SANAL KABLOSUZ DUYARGA AĞ TABANLI BİR DAĞITIK-PARALEL SİBER FİZİKSEL SİSTEM TASARIMI

Güngör YILDIRIM

Doktora Tezi
Bilgisayar Mühendisliği Anabilim Dalı
Danışman: Prof. Dr. Yetkin TATAR
KASIM-2017

T.C.
FIRAT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

SANAL KABLOSUZ DUYARGA AĞ TABANLI BİR DAĞITIK-PARALEL SİBER FİZİKSEL SİSTEM TASARIMI

DOKTORA TEZİ

Güngör YILDIRIM

(131129201)

Anabilim Dalı: Bilgisayar Mühendisliği

Programı: Donanım

Danışman: Prof. Dr. Yetkin TATAR

Tezin Enstitüye Verildiği Tarih: 07 Kasım 2017

KASIM-2017

T.C.
FIRAT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

SANAL KABLOSUZ DUYARGA AĞ TABANLI BİR DAĞITIK-PARALEL SİBER FİZİKSEL SİSTEM TASARIMI

DOKTORA TEZİ

Güngör YILDIRIM

(131129201)

Tezin Enstitüye Verildiği Tarih : 07 Kasım 2017
Tezin Savunulduğu Tarih : 30 Kasım 2017

Tez Danışmanı :	**Prof. Dr. Yetkin TATAR (F.Ü)**
Diğer Jüri Üyeleri :	**Prof. Dr. M. Ali AKÇAYOL (Gazi Ünv.)**
	Prof. Dr. Ali KARCI (İnönü Ünv.)
	Doç. Dr. Galip AYDIN (F.Ü)
	Yrd. Doç. Dr. Mustafa KAYA (F.Ü)

KASIM -2017

ÖNSÖZ

Bu tez çalışması, Fırat Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı Doktora Programı'nda hazırlanmıştır.

İnternetin yaygınlaşması ile "Nesnelerin İnterneti (Internet of Things - IoT)" ve farklı iletişim ağları üzerinden haberleşebilen kompleks "Siber Fiziksel Sistemler (Cyber Physical Systems-CPSs)", geleceğin şekillenmesinde önemli rol oynayacaklardır. Bu sistemler genellikle bünyelerinde hetorejen yapılar içermektedir. Bu heterojen yapılardan biri ise son yirmi yılda, akademik ve endüstriyel alanda kendine önemli yer bulan "Kablosuz Duyarga Ağları (KSA,WSNs)" dır. WSN 'lerin klasik çalışma mantıklarının dışına çıkarak daha paylaşılabilir olmaları ise WSN sanallaştırma teknikleri ile mümkün olmuştur. WSN 'lerin doğası gereği sahip olduğu kısıtlar dikkate alındığında ise ne tür bir kaynak paylaşımının yapılaması gerektiği ayrı bir önem kazanmaktadır. IoT/CPS sistemlerin ölçeği, istemci ve bu sistemler içinde yer alan veri kaynaklarının sayısı dikkate alındığında oluşacak büyük verilerin işlenmesi ise bir başka önemli konu başlığını oluşturmaktadır. Bu tez çalışmasında, heterojen WSN kaynaklar içeren IoT/CPS sistemlerde WSN 'ler arası kaynak paylaşımı, düğüm-düğüm etkileşimi ve büyük veri yönetimi yapabilen bir WSN sağlayıcı mimarisi tasarımı ve bu mimari temelinde sağlayıcı sistemin gerçekleştirilmesi açıklanmıştır.

Bu çalışmanın yapılmasında yaptığı yönlendirme ve katkılardan dolayı değerli hocam sayın **Prof.Dr. Yetkin TATAR'**a, tez izleme komite üyelerim sayın **Doç.Dr. Galip AYDIN** 'a, sayın **Yrd.Doç.Dr. Mustafa KAYA** 'ya teşekkür ve minnetimi özellikle belirtmek istiyorum.

Ayrıca sağlamış olduğu teknik desteklerinden dolayı sayın **Bilgisayar Yük. Müh. İrfan KILIÇ '**a teşekkürlerimi sunarım.

Maddi ve manevi desteklerini yıllarca benden esirgemeyen başta değerli eşim sayın **Bilgisayar Yük. Mühendisi Suna YILDIRIM'**a, evlatlarıma ve tüm aile efradıma teşekkürü bir borç bilirim.

Ayrıca " MF.16.19 - Sanal Kablosuz Duyarga Ağ Tabanlı bir Dağıtık-Paralel Siber Fiziksel Sistem Tasarımı" isimli proje desteği için Fırat Üniversitesi Rektörlüğü, Bilimsel Araştırma Projeleri Koordinasyon Birimine teşekkürlerimi sunarım.

