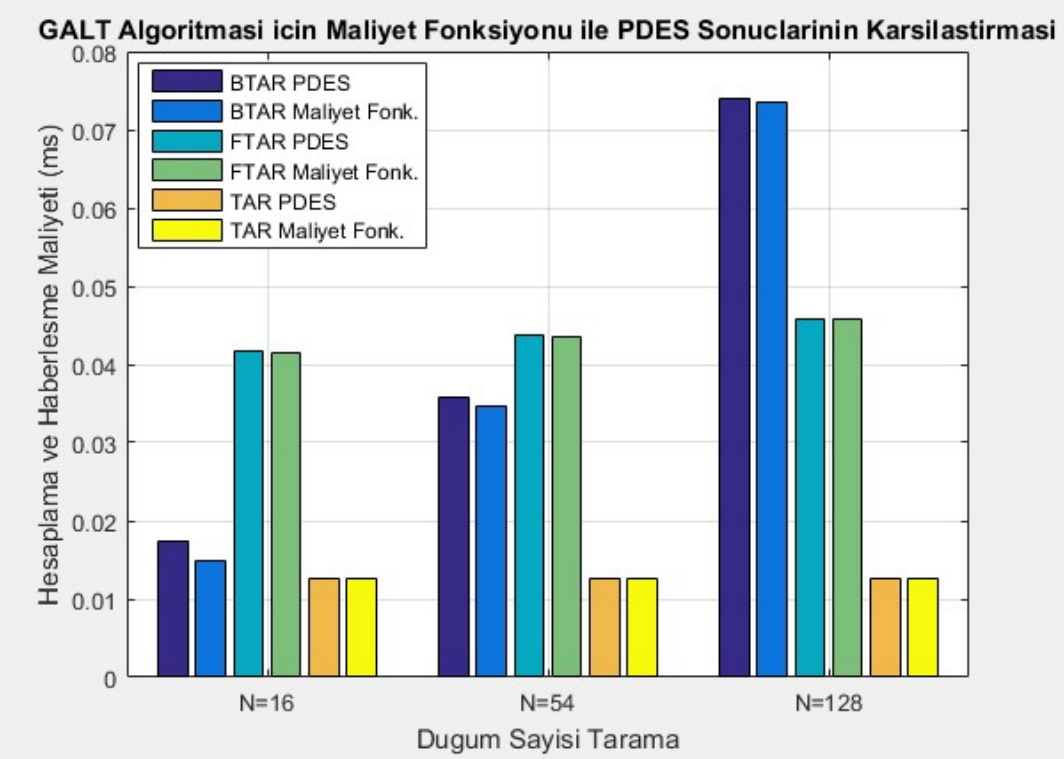
6. BENZETİM ÇALIŞMALARI

Bu bölümde ilk olarak PDES kullanılarak gerçekleştirilen benzetimler yardımıyla daha önce ağda trafik olmayan durum için çıkarılan maliyet fonksiyonları doğrulanmıştır. IS ile CS'nin trafik altında karşılaştırmasını gerçekleştirmek için ise PDES kullanılarak çeşitli senaryolarda testler yapılmıştır.

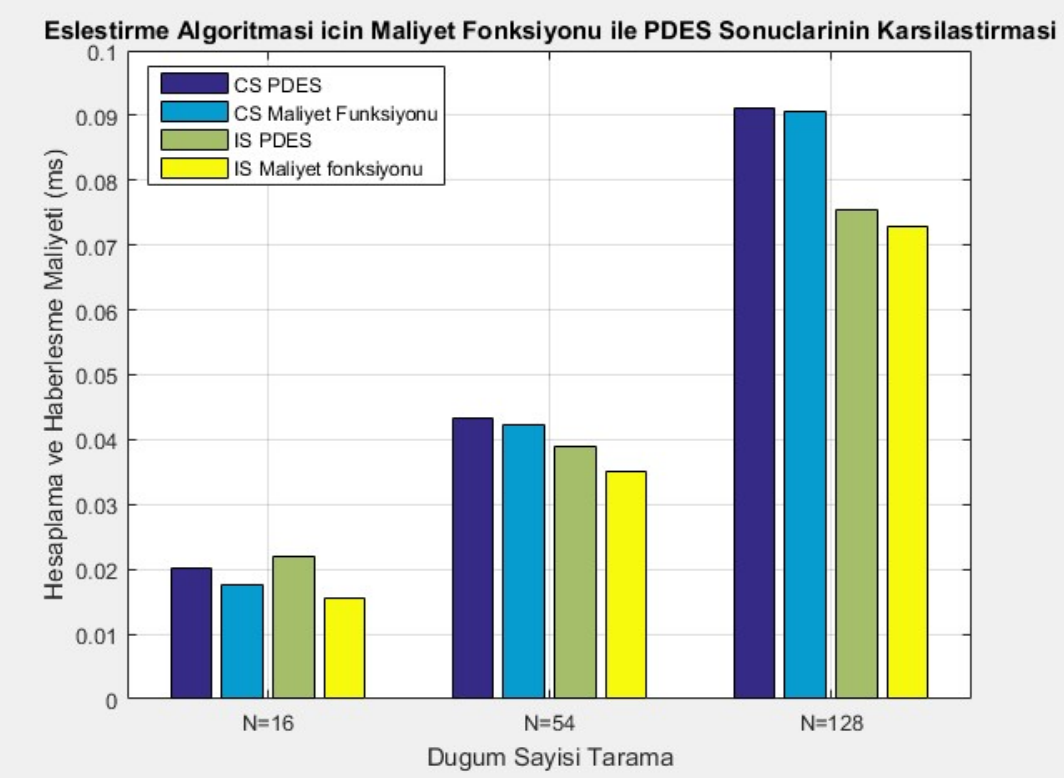
6.1 Paralel Ayrık Olay Benzetici ile Maliyet Fonksiyonlarının Doğrulanması