Güngör YILDIRIM

Elazığ-2017

İÇİNDEKİLER

Numara

ÖZET

Bu tez, farklı konum ve teknolojilere sahip Kablosuz Algılayıcı Ağları (WSN 'ler) içeren "Internet of Things -IoT" ve "Cyber-Physical System- CPS" sağlayıcılarına yönelik yeni yaklaşımlar önermektedir. Bu amaçla, tez çalışmasında, etkileşimli bir WSN kaynak paylaşım mekanizması kullanan IoT/CPS sağlayıcı mimarisi ve bunun fiziksel uygulanması gerçekleştirilmiştir. Bugün IoT/CPS projeleri kapsamı içerisinde altyapı bileşenleri olarak farklı heterojen WSN teknolojileri kullanılabilmektedir. Bulut sistemler, akıllı şehirler ve akıllı şebekeler gibi büyük ölçekli IoT/CPS projelerinin çatısı altındaki WSN sistemler, birbirleri ile etkileşim kurarak veya verilerini merkezi bir IoT/CPS birimine göndererek farklı hizmetler sunabilmektedirler. Otonom heterojen yapıların dinamik olarak değişebileceği düşünüldüğünde, farklı WSN 'ler arasındaki M2M etkileşim, kaynak paylaşımı ve veri yönetimi hayati öneme sahiptir. Geleneksel WSN 'lerde kaynak paylaşım ve etkileşim mekanizmalarından olan WSN sanallaştırma teknikleri, IoT / CPS projelerinde yetersiz kalmaktadır. Buna ek olarak, geleneksel WSN sistemlerinde kullanılan veri depolama ve yönetim metotları IoT/CPS ölçeğinde verimsizdir. Bu bağlamda bu tez çalışması;

- IoT/CPS projeler kapsamındaki farklı heterojen WSN'ler arasında komut/sorgu geçişkenliğini sağlayarak, insan faktörü olmaksızın, WSN ağlara ait broadcast alanlarının genişletilebilmesi ve kaynak paylaşımının sağlanması,
- İstemci tanımlı filtreleme ve analiz işlemlerinin sunucu tarafta yapılmasına olanak sağlayarak, sunucu kullanıcı arası trafiğin minimize edilmesine imkân verilmesi,
- İstemci ve alt WSN parametrelerine göre verimli kaynak ataması yapılabilmesi,
- Sunucu tarafın zaman ve şart kısıtlarının kullanıcı uygulamaları için büyük ölçüde kaldırılması,
- Oluşabilecek büyük verilerin, gereksiz/fazlalık veri kontrolü ile optimize edilmesi
- Tüm sağlanan özelliklerin, bir bulut teknolojisi ile ölçeklenebilir ve sürdürülebilir olması

için alternatif yöntemler kullanan bir IoT/CPS WSN bulut sağlayıcı sistem mimarisi önermektedir. Önerilen sistemin doğruluğu ve uygulanabilirliği, gerek simülasyon gerekse gerçek sistemler üzerinde çalıştırılması ile ispatlanmıştır.

Anahtar Kelimeler: WSN, IoT, CPS, WSN sanallaştırma, bulut sistemler

SUMMARY

DESIGN OF A DISTRIBUTED-PARALLEL CYBER PHYSICAL SYSTEM BASED ON VIRTUAL WIRELESS NETWORK

This thesis proposes new approaches towards "Internet of Things-IoT" and "Cyber-Physical System-CPS" providers, which include Wireless Sensor Networks (WSNs) with different location and technologies. For this purpose, an IoT/CPS provider architecture using an interactive WSN resource-sharing mechanism and its physical implementation are performed. IoT/CPS projects may use heterogeneous WSN technologies as infrastructure components within their scope. The WSN systems under the roof of big scale IoT/CPS projects such as cloud systems, smart cities, and smart grids can provide different services by either interacting with each other or sending their data to a central IoT/CPS unit which collects and evaluates the data. When it is considered that the autonomous heterogeneous structures may change dynamically, it is clear that M2M interaction, resource sharing and data management between the different WSNs have paramount importance. WSN virtualization techniques, which are important resource-sharing and interaction mechanisms in WSNs, remain incapable in the IoT/CPS projects. In addition, the data storage and management methods used in traditional WSN systems are inefficient in IoT/CPS scale. In this context, the thesis study provides the following advantages;

- Without any human intervention, an extensible broadcast area and resource sharing mechanism through providing code/command and algorithm transitivity
- Minimum data traffic between server and clients by enabling client-defined filtering and algorithmic analysis on server side
- An alternative resource selection/allocation algorithm, which considers both the client and the sub-WSN parameters at the same time
- An optimum big data storage, management and sharing mechanism
- A scalable and sustainable cloud system that involves these all features

The accuracy and success of the proposed system have been shown by both simulation studies and real system implementations

Keywords: WSN, IoT, CPS, WSN virtualization, cloud system

ŞEKİLLER LİSTESİ

Numara

TABLOLAR LİSTESİ

Numara

SEMBOLLER LİSTESİ

$\boldsymbol{W_k}$: k 'nıncı WSN sistem

$\mathbf{d_i}$: WSN sistem düğümü

$\mathbf{T}$: Sensör Tip Kümesi

$\mathbf{t_j}$: Sensör Tipi

Ë : Kaynak varlık kümesi

$\mathbf{k_j^{d_n}}$: Düğüm n 'deki j 'ninci kaynak durum değişkeni

$\mathbf{B}$: Kaynaklar veri boyutu matrisi

$\mathbf{b_j}$: Kaynak j için üretilen veri boyutu

$\mathbf{N_s}$: WSN ağ kaynak durum matrisi

$\mathbf{f}$: Ağ çalışma oranı

$\mathbf{D}$: WSN ağ veri üretim boyutu

$\mathbf{E^{Tx}}$: Transmisyon enerji tüketimi

$\mathbf{E^{Rx}}$: Paket alım durumu enerji tüketimi

Ɲ : Radyo devre sabitesi

Ɛ : Radyo devresi Yükselteç sabitesi

$\mathbf{d}$: Transmisyon mesafesi

$\boldsymbol{T_k}$: Klasik WSN bulut sisteminde değerlendirme zaman maliyeti

$\boldsymbol{T_f}$: FVWSN bazlı bulut sisteminde değerlendirme zaman maliyeti

$\boldsymbol{T_r}$: Maksimum kaynak erişim süresi

$\boldsymbol{T_{ro}}$: Sistem taraflı optimum maksimum kaynak erişim süresi

$\boldsymbol{R_{max}}$: Vnd başına maksimum yönetilebilir kaynak sayısı

$\boldsymbol{C_{sr}}$: Tek bir VNd kaynağı için maksimum okuma sayısı

$\boldsymbol{f_R^{VNd}}$: Maksimum yönetilebilir kaynak sonucu fonksiyonu

$\boldsymbol{f_C^{VNd}}$: Kaynak okuma sonucu fonksiyonu

$\boldsymbol{f_A^{VNd}}$: Aktuatör durum fonksiyonu

$\boldsymbol{T_{se}}$: Maksimum VNd yürütüm fonksiyonu

$\boldsymbol{\sigma_w}$: Zaman çerçeve toleransı

$\boldsymbol{T_{st}}$: Senkronlama için zaman sınırı

S : Alt WSN ‘ler kümesi

O : Alt WSN sahipleri

$\boldsymbol{R_s}$: Sensör Kaynaklar kümesi

$\boldsymbol{R_a}$: Aktuatör Kaynaklar Kümesi

$\boldsymbol{R_d}$: Diğer kaynaklar kümesi

$\boldsymbol{K_{T_l}}$: l ‘ninci kaynak tipi

$\boldsymbol{K_T}$: Ön tanımlı kaynak tip kümesi

$\boldsymbol{U_T}$: Uygulama tip kümesi

$\boldsymbol{g_{tip}}$: Alt kaynak tipi atama fonksiyonu

$\boldsymbol{N_i^{W_k}}$: Alt WSN k ‘daki i ‘ninci düğümün özellik notasyonu

$\boldsymbol{M_{d_i}}$: Düğüm i ‘nin işlem oranı

$\boldsymbol{E_{d_i}}$: Düğüm i ‘nin mevcut enerji miktarı

$\boldsymbol{L_{d_i}}$: Düğüm i ‘nin kullanım maliyeti

g : Kaynağın üretebileceği veri boyutu

m : Kaynağın işlem yükü

ϴ : Kaynak kullanım maliyeti

Ê : Kaynak notasyonu

a : Aktuatör kaynak erişim izin sayısı

r : Sistemde tanımlı kaynak notasyonu

ẗ : Kaynak notasyonunda uygulama tipi

$\{\boldsymbol{P_r}\}$: Kaynak çalışma zamanı kümesi

$\boldsymbol{R_{pos}}$: Kaynak koordinatı

$\boldsymbol{I_p}$: İstemci notasyonu

$\boldsymbol{I_p.\Phi}$: İstemci tipi

$\boldsymbol{V^{I_p}}$: İstemci VNd ‘si

$\boldsymbol{q_i^{W_k}}$: Alt WSN k’da tanımlı i ‘ninci sorgu/komut

$\boldsymbol{q_i^S}$: Sistemde tanımlı i ‘ninci sorgu/komut

$\boldsymbol{U}$: VNd uygulama notasyonu

$\boldsymbol{U_{id}}$: Uygulama kimlik numarası

$\boldsymbol{U_{tip}}$: Uygulama tip

$\boldsymbol{U_{alan}}$: Uygulama hedef alan

$\{P_u\}$: Uygulama çalışma periyodu

CI_i : Kaynak uygulama çalışma notasyonu

P_i^{Tr} : Düğüm iletim güç tüketimi

P_i^{Rc} : Düğüm alım güç tüketimi