PDES olarak paralel ayrık olay benzetimlerinde kullanılmak üzere tasarlanan çeşitli ağ anahtarlarının performansının incelenebileceği bir yazılım kullanılmıştır [4]. PDES şişman ağaç topolojisinde tanımlı bir ağda verilen Ethernet anahtarı kapı sayısı için maliyet fonksiyonu ölçebilme şansı vermektedir. PDES ile mesaj ilklendirme zamanını (α_1, α_3) Alfa_Cost fonksiyonu içinde, her adımdaki anahtar gecikmesini Switch_Cost fonksiyonu içinde ve mesaj uzunluğu ile link bant genişliğinin çarpımı olan iletim gecikmesini ise Transmission_Cost içinde girilmesini sağlamaktadır. Mesajların düğümde işlenmesi için geçen zaman ise HLA_callback fonksiyonları içinde eklenebilmektedir. Bu maliyet değerleri IS veya CS'nin belirli gecikme değerlerinin yapılandırılıp farklı trafik yüklerinde performanslarının karşılaştırılması için kullanılabilecektir. Farklı gecikmeler BTAR, FTAR, TAR, SOCA ve UAV mesajlarında ortaya çıkacaktır. MPI kütüphanesi tabanlı PDES yazılımı sayesinde düğümlerde koşacak yazılımlar geliştirilebilmektedir. Aşağıda, hesaplanan maliyet fonksiyonları PDES'de trafik olmadığı durumda 4, 6 ve 8 kapılı Ethernet anahtarları için doğrulanmıştır.

Matlab 2015b'de analiz edilen hesaplama ve haberleşme maliyet fonksiyonları, Microsoft Visual Studio 2022 üzerinde çalıştırılan PDES test sonuçları ile karşılaştırılmıştır. Testler Ethernet anahtarlarının 4, 6 ve 8 kapısı için yapıldığından dolayı düğüm sayıları 16, 54 ve 128 olarak değişmektedir. Mesajının boyutu tüm testlerde 64Byte olarak alınmıştır. GALT ve eşleştirme algoritmasının tüm mesaj uzunlukları ve tüm düğüm sayıları için mesaj işleme süresi ölçümlerinde ZC702 geliştirme kartındaki ARM A9 kullanılmıştır. A9'da ölçülen süre beşe bölünmüştür. Çünkü HLA gibi benzetimlerin koştukları bilgisayarlar genelde 3GHz civarı çalışan ve test sistemimizde bulunan A9'un yaklaşık beş katı frekansta çalışan bilgisayarlardır. Şekil 6.1'de GALT algoritmasının maliyet fonksiyonu analizleri ile PDES sonuçları Şekil 6.2'de ise eşleştirme algoritmasının maliyet fonksiyonu analizleri ile PDES sonuçları verilmiştir.



Şekil 6.1. GALT Algoritması Maliyet Fonksiyonu ile PDES Sonuçları Karşılaştırması.



Şekil 6.2. Eşleştirme Algoritması Maliyet Fonksiyonu ile PDES Sonuçları Karşılaştırması.

Karşılaştırma sonuçları analizler ile benzetimin uyumunu ortaya koymuştur. PDES'de ölçülen gecikme değerlerinin analizlere göre daha yüksek çıkmasının sebebi ise daha önce anlatılan veri işleme hattının dolma gereksinimindendir ve ağ boyutu artıkça ihmal edilebilir bir değer olmaktadır. Analiz sonucunda GALT algoritmasında IS kullanımı ile elde edilen iyileşme FTAR için %69 ile %72 arasında değişmektedir. Eşleştirme algoritmasında ise bu iyileşme %10 ile %19 arasındadır. Testlerde IS performans iyileştirmesinin artan düğüm sayısından olumlu etkilendiği gözlenmiştir. Anahtar-tabanlı yaklaşım ile herkese yayın-tabanlı yaklaşım düşük ağ boyutlarında (N=16) GALT algoritması için yakın maliyet sonuçları vermekte iken ağ büyüdükçe herkese yayın-tabanlı yaklaşımın maliyeti artmaktadır. Herkese yayın-tabanlı yaklaşımın büyük ağlarda iyi ölçeklenememektedir. Fakat küçük ağ boyutlarında şişman ağaç-tabanlı yaklaşıma göre daha iyi sonuç vermektedir. Anahtar-tabanlı ve şişman ağaç-tabanlı yaklaşımlar ağ büyüdükçe iyi ölçeklenebilmektedirler. Fakat anahtar-tabanlı yaklaşım Şekil 6.1'ye göre yaklaşık 3 kat daha hızlı çalışmaktadır. Eşleştirme algoritmasının performansı IS'de çok az daha iyi çıkmıştır. İyileşmenin temel sebebi düğümlerde koşan algoritmanın anahtara kaydırılması kaynaklıdır. CS'de abone bilgileri alındıkça yayıncılar eşleştirme algoritmasını çalıştırmakta ve sonucunda çoklu yayın grubu oluşturup CS'ye iletmektedir. IS'de ise abone bilgileri düğümlere iletilmemektedir. Yani düğümlerde abone bölgesi değişimi durumu dikkate alınmaz. Bunun sonucu olarak sık abonelik değişimi durumu IS için bir performans kaybına sebebiyet vermeyecektir. Fakat tersi durum CS için geçerli değildir. Abone bölge değişim sıklığı CS'nin performansını doğrudan kötüleştirecektir. Bu bölümde yapılan analiz ve performans çalışmalarında abone bölge değişim sıklığının etkisi ayrıca incelenmemiştir. Bir sonraki bölümde PDES üzerinde çeşitli senaryolar altında IS ve CS karşılaştırılmıştır.