f_{OYI} : Düğüm veri işleme güç tüketim fonksiyonu

β_1 : Radyo termal kayıp sabitesi

β_2 : Radyo ünitesi sabitesi

d_{ih} : Düğüm i ve h arası mesafenin

P_i^{tot} : Düğüm i 'nin toplam güç tüketimi

γ : Yol kayıp katsayısı

ρ : İletilen bit bajına enerji kaybı sabitesi

R_i^T : Düğüm i 'nin kapsama yarı çapı

P_{trs} : Paket alım RSSI eşik değeri

g_0 : Anten sabitesi

$I_{f_{hi}}$: Düğüm i ve h arasındaki veri akışı

$O_{f_{ih}}$: Düğüm i ve h arası yönlendirme veri akışı

G : Kaynak üretim sınır katsayısı

$Ŧ_{WS}^i$: Alt WSN kaynağından veri edinim süresi

$Ŧ_{SI}^i$: Verinin ilgili istemciye iletilmesi süresi

$Ŧ_y$: İstemci toplam yürütüm süresi

$Ŧ_{IW}^j$: Aktuatör kaynak yürütüm süresi

d_{ih}^{wrst} : Düğüm i ve h arası en uzak mesafe ihtimali

f_{Tse} : VNd güncel yürütüm süresi

Sg_i : Sanal görev i

$Sg_{N_{max}}$: N_{max} sanal görev talebinin kuyrukta olması olasılığı

λ : Ortalama talep gelme hızı

μ : Ortalama servis hızı

Sg_0 : Herhangi bir sanal görevin kuyrukta olmama ihtimali

$O_{sg}{}^s$: Sistemdeki Ortalama sanal görev sayısı

$B_{sg}{}^s$: Bir sanal görevin sistemde ortalama bekleme süresi

$B_{sg}{}^k$: Bir sanal görevin kuyrukta ortalama bekleme süresi

$\mathbf{D_T}$: Sistem toplam gecikme

$\mathbf{D_i}$: İstemci sistem arası gecikme

$\mathbf{D_p}$: VNd yürütüm gecikmesi

$\mathbf{D_{sp}}$: Sanal görev oluşturma gecikme süresi

$\mathbf{D_f}$: Kaynak okuma gecikmesi

$\mathbf{D'_E}$: Düğüm iç yürütüm gecikmesi

$\mathbf{E_{sense}}$: Algılama esnasında enerji tüketimi

$\mathbf{E_{proc}}$: Düğüm yürütüm enerji sarfiyatı

$\mathbf{K}$: Düğüm yaşam ömrü sabitesi

$\boldsymbol{LT_i}$: Düğüm i için yaşam ömrü

$\boldsymbol{H^{V^{I_p}}}$: Bir VNd için hata tekrar sayısı

$\boldsymbol{Rep_{V^{I_p}}}$: VNd için replikasyon notasyonu

$\boldsymbol{t_{V^{I_p}}}$: VNd 'nin OperationRunner yürütüm zamanı

$\boldsymbol{t_{Sg_{V^{I_p}}}}$: VNd 'nin kuyruk yürütüm zamanı

$\boldsymbol{Ol_{W_k}}$: WSN ağ hata olma olasılığı

$\mathbf{J(.,.)}$: Jaccard benzerlik fonksiyonu

$\boldsymbol{\partial}$: Veri benzerlik eşik değeri

$\boldsymbol{v_i}$: Büyük veri bileşeni

KISALTMALAR LİSTESİ

WSN : Wireless Sensor Networks
KAA : Kablosuz Algılama Ağları
KSA : Kablosuz Sensör Ağları
IoT : Internet of Things
CPS : Cyber Physical System
SFS : Siber Fiziksel Sistemler
GSM : Global System for Mobile Communications
VWSN : Virtual Wireless Sensor Networks
SKSA : Sanal Kablosuz Sensör Ağları
DBS : Düğüm Bazlı Sanallaştırma
ABS : Ağ Bazlı Sanallaştırma
ATBS : Arakatman Temelli Bulut Sanallaştırma
EOS : Embedded Operation System
SD : Sanal Düğüm
VNd : Virtual Node
RFID : Radio Frequency Identification
NFC : Near Field Communication
UID : Universal/Unique Identifier
WISP : Wireless Identification and Sensing Platforms
IPSO : IP for Smart Objects
M2M : Machine to Machine
RPL : IPv6 Routing Protocol for Low-Power and Lossy Networks
6LoPAN : IPv6 over IEEE 802.15.4
CoAP : Constrained Application Protocol
EXI : Efficient XML Interchange Format
HTML : Hypertext Markup Language
XML : Extensible Markup Language
HTTP : Hyper-Text Transfer Protocol
IETF : Internet Engineering Task Force