6.2 PDES Senaryo Testleri

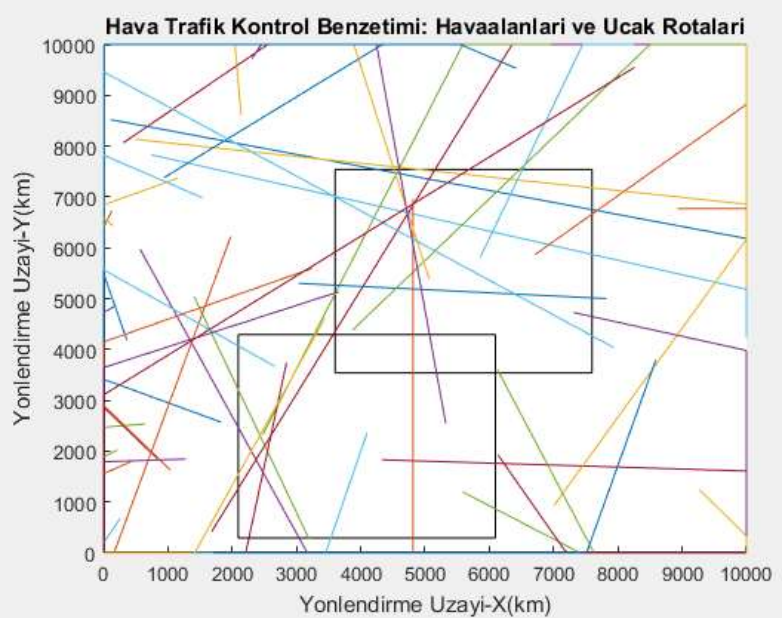
IS performansı farklı trafik yükleri yaratan senaryolar altında test edilmiştir. Test edilen ilk senaryo hava trafik kontrol uygulaması ikinci senaryo ise molekül dinamiği uygulamasıdır. Testlerde IS ile CS performansı karşılaştırılırken yine düğümlerde ARM A9'un beş katı hızlı PE'lerin bulunduğu durum ele alınmıştır. Benzetimler düğümlerde koşan yazılımın herhangi bir sınıfın herhangi bir özelliğine abone olabilecekleri abone (SOCA) veya yayın (UAV) mesajı ile başlar. Zaman ilerleme isteği (TAR) mesajı ile devam eder. TSO yığınındaki mesajların tüketilmesi (RAV) ve zaman ilerleme onayı

(TAG) mesajının alınması ile benzetim tamamlanır. Benzetim zamanı belirlenirken ilk olarak GALT değeri hesaplanır. Her bir federe ilerletilmek istenen zamana eşit ya da daha küçük zamana sahip, zaman bağımlı mesajları TSO yığınından alıp tüketir. Son olarak ise benzetim zamanı belirlenir. Zaman ilerletme isteği otomatik GALT hesabını başlatır. Performans ölçümleri ise yayın mesajı ile başlar zaman ilerleme onayı ile biter. Aşağıda bu senaryolar sonuçları ile birlikte detaylı anlatılmıştır.

6.2.1 Hava Trafik Kontrol Senaryosu

Bu senaryo referans çalışma [50] ile olabildiğince uyumlu olacak şekilde kurgulanmıştır. Senaryoda abone olarak iki farklı hava alanı bulunmaktadır. Uçaklar hava alanlarının etrafında rastgele rotalar ile uçmaktadırlar. Benzetimin yönlendirme alanı referans senaryo gibi 10000km x 10000km'dir. Birinci havaalanı (AP1) bitiş koordinatları (0,0) noktasına referans ile rastgele olarak 2000km ile 4500km arasında değişmektedir. İkinci havaalanı (AP2) benzer şekilde üretilmektedir. Fakat bu sefer AP2'nin referans noktası (10000,0) olur. Böylelikle AP1 ve AP2 birbirlerine en fazla 1000km yaklaşabilirler. Havaalanlarının maksimum radar menzili 4000km'dir. Referans yayında z ekseni de benzetime katılmıştır. Fakat bu benzetimde kolaylık sağlamak için z ekseni çıkarılmıştır. Benzetimde 2 düğüm abone havaalanları olarak tanımlanmaktadır. Ağın büyüklüğüne göre kalan düğümler benzetimdeki uçakları canlandırmaktadır. Yayıncı uçaklar rasgele bir hız ve uçuş açısı ile oluşturulurlar. Uçaklar her benzetim adımında pozisyonlarını yayınlarlar. Abone havaalanları ise eşleştirme algoritmasının sonucuna göre uçaklardan pozisyon mesajını alırlar. Uçaklar bir benzetim adımında maksimum 100km yer değiştirirler. Toplamda ise benzetim adımı 3 ile 300 arasında değişmektedir.

Havaalanlarının radar menzili ile uçakların uçuş rotalarının 300 benzetim adımı için rastgele oluşturulduğu durum 54 düğüm için Şekil 6.3'de gösterilmiştir. Havaalanı radar menzili kare ile uçakların rotası ise çizgi ile gösterilmiştir. Aslında gerçek radarların menzili kare olmaz daire olur. Fakat eşleştirme algoritmalarının daha hızlı hesaplayabilmesi için kare ile işlem yapılıp radar sinyali ile bir işlem yapılması gerektiğinde tekrar gerçek denklemeler kullanılmaktadır. Şekil 6.3'de toplam 52 adet uçak bulunmaktadır. Eğer herhangi bir uçak benzetim alanının sınırına ulaşırsa sınırı takip eder.



Şekil 6.3. 54 Düğüm için Hava Trafik Kontrol Uygulaması.

Çizelge 6.1'de IS ve CS'nin karşılaştırma sonuçları verilmiştir. IS ve CS performansını doğru karşılaştırabilmek için sabit oluşturulmuş senaryo kayıt edilip aynı sabit senaryo iki anahtar içinde kullanılmaktadır. Senaryo adım sayısı 3 olarak alınmıştır. Düğüm sayıları 16, 54 ve 128'dir. Mesaj boyutları ise 64, 256 ve 1024B'dir. Mesaj boyutu normalde TAR ve UAV mesajlarında 64B iken bu örnekte UAV mesaj boyutu artırılmış ve arka arkaya uçağın ardışık konumları mesaj içerisine yerleştirilmiştir. Böylece mesaj boyutu arttıkça eşleştirme algoritması da ardışık olarak çalışabilecektir. Çizelge 6.1'de düğüm sayısı artıkça iyileşme artmamaktadır. Bunun sebebi ise düğüm sayısı artıkça yayın/abone oranı arttığı için oluşan trafik nedeniyle elde edilen iyileşmenin etkisinin azalmasıdır. Ayrıca mesaj boyutu artıkça küçük mesaj boyutlarında iyileşme artışı olurken 64B mesaj boyutundan sonra iyileşme artışı olmamaktadır. Bunun sebebi ise 64B altında tek eşleştirme yapılırken daha uzun mesajlarda 28B'da bir eşleştirme yapılmasıdır. Hava trafik kontrol uygulamasında iyileştirme minimum %6 maksimum %20 olarak bulunmuştur. Sonuçlardan gözlendiği üzere yayın/abone mesaj oranı düşük iken düşük mesaj boyutlarında bir iyileşme elde edilmiştir. Bu durum ise PE'de yapılan bir işlemin ardından ağa bir yayın gönderilip arkasından zaman ilerlemesi yapılan küçük parçalı benzetimlere karşılık gelmektedir. Büyük parçalı (İng. Coarse-grained) benzetimlerde ise PE'de yapılan işlem ve yayınlar daha fazla olup zaman ilerleme istekleri daha seyrek yayın/abone oranları ise daha yüksektir.

Çizelge 6.1. Hava Trafik Kontrol Uygulaması için PDES Test Sonucu

Mesaj Boyutu (Bytes)	CS (µs)			IS (µs)			İyileşme (%)		
	Düğüm Sayıları			Düğüm Sayıları			Düğüm Sayıları		
	16	54	128	16	54	128	16	54	128
64	34	92	209	28	85	197	18	8	6
256	129	368	829	103	330	781	20	10	6
1024	503	1457	3303	404	1313	3122	20	10	6

6.2.2 Molekül Dinamiği Benzetim Senaryosu

Atomların ve moleküllerin 3 boyutlu uzayda fiziksel hareketi sonucu oluşan etkileşimler molekül dinamiği (MD) benzetimi ile incelenir. Parçacıkların konum ve hız bilgileri kullanılarak her benzetim adımında parçacıklara etkiyen kuvvetler hesaplanır. Newton'un hareket kanununa göre de bu kuvvetler kullanılarak parçacıkların yeni konum ve hız bilgileri bulunur. MD benzetimi metallerin yapısal ve dinamik özelliklerinin hesaplanması gibi uygulamalarda kullanılır.

MD benzetiminde çeşitli paralel yöntemler kullanılmaktadır. Bunların en kolay uygulananı atom ayrıştırma yöntemidir. Bu yöntemde benzetimi yapılacak parçacıklar tüm işlemcilere eşit olarak dağıtılır. Tüm işlemciler diğer tüm atomların konum bilgilerini takip eder. Bunun sonucu olarak ağ üzerinden sürekli diğer atomların konum bilgilerini alma ve bu bilgileri hafızada saklama ihtiyacı ortaya çıkar. Atom sayısının az olduğu küçük uygulamalarda tercih edilir. Bir diğer yöntem olan kuvvet ayrıştırma yönteminde ise atomlar arasındaki etkileşim kuvveti işlemcilere dağıtılır. Bu yöntemde her işlemci sadece üzerindeki etkileşimleri hesapladığı için işlem yükü daha azdır. Orta ölçekli uygulamalarda tercih edilmektedir. Alan ayrıştırma yönteminde ise tüm uzay eşit parçalara bölünür ve bu parçaların her birisinin sorumluluğu bir işlemciye verilir. Her işlemci sadece kendisine atanan hücreye komşu hücreler ile haberleşeceği için haberleşme yükü daha azdır. Parçacıklar hareketli olduğu için parçacıkların sahipliğinin değişmesi gerekmektedir. HLA sahiplik değişimini desteklediği için alan ayrıştırma yönteminde kullanılmaya uygundur. Fakat sahiplik değişimi, yazılan kodun karmaşıklığını artırmaktadır. Ağ trafiğini azaltmasından dolayı büyük ölçekli uygulamalarda tercih edilen bir yöntemdir [7], [62]. Bu çalışmada kullanılan MD senaryosunda alan ayrıştırma yöntemi kullanılmıştır.

Bu senaryoda ilk olarak olası bir MD benzetiminde ortaya çıkacak haberleşme trafiği oluşturulmaya çalışılmıştır. MD benzetiminde tüm moleküller 3 boyutlu uzayda birbirleri ile bağlantı içindedirler. Bu 3 boyutlu uzayda benzetimi yapılan moleküller iki boyutlu uzayda tüm düğümlerin Torus bağlantı ile birbirlerine bağlı olduğu ve ayrıca karşılıklı kenarlar haricinde dört köşenin de birbirine bağlı olduğu ağ topolojisi ile ifade edilmektedir. Bu ağda her bir federe atomların bulunabileceği bir alanı göstermektedir. Bu analizde tüm federeler önce bir abonelik isteğinde bulunup sonra arka-arkaya değişen sayıda yayın yapabilmekte ve en son zaman ilerleme isteğinde bulunmaktadır. Performans karşılaştırmalarında süre ölçümü yine yayın mesajı ile başlayıp benzetim zamanın belirlenmesi ile bitirilmektedir. Benzetim döngüsü adım sayısı 1, 10 ve 100 için testler yapılmıştır. Bu sayılar yayın/abone mesaj (UAV/SOCA) oranları içinde aynıdır. PDES test sonuçları Çizelge 6.2'de verilmiştir.

Çizelge 6.2. Molekül Dinamiği Benzetimi için PDES Test Sonucu.