ETSI : European Telecommunications Standards Institute
W3C : World Wide Web Consortium
SOAP : Simple Object Access Protocol
ReSTFul : Representational state transfer web services
NSF : National Science Foundation
PHY : Physical Layer
MAC : Media Access Control Layer
ISM : Industrial Scientific Medical band
RF : Radio Frequency
BE : Basit Entegrasyon
KE : Kompleks Entegrasyon
MW : Middleware
ÖABS : Örtüşen Ağ Bazlı Sanallaştırma
KABS : Kümesel Ağ Bazlı Sanallaştırma
PUB/SUB : Publish / Subscribe Middleware
MENO : Managed Ecosystems of Networked Objects
TCP : Transmission Control Protocol
UDP : User Datagram Protocol
ASF : Apache Software Foundation
HDFS : Hadoop Distributed File System
YARN : Yet Another Resource Negotiator
AYÇ : Adaptör Yazılım Çerçevesi
WSNSS : WSN Sağlayıcı Servisleri
KTD : Kaynak Tanımlama Dökümanı
RDD : Resource Description Document
SensorML : Sensor Markup Language
FVWSN : Firat Virtual Wireless Sensor Network
IDE : Integrated development environment
KOA : Kullanıcı Operasyon Arayüzü
UOI : User Operations Interface Class
KFS : Kullanıcı Fonksiyon Sınıfları
UFA : User Functions Abstract Class
SFI : Server Functions Interface

ID : Identification

API : Application Programming Interface

RSIn : Resource Interface

SerOP : Service Operator

MIPS : Million Instructions per Second

RSSI : Received signal strength indication

RPC : Remote Procedure Call

TCS : Time Clock Simulation

DES : Discrete Event Simulation

OPNET : Optimized Network Engineering Tools

FSM : Finite State Machine

KOOS : Kritik Olay Olma Süresi

SYNC : Senkron Mod

NoSYNC : Senkron Mod Pasif

AMQP : Advanced Message Queuing Protocol

JSF : Jaccard Similarity Function

1. GİRİŞ

1.1. Genel Bilgi

Kablosuz Sensör Ağlar (WSN, KAA, KSA), algılama veya kontrol edebilme yeteneğine sahip birçok düğümün belirli kablosuz iletişim standart ve protokollerine göre birlikte çalışmasının sağlandığı bir kablosuz ağ yapısıdır [1-2]. Fiziksel ortam ile siber dünya arasında önemli bir köprü olan WSN 'ler insan-çevre veya cihaz-cihaz-çevre etkileşimini kolaylaştırıcı çok önemli bir rol oynarlar. Günümüzde gelişimini devam ettirmekte olan Nesnelerin interneti (Internet of Things -IoT) ve Siber Fiziksel Sistemlerinin (CPS, SFS) de temel taşı olan WSN 'ler spesifik çalışma karakteristiklerine sahiptir [3-4]. WSN düğümler, beslendikleri bataryalarla ve uygun çalışma düzenleri ile aylarca hatta yıllarca sorunsuz çalışabilecek bir şekilde yapılandırılabilirler. Askeri alanlardan, medikal projelere, akıllı şehir, akıllı tarım gibi özel uygulamalardan güvenlik projelerine kadar sayısız alanda tesis edilen bu ağlar da klasik ağlar gibi katmanlı bir ağ modeline göre yapılandırılırlar [5-8]. Fakat doğaları gereği, klasik kablosuz ağlara kıyasla daha kısıtlı enerji kaynaklarına, kısa menzilli iletişime, düşük iletişim hızına, düşük işlemci ve RAM kapasitesine sahiptirler. İlk WSN sistemler daha çok belli bir amaca yönelik tasarlanmıştır. Bu tip klasik sistemler merkezi bir birim tarafından kontrol edilmekte ve toplanan veriler yine bu merkezi birime gönderilerek değerlendirilmekte idi. WSN sistemlerin çevresel veri toplama alt yapısı olarak kullanılabilir olması, bilim ve endüstriyel alanda kısa sürede yaygınlaşmasına yardımcı oldu. Pek çok üniversite, araştırma grubu ve ticari kuruluş kendi uygulama alanlarına bağlı olarak bu teknolojiden faydalanmak istedi ve bu amaçla kendi WSN dizaynlarını oluşturdu. Esnek yapısından dolayı hem donanımsal hem de yazılımsal olarak üzerinde birçok farklı varyasyonlarının oluşturulmasına izin veren bu teknoloji, sonuçta pek çok farklı versiyonları ile günümüz teknoloji dünyasında kendine önemli bir yer edindi. Gömülü sistem teknolojilerinin baş döndürücü bir hızla gelişmesi ile WSN sistemlerin etkinliği bir kat daha artmıştır.