Yayın/Abone Sayıları	CS (µs)			IS (µs)			İyileşme (%)		
	Düğüm Sayıları			Düğüm Sayıları			Düğüm Sayıları		
	16	54	128	16	54	128	16	54	128
1	23	26	28	13	14	13	46	48	53
10	54	60	63	44	44	44	19	27	30
100	428	430	432	412	412	412	4	4	5

Yayın/abone oranına göre performans iyileşmesi; oran 1 olduğunda %46 ile %53 arasında değişirken, oran 100 olduğu durumda %4'e düşmektedir. Buna göre yayın mesajının abone mesajına oranının eşit olduğu, daha küçük parçalı benzetimlerde IS yine daha iyi sonuç vermektedir. Ayrıca ağ boyutu artışında aynı yayın/abone oranı için IS'in performans iyileşmesi beklendiği gibi artmaktadır.

Yapılan ARM A9 testlerinde GALT algoritmasında hesaplama gecikmesinde en iyi iyileştirme oranı 2000 düğüm ve 2048B mesaj için 60300 olarak yakalanmıştır. Bu durumda CS kullanılarak ARM A9 işlemci içeren düğümlerde GALT işlemi BTAR için 60.3ms sürerken IS kullanıldığında TAR için 1us sürmektedir. Eşleştirme algoritmasında hesaplama gecikmesinde en iyi iyileştirme oranı 118888 olarak yakalanmıştır. Bu durumda CS kullanılarak ARM A9 işlemci kullanıldığında 107ms sürmektedir. IS kullanıldığında 0.9us sürmektedir. GALT algoritmasında işlemci yaklaşık 60ms'lik bir

ekstra süre kazanmakta iken eşleştirme algoritması yaklaşık 107ms'lik bir ekstra süre kazanmaktadır.

Çizelge 6.3'de 16 çekirdekli bir iş istasyonunda OpenCV kullanılarak 1920x1080 çözünürlüğünde bir resme uygulanan görüntü işleme algoritmaları ve algoritmanın çalışması için geçecek zaman verilmiştir [63]. Son 2 sütunda ise GALT algoritması ve eşleştirme algoritması ile sağlanan iyileşme sayesinde bu algoritmalardan kaç defa yapmak için süre kazanıldığı verilmiştir. Karşılaştırmalarda iş istasyonunda tek iş parçacığı çalıştırıldığı durum ele alınmıştır. Bu sonuçlardan gözlendiği üzere GALT algoritmasında kazanılan süre sayesinde en kötü durumda fazladan 1, eşleştirme algoritmasında ise fazladan 2 Canny kenar algılama (İng. Edge detection) algoritması koşulabilecektir.

Çizelge 6.3. GALT ve Eşleştirme Algoritması ile Kazanılan Zamanların Değerlendirmesi.

Algoritma\İyileştirme	Süre(ms)	GALT	Eşleştirme
Smoothing (Yumuşatma)	20,9	2	5
Erosion (Aşındırma)	6,6	9	16
Image Pyramid (Resim Piramidi)	8,5	7	12
Sobel Derivate Operator (Sobel Türev Operatörü)	28,1	2	3
Canny Edge Detection (Canny Kenar Algılama)	41,4	1	2

7. SONUÇLAR VE YORUMLAR

Bu çalışmada ortaya konulan anahtar-tabanlı yaklaşım ile geliştirilmiş GALT ve eşleştirme algoritmalarının IS üzerinde, HLA'nın TM ve DDM servislerinin performansını iyileştirdiği gösterilmiştir. Bu algoritmalara ait maliyet fonksiyonları çıkarılıp, çıkarılan bu maliyet fonksiyonları PDES ile doğrulanmıştır. Analizler ve testler 16, 54 ve 128 düğüm ile 64B-1024B arasında mesaj boyutu için yapılmıştır. PDES testleri ile analizler sonuçları uyumlu bulunmuştur. Yapılan analizler sonucunda 64B mesaj uzunluğu için IS ile CS'nin maliyet fonksiyonları karşılaştırılmış ve ağda trafik olmayan durumda GALT algoritması için %72'ye varan, eşleştirme algoritması için ise %19'a varan bir iyileşme bulunmuştur. PDES kullanılarak çeşitli senaryolarda farklı trafik yüklerinde yapılan testlerde, hava trafik kontrol uygulamasında en fazla %20 ve molekül dinamiği benzetim uygulamasında ise en fazla %53 seviyesinde performans iyileşmeleri gözlenmiştir. Hava trafik kontrol uygulamasında düğüm sayısı artıkça yayın mesaj yoğunluğunun abone mesaj yoğunluğuna oranı arttığı için iyileşme eğiliminde bir azalma gözlenmiştir. Bu durum ise IS'nin küçük parçalı benzetimler için daha uygun olduğunu ortaya koymaktadır. Molekül dinamiği benzetim uygulamasında yayın mesaj yoğunluğunun abone mesaj yoğunluğuna eşit olduğunda %53 performans iyileşmesi elde edilirken yayın mesaj yoğunluğu arttıkça iyileşme %4'e kadar düşmektedir. IS'nin molekül dinamiği benzetim senaryosunda, yine ağ boyutu artması durumunda CS'ye göre daha düşük gecikme ile çalıştığı gösterilmiştir. Bu çalışmada elde edilen sonuçlar da yine IS'nin küçük parçalı benzetimler için daha uygun olduğu yönündedir. Ayrıca anahtar-tabanlı GALT algoritmasının maliyet fonksiyonu düğüm sayısına ve anahtar kapı sayısına bağımlı olmadığı için alternatif yaklaşımlara göre daha iyi ölçeklenebilmektedir. Ölçeklenebilir IS; PE'lerde ki işlem yükünü ortadan kaldırması, GMD ve DSD avantajları yanında daha düşük gecikme ile çalışarak benzetim performansını artırmaktadır. Fakat buna karşılık anahtar kapı maliyetine %10'luk bir artış getirmektedir.