Bunun yansıra pek çok cihazın dahili sensörler içermesi, teknolojik nesnelerin "akıllı" sıfatı ile anılmasına olanak sağlamıştır. Akıllı telefonlar, akıllı saatler, akıllı televizyonlar, akıllı sayaçlar gibi teknolojik tanımlamalar, iletişim alt yapılarındaki teknolojik gelişmeyle birlikte akıllı binalar, akıllı enerji sistemleri, akıllı şehirler gibi daha büyük ve karmaşık

sistem konseptlerinin doğmasına ön ayak oldu [9-11]. Bu konseptlerin kendi içlerinde veya birbirleri ile etkileşmesi ise daha genel iki kavram olan "Nesnelerin İnterneti - IoT" ve "Siber Fiziksel Sistemler - SFS/CPS" teknolojilerini doğurdu. IoT, internet altyapısını kullanabilen cihazların bu altyapı üzerinden birbirleri ile entegre olabilmelerine ve daha farklı sistem grupları oluşturabilmelerine imkân sağlayan teknolojidir. Siber fiziksel sistemler (CPS) ise fiziksel dünya ile siber ortam arasındaki etkileşimleri sağlamak üzere genellikle mühendislik temelli uygulamalar için geliştirilen karmaşık sistemler bütünüdür. Bünyesinde otonom sistemleri (farklı gömülü sistemler, kontrol sistemler vb), farklı teknolojileri (PC, tablet PC, akıllı telefon) ve farklı protokol yapıları kullanan ağ teknolojileri (Ethernet, GSM, 802.15.4 vb.) barındırabilir. CPS, internet altyapısına uyumlu olmakla beraber interneti kullanma gibi bir zorunluluğu bulunmamaktadır [12-14].

IoT ve CPS konseptlerinin ortaya çıkmasıyla WSN 'ler daha etkileşimli ve paylaşılabilir sistemler haline dönüşmeye başladı. Bu etkileşim ve paylaşımla, sayısız sensör teknolojisi ortaya muazzam bir veri toplama havuzu çıkardı. Bu da farklı disiplinlerin daha az maliyet ve zaman tüketimi ile daha çok bilgiye ulaşmasına olanak tanıdı. WSN kaynakları buna ek olarak, aktuatör kaynak içeren WSN 'lerin de bu değişime dahil olması ile birlikte daha etkin kontrol sistemleri teknoloji dünyasında boy göstermeye başladı. Özellikle WSN sanallaştırma teknikleri kullanılarak oluşturulan sanal WSN sistemlerin [15-30] geliştirilmesi ile WSN kaynak paylaşılabilirliği daha verimli bir hale geldi. Sanal WSN 'ler (VWSN,SKSA), istemcilerin kendi belirledikleri veya ihtiyaç duydukları kaynaklara her hangi bir alt yapı bilgisine ihtiyaç duymadan ulaşabilmesini sağlayan mantıksal yapılardır. Bu tip esnek WSN sistemlerinin diğer teknolojiler ile işbirliği yapabilmesi bilimsel ve endüstriyel dünyanın yanında ticari alanda da yeni sektörlerin oluşmasına olanak sağladı. Örnek olarak, akıllı şehir ve akıllı enerji sistemleri gibi kompleks sistemler sensör/aktuatör ağların entegrasyonları ile bir çok farklı hizmetin aynı proje altında sunulmasına olanak sağlayabilmektedir. Sonuç olarak VWSN 'ler gibi yeni kabiliyetlere sahip olan WSN sistemler, hem IoT hem de CPS sistemlerin vazgeçilmez bileşenlerinden biri oldu.

Büyük ölçekli IoT ve CPS sistemlerde elde edilen veya işlenen veri devasa boyutlara ulaşabilmektedir. Özellikle sayısı binlerle ifade edilebilen ve yüksek veri trafikli alt WSN bölgeleri içeren kompleks sistemler için veri boyutu, ciddi bir parametredir. Günümüz teknoloji dünyasında "Big Data (Büyük Veri)" başlığı altında incelenen bu kavram bilişim dünyasının farklı teknolojileri tarafından yürütülmekte ve geliştirilmektedir [31]. Büyük ölçekli, oldukça fazla kaynak/kullanıcı içeren veya sürekli veri akışlarının olduğu sistemler

büyük verilerin elde edildiği ortamlardır. Bu nedenle büyük veri işleme teknolojileri, büyük ölçekli IoT/CPS projelerinde önemli bir altyapı sağlamaktadır. Büyük verilerin depolanması, işlenmesi veya dağıtılmasının minimum maliyette ve maksimum verimde başarılması özellikle bu tip büyük projelerin ekonomik gerçekleştirilmesinde hayati rol oynamaktadır. Günümüz büyük veri işleme teknolojileri bu bakımdan tatmin edici özelliklere sahiptir. Burada önemli olan, ölçekleme ve sürdürülebilirlik gibi önemli özelliklerin dikkate alınarak sisteme dahil edilmesidir.