IS'nin DDM servisi uygulamasında abone mesajlarının düğümlere çıkarılmaması sayesinde aslında abone bölge bilgilerinin sık güncellediği uygulamalarda avantaj sağlayacağı aşikârdır. Bu çalışmada IS üzerinde abone mesajları için mesajlarının düğümlere çıkarılmaması haricinde bir iyileştirme yapılmadığı için dinamik abone bölgesi güncelleme durumu analiz edilmemiştir. Fakat abone mesajlarının maliyetlerinde

bir iyileştirme yapmak mümkündür. IS'ye eklenecek donanım ile tüm abone istekleri için bölge kaydı tutulabilir. Aynı sınıf kimliğinde gelen abone istekleri için bölge kaydını güncelleyerek en geniş bölge bilgisi için abone istekleri saklanabilir. Böylece abone bölge bilgisi ise sadece abone bölgesi genişledikçe üsteki anahtarlara aktarılarak ağ trafiği azaltılabilir.

Bu çalışmada sakla ve ilet yöntemi ile çalışan bir Ethernet anahtarı yerine kestirme (İng. Cut-through) yöntemi ile çalışan bir Ethernet anahtarı kullanılmış olsaydı performansın mesaj boyutuna çok bağlı olması engellenmiş olur ve gecikme değerleri çok daha düşük bir Ethernet anahtarı ortaya çıkarılabilirdi. Fakat kestirme yönteminde mesajların tamponlanmamasının yanında paket kontrol işlemi de yapılmamaktadır. Bu durum ise güvenilirlik açısından problem yaratacağı için tercih edilmemiştir.

Zaman ilerleme isteği gönderen abone zaman ilerleme onayı gelene kadar boşta bekleyeceği için GALT algoritmasının hızlı çalışması benzetim sistemi performansını artırmada yüksek öneme sahiptir. Ayrıca ağda kalan geçiş mesajları için IS üzerinde ortaya konulan yaklaşım ile sunulan çözüm, benzetimde geri alma işlemlerini tamamen ortadan kaldıracağı için dolaylı olarak bir performans iyileşmesi sağlayacaktır. Sonuç olarak bu çalışmada ortaya konulan, anahtar-tabanlı GALT ve eşleştirme algoritmaları kullanılan ve ağda geçiş mesajı kalmasına izin vermeyen ölçeklenebilir IS, çok düğümlü ağlarda HLA tabanlı paralel ayrık olay benzetim sistemlerinin küçük parçalı uygulamalarının performansının artırılmasını sağlamıştır.

8. KAYNAKLAR

[1] F. Yilmaz, U. Durak, K. Taylan, H. Oğuztüzün, Adapting Functional Mockup Units for HLA-compliant Distributed Simulation, Proc. 10th Int. Model. Conf. March 10-12, 2014, Lund, Sweden. 96 (2014) 247–257. https://doi.org/10.3384/ecp14096247.

[2] S. Borkar, Thousand core chips - A technology perspective, Proc. - Des. Autom. Conf. (2007) 746–749. https://doi.org/10.1109/DAC.2007.375263.

[3] G. Heiser, Many-Core Chips — A Case for Virtual Shared Memory, Proc. Work. Manag. Many-Core Syst. (MMCS '09). (2009) 5–8. http://www.cercs.gatech.edu/mmcs09/.

[4] K.M. Imre, Simulating the Programmable Networks for Hla Compatible High-Performance Simulators, Proc. - Eur. Counc. Model. Simulation, ECMS. 2022-May (2022) 291–295. https://doi.org/10.7148/2022-0291.

[5] J.W. Hollenbach, Inconsistency, neglect, and confusion; A historical review of DoD distributed simulation architecture policies, Simul. Interoperability Stand. Organ. - Spring Simul. Interoperability Work. 2009. (2009) 531–536.

[6] T. Çelik, Yüksek Düzeyli Mimari İçin Bir Yazılım Modelleme Aracı, Hacettepe Fen Bilim. Enstitüsü. (2005).

[7] T. Erçelebi, Yüksek Düzeyli Mimarinin, Kosut İslem Gereksinimi Olan Bilimsel Benzetim Uygulamalari İçin Uygunlugunun İncelenmesi, Hacettepe Fen Bilim. Enstitüsü. (2008).

[8] I.C. Society, 1516.1-2010 - IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)-- Object Model Template Specification, 2010.

[9] J.S. Dahmann, High Level Architecture for Simulation, (1997) 9–14.

[10] I.C. Society, 1516.1-2010 - IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)-- Federate Interface Specification, IEEE, 2010. https://doi.org/10.1109/IEEESTD.2010.5557728.

[11] D.M. and S.O. Department of Defense, RTI 1.3-Next Generation Programmer's Guide Version 5, (2002) 122.

[12] S.J.E. Taylor, X. Wang, S.J. Turner, J. Ladbrook, Investigating Distributed Simulation at The Ford Motor Company, (2005).

[13] X. Cai, H. Li, J. Yang, Research on HLA/TM in GNSS system, Int. Conf. Signal Process. Proceedings, ICSP. 2015-Janua (2014) 2403–2406.

https://doi.org/10.1109/ICOSP.2014.7015425.

[14] M.G. Seok, T.G. Kim, C.B. Choi, D. Park, An HLA-Based Distributed Cosimulation Framework in Mixed-Signal System-on-Chip Design, IEEE Trans. Very Large Scale Integr. Syst. 25 (2017) 760–764. https://doi.org/10.1109/TVLSI.2016.2594948.

[15] C. Sung, T.G. Kim, Framework for Simulation of Hybrid Systems : Interoperation of Discrete Event and Continuous Simulators Using HLA / RTI, (2011).

[16] H. Georg, S. Member, S.C. M, S. Member, C. Rehtanz, S. Member, C. Wietfeld, S. Member, Analyzing Cyber-Physical Energy Systems : The INSPIRE Cosimulation of Power and ICT Systems Using HLA, 10 (2014) 2364–2373.

[17] R. Bottura, D. Babazadeh, K. Zhu, A. Borghetti, L. Nordström, C.A. Nucci, SITL and HLA Co-simulation Platforms : Tools for Analysis of the Integrated ICT and Electric Power System, (2013) 918–925.

[18] R.E. De Grande, A. Boukerche, Dynamic Partitioning of Distributed Virtual Simulations for Reducing Communication Load, (2009).

[19] R.E. De Grande, Distributed Dynamic Balancing of Communication Load for Large-Scale HLA-based Simulations †, (n.d.) 1109–1114.

[20] R.E. De Grande, A. Boukerche, H. Mohamed, S. Ramadan, Decreasing Communication Latency through Dynamic Measurement , Analysis , and Partitioning for Distributed Virtual Simulations, 60 (2011) 81–92.

[21] R.E. De Grande, A. Boukerche, R. Alkharboush, Time Series-Oriented Load Prediction Model and Migration Policies for Distributed Simulation Systems, 28 (2017) 215–229.

[22] P. Haghi, A. Skjellum, M.C. Herbordt, Workload Imbalance in HPC Applications : Effect on Performance of In-Network Processing, (2021).

[23] H. Tajbakhsh, R. Parizotto, M. Neves, A. Schaeffer-Filho, I. Haque, Accelerator-Aware In-Network Load Balancing for Improved Application Performance, 2022 IFIP Netw. Conf. IFIP Netw. 2022. (2022). https://doi.org/10.23919/IFIPNetworking55013.2022.9829787.

[24] R. (School of C.S.& E. Fujimoto, G.I. of Technology), PARALLEL AND DISTRIBUTED SIMULATION, Suparyanto Dan Rosad (2015. 5 (2020) 248–253.

[25] D.Y. Hong, F.P. Pai, S.H. Lo, Y.C. Chung, A Scalable HLA RTI System Based on Multiple-FedServ Architecture, in: 2010 12th Int. Conf. Comput. Model. Simul., IEEE, 2010: pp. 527–532. https://doi.org/10.1109/UKSIM.2010.102.

[26] P. Pacheco, An Introduction to Parallel Programming, 2011. https://doi.org/10.1016/C2009-0-18471-4.

[27] O. Arap, L.R.B. Brasilino, E. Kissel, A. Shroyer, M. Swany, Offloading Collective Operations to Programmable Logic, IEEE Micro. 37 (2017) 52–60. https://doi.org/10.1109/MM.2017.3711654.

[28] O. Arap, L.R.B. Brasilino, E. Kissel, A. Shroyer, M. Swany, Offloading Collective Operations to Programmable Logic on a Zynq Cluster, IEEE 24th Annu. Symp. High-Performance Interconnects Offloading. (2016). https://doi.org/DOI 10.1109/HOTI.2016.13.

[29] R.L. Graham, D. Bureddy, P. Lui, H. Rosenstock, G. Shainer, G. Bloch, D. Goldenerg, M. Dubman, S. Kotchubievsky, V. Koushnir, L. Levi, A. Margolin, T. Ronen, A. Shpiner, O. Wertheim, E. Zahavi, Scalable Hierarchical Aggregation Protocol (SHArP): A Hardware Architecture for Efficient Data Reduction, Proc. COM-HPC 2016 1st Work. Optim. Commun. HPC Runtime Syst. - Held Conjunction with SC 2016 Int. Conf. High Perform. Comput. Networking, Storage Anal. (2017) 1–10. https://doi.org/10.1109/COMHPC.2016.006.

[30] M.G. Venkata, G. Bloch, G. Shainer, R. Graham, Accelerating OpenSHMEM collectives using in-network computing approach, Proc. - Symp. Comput. Archit. High Perform. Comput. 2019-Octob (2019) 212–219. https://doi.org/10.1109/SBAC-PAD.2019.00042.

[31] B. Ramesh, K.K. Suresh, N. Sarkauskas, M. Bayatpour, J.M. Hashmi, H. Subramoni, D.K. Panda, Scalable MPI Collectives using SHARP: Large Scale Performance Evaluation on the TACC Frontera System, Proc. ExaMPI 2020 Exascale MPI Work. Held Conjunction with SC 2020 Int. Conf. High Perform. Comput. Networking, Storage Anal. (2020) 11–20. https://doi.org/10.1109/ExaMPI52011.2020.00007.

[32] K. Lakhotia, F. Petrini, R. Kannan, V. Prasanna, In-network reductions on multi-dimensional HyperX, (2021).

[33] Y. Liu, J. Zhang, S. Liu, Q. Wang, W. Dai, R. Chak, C. Cheung, Scalable Fully Pipelined Hardware Architecture for In-Network Aggregated AllReduce, IEEE Trans. Circuits Syst. I Regul. Pap. 68 (2021) 4194–4206. https://doi.org/10.1109/TCSI.2021.3098841.

[34] J. Ma, D. Dong, C. Li, K. Wu, L. Xiao, PAARD: Proximity-Aware All-Reduce Communication for Dragonfly Networks, 19th IEEE Int. Symp. Parallel Distrib.

Process. with Appl. 11th IEEE Int. Conf. Big Data Cloud Comput. 14th IEEE Int. Conf. Soc. Comput. Netw. 11th IEEE Int. (2021) 255–262. https://doi.org/10.1109/ISPA-BDCloud-SocialCom-SustainCom52081.2021.00045.

[35] K.T. Pham, T.T. Nguyen, H. Yamaguchi, Y. Urino, M. Koibuchi, Scalable Low-Latency Inter-FPGA Networks, Proc. - 2022 IEEE 36th Int. Parallel Distrib. Process. Symp. IPDPS 2022. (2022) 234–245. https://doi.org/10.1109/IPDPS53621.2022.00031.

[36] X. Jin, X. Li, H. Zhang, R. Soulé, J. Lee, N. Foster, C. Kim, I. Stoica, NetCache: Balancing Key-Value Stores with Fast In-Network Caching, SOSP 2017 - Proc. 26th ACM Symp. Oper. Syst. Princ. (2017) 121–136. https://doi.org/10.1145/3132747.3132764.

[37] K.S. Perumalla, A.J. Park, V. Tipparaju, GVT algorithms and discrete event dynamics on 129K processor cores, 18th Int. Conf. High Perform. Comput. HiPC 2011. (2011) 1–11. https://doi.org/10.1109/HiPC.2011.6152725.