Bu tezin de odak noktası olan büyük ölçekli IoT-WSN veya CPS-WSN projelerinde her zaman ön plana çıkan belli başlı bazı temel zorluklar vardır. Bunlardan en önemlisi, bu ölçekteki projelerde farklı teknolojilerde ve standartlarda üretilmiş alt komponent veya alt sistemlerin bulunduğu "heterojen" bir yapının oluşabilmesidir [32]. Yukarıda da bahsedildiği gibi WSN sistemler gibi genel manada standartlaşamamış veya belli bir amaca yönelik tasarlanmış farklı tasarımlı sistemlerin bir proje çatısı altında birbirleri ile güvenilirlik ve sürdürülebilirlik parametrelerini dikkate alarak etkileşebilmesi teknoloji dünyasının üzerinde çalıştığı önemli bir sorunsaldır. Büyük ölçekli IoT/CPS projelerinin ana özelliklerinden biri olan kaynak paylaşımı, bu tip heterojen sistemler söz konusu olduğunda daha farklı bir şekilde irdelenmelidir. Alt sistemlerin donanımsal ve yazılımsal sınırları, birbirleri arasında ne tür bir iletişim altyapı teknolojisinin kullanılması gerektiği, gecikme karakteristikleri ve maliyet gibi daha pek çok soru bu tip büyük projelerin tasarımında mutlaka cevaplandırılmalıdır. Dünyanın farklı coğrafi bölgelerinde bulunan alt otonom sistemlerin kendi iç işleyişlerine zarar vermeden başka bir projenin de bir parçası olabilmesi, kaynaklarının hepsinin değil de belirli bir bölümünün paylaşılmasını istemesi, bu otonom sistem sahiplerinin bu ortak kullanımda ticari kar hesapları talep etmeleri de dikkate alınması gereken diğer olgulardır. Bununla birlikte bu heterojen sistemlerden elde edilen verilerin işlenmesi, veri trafiğinin kontrol edilebilmesi ve dışarıdan bakıldığında bu tip büyük ölçekteki bir heterojen sistemin istemcilere tek bir sistemmiş gibi görünmesi her zaman temel prensiptir. Bu ise genel kapsamlı bir "WSN sağlayıcı/sunucu sistem" ile başarılabilmektedir. WSN sağlayıcı sistemler, kaynak paylaşımına olanak sağlayan alt WSN sistemler ile bu kaynaklardan faydalanmak isteyen istemciler arasındaki etkileşimleri düzenleyen arabulucu sistemlerdir. Tez boyunca bu sistemler, "IoT/CPS WSN sağlayıcılar" olarak ifade edilecektir. Arka plandaki karmaşıklığın istemcilerden gizlenmesi, oluşacak problemlerin tolere edilmesi, güvenlik, abone ve maliyet yönetimi, veri depolanması/işlenmesi gibi kavramlar IoT/CPS WSN sağlayıcının sorumluluğundadır.

Büyük ölçekli IoT/CPS projeleri yöneten bu sağlayıcılar, heterojen sistemler arasındaki kaynak paylaşımını veya sistem karmaşıklığının istemciden soyutlanmasını genellikle "sanallaştırma" teknikleri ile yapmaktadırlar. Uzunca süredir bilişim dünyasının önemli kavramlarından olan sanallaştırma, özelikle zaman ve maliyet açısından büyük avantajlara sahiptir. Genellikle gelişmiş bilgisayar sistemlerinde kullanılan sanallaştırma kavramı söz konusu WSN 'ler olunca farklı bir boyutta değerlendirilmektedir. WSN 'lerin doğası gereği sahip oldukları düşük konfigürasyon özellikleri (kısıtlı hafıza, batarya temelli enerji kaynakları vb.) günümüz sanallaştırma tekniklerinin doğrudan WSN düğümler üzerinde uygulanmasını imkansızlaştırmaktadır. Bu nedenle WSN sanallaştırma teknikleri bilişim dünyasının diğer sanallaştırma tekniklerinden ayrılmaktadır. Farklı WSN teknolojilerinin varlığı dikkate alındığında WSN sanallaştırma tekniklerinin de farklılık arz edeceği açıktır. Bunun diğer bir anlamı, farklı teknolojili otonom WSN yapılar içeren büyük ölçekli IoT/CPS sistemler için bu problemin daha da karmaşık hale geleceğidir. Çünkü her WSN sistem belli bir standart üzerine olmayan farklı donanımsal veya yazılımsal özelliklere sahip olabilir ve bunlardan birinde uygulanabilen sanallaştırma yöntemi diğerlerine uygulanamayabilir. Dahası bu tip bir IoT/CPS sisteme dinamik olarak eklenip çıkabilen farklı coğrafyalardaki alt otonom WSN sistemler olabileceği düşünüldüğünde bu problemin boyutunun daha da artacağı bir gerçektir. Bu nedenle WSN'lerin de içinde bulunduğu IoT/CPS sistemleri yönetecek olan sağlayıcı sistemlerin kullanacakları sanallaştırma yöntemi/yöntemleri de en önemli parametreler arasında olacaktır. Buna ek olarak farklı kullanıcı profilleri, elde edilen verilerin anlık dağıtımı gibi fonksiyonlar için sağlayıcı sistemler uygun arakatman (middleware) teknolojilerini kullanmak zorundadırlar. Bu bağlamda ölçeklenebilirlik ve soyutlama kullanılacak arakatman yazılımının belirlenmesinde iki önemli parametredir.