[38] B.K. Görür, K.M. İmre, H. Oğuztüzün, L. Yilmaz, Predetermined Rollbacks: An extension to Time Warp for spatially parallel agent-based simulation, Simul. Model. Pract. Theory. 95 (2019) 60–77. https://doi.org/10.1016/j.simpat.2019.04.008.

[39] S.C. Tay, Y.M. Teo, Performance optimization of throttled time-warp simulation, Proc. IEEE Annu. Simul. Symp. (2001) 211–218. https://doi.org/10.1109/simsym.2001.922134.

[40] A. Boukerche, C. Dzermajko, K. Lu, Alternative approaches to multicast group management in large-scale distributed interactive simulation systems, Futur. Gener. Comput. Syst. 22 (2006) 755–763. https://doi.org/10.1016/j.future.2006.02.001.

[41] E.S. Liu, G.K. Theodoropoulos, An Approach for Parallel Interest Matching in Distributed Virtual Environments, (2009). https://doi.org/10.1109/DS-RT.2009.34.

[42] A. Boukerche, K. Lu, Optimized Dynamic Grid-based DDM Protocol for Large-scale Distributed Simulation Systems ∗, (n.d.) 17–22.

[43] R. Ayani, F. Moradi, G. Tan, Optimizing Cell-size in Grid-Based DDM, (2000) 1–8.

[44] G. Tan, R. Ayani, Y. Zhang, F. Moradi, Grid-Based Data Management in

Distributed Simulation, (n.d.) 1–7.

[45] S.H. Lo, C.A. Chiu, F.P. Pai, D.Y. Hong, Y.C. Chung, MGRID: A modifiable-grid region matching approach for DDM in the HLA RTI, Spring Simul. Multiconference 2009 - Co-Located with 2009 SISO Spring Simul. Interoperability Work. (2009).

[46] G. Tan, Y. Zhang, R. Ayani, A hybrid approach to data distribution management, (2000) 55–61.

[47] K. Pan, J.T. Stephen, W. Cai, Z. Li, A dynamic sort-based DDM matching algorithm for HLA applications, ACM Trans. Model. Comput. Simul. 21 (2011). https://doi.org/10.1145/1921598.1921601.

[48] Y. Jun, C. Raczy, G. Tan, Evaluation of a Sort-based Matching Algorithm for DDM, (2002) 68–75.

[49] K. Pan, S.J. Turner, W. Cai, Z. Li, An Efficient Sort-Based DDM Matching Algorithm for HLA Applications with a Large Spatial Environment, (2007).

[50] G. Tan, L. Xu, S. Taylor, An Agent-based DDM for High Level Architecture *, (2001) 75–82.

[51] S.H. Lo, Y.C. Chung, Offloading Region Matching of Data Distribution Management with CUDA, (2010) 306–311. https://doi.org/10.1109/ISMS.2010.64.

[52] A. Santoro, R.M. Fujimoto, Offloading Data Distribution Management to Network Processors in HLA-Based Distributed Simulations, 19 (2008) 289–298.

[53] K. Imre, N. Sevim, The High Level Architecture (HLA) on photonic torus: Hardware and software co-design, Proc. - 8th EUROSIM Congr. Model. Simulation, EUROSIM 2013. (2015) 550–554. https://doi.org/10.1109/EUROSIM.2013.97.

[54] K.M. İmre, C. Baransel, H. Artuner, Efficient and scalable routing algorithms for collective communication operations on 2D all-port torus networks, Int. J. Parallel Program. 39 (2011) 746–782. https://doi.org/10.1007/s10766-011-0169-2.

[55] M. Al-Fares, A. Loukissas, A. Vahdat, A scalable, commodity data center network architecture, Comput. Commun. Rev. 38 (2008) 63–74. https://doi.org/10.1145/1402946.1402967.

[56] S. Kumar, L. V Kal, Scaling All-to-All Multicast on Fat-tree Networks, (n.d.).

[57] I. American National Standards Institute, ANSI/VITA 65-2010 (R2012) OpenVPX™ System Specification, (2012). http://www.vita.com.

[58] Micrel, Micrel KS8999 Integrated 9-Port 10/100 Switch with PHY and Frame Buffer, (2005). http://www.micrel.com.

[59] Microsemi, VSC7442-02, VSC7444-02, VSC7448-02, and VSC7449-02 Datasheet Family of L2 / L3 Enterprise Gigabit Ethernet Switches with 10 Gbps Links, (2019). www.microsemi.com.

[60] Xilinx, 1G/2.5G Ethernet PCS/PMA or SGMII v15.0, (2015). https://www.xilinx.com/.

[61] Xilinx, Tri-Mode Ethernet MAC v9.0, LogiCORE IP Product Guide, (2016). https://www.xilinx.com/.

[62] K.J. Bowers, E. Chow, H. Xu, R.O. Dror, M.P. Eastwood, B.A. Gregersen, J.L. Klepeis, I. Kolossvary, M.A. Moraes, F.D. Sacerdoti, J.K. Salmon, Y. Shan, D.E. Shaw, Scalable algorithms for molecular dynamics simulations on commodity clusters, Proc. 2006 ACM/IEEE Conf. Supercomput. SC'06. (2006). https://doi.org/10.1145/1188455.1188544.

[63] S. Matuska, R. Hudec, M. Benco, The comparison of CPU time consumption for image processing algorithm in Matlab and OpenCV, Proc. 9th Int. Conf. ELEKTRO 2012. (2012) 75–78. https://doi.org/10.1109/ELEKTRO.2012.6225575.

9. EKLER

9.1 EK 1 - Tezden Türetilmiş Yayınlar

O.V. Karaca, K.M. İmre, A.Z. Alkar, Network Accelerator for Parallel Discrete Event Simulations, J. Supercomput. (2022) 1–23. https://doi.org/https://doi.org/10.21203/rs.3.rs-2398540/v1 (Submitted)