Bu tez çalışmasında günümüzde kendini göstermeye başlayan ve geleceğin temel sistemlerinden olan, yapısında otonom WSN'ler içerebilen heterojen yapılı yeni bir IoT/CPS sağlayıcı sistem mimarisi önerilmiştir. Bu tür bir IoT/CPS WSN sağlayıcı ile

- Farklı teknolojili ve lokasyonlu WSN sistemlerin yazılımsal bir arakatman vasıtasıyla sağlayıcı üzerinden ön ayarsız bir şekilde etkileşebilmelerine olanak sağlayabilmek,
- Yazılımsal ara katmanın yardımıyla alternatif bir WSN sanallaştırma tekniği uygulayarak, istemcilerin kendilerine ait yönetimsel komutlarını/algoritmalarını bu

sağlayıcı üzerinde ön tanımsız bir şekilde çalıştırabilmek ve seçmiş oldukları fiziksel WSN kaynakları tam bir şeffaflıkla yönetebilmelerini sağlamak,

- Oluşabilecek büyük verilerin ise yönetilebilir olmalarını mümkün kılabilmek

amaçlanmıştır.

Bu tez kapsamında, ilk olarak önerilen mimariye ait oluşturulmuş model detayları açıklanacak, daha sonra sırasıyla bu mimariye uygun geliştirilen sağlayıcı sistemin simülasyonu ve fiziksel gerçekleştirme başarımları gösterilecektir. Tez aynı zamanda farklı simülasyon senaryolarına bağlı olarak en uygun sistem parametre seçiminin nasıl yapılabileceğine dair bir çözüm yöntemi de önermektedir.

Sisteme ait analitik analizler ve sistemin fiziksel gerçekleştirilmesi Fırat Üniversitesi Bilgisayar Mühendisliği Bölümü Kablosuz Sensör Ağları Laboratuvarında gerçekleştirilmiş olup simülasyon platformu olarak OPNET Modeler platformu [33] tercih edilmiştir.

1.2. Problemin Tanımı

Bu tez iki temel problem üzerine yoğunlaşmaktadır. Bunlardan birincisi, günümüzdeki birbirinden farklı WSN teknolojilerinin, IoT/CPS sistemler üzerinden birbirleri ile nasıl aktif etkileşimde olacakları ve nasıl tek bir sistem gibi davranabilecekleridir. Bu problemin en önemli çözümünde WSN sanallaştırma teknikleri esas alınmıştır. Bu amaçla alternatif ve etkin bir WSN sanallaştırma yaklaşımı önerilmiştir. İkinci problem ise bu tip bir etkileşime sahip büyük ölçekli bir IoT/CPS sistem çatısı altında oluşabilecek veri boyutunun nasıl yönetilebileceğidir.

Birinci problem, farklı sistemler arasında kaynak paylaşımı ve beraber çalışabilirlik zorlukları temeline dayalıdır. Bu tip problemlerin en etkin ve ekonomik çözümlerinden birinin sanallaştırma tekniklerinin olduğu bilişim dünyasının kabulüdür. Bu konuyla ilgili birçok çalışma literatürde kendisine yer bulmuştur. Detayları Bölüm 2 ‘de verilecek olan günümüz WSN sanallaştırma teknikleri bu tezde temel olarak üç başlık altında toplanmıştır. Bunlar, “Düğüm Bazlı Sanallaştırma -DBS”, “Ağ Bazlı Sanallaştırma-ABS” ve “Arakatman Yazılımı Temelli Bulut Sanallaştırma - ATBS” dir. DBS ve ABS sınıflandırması [15] ‘de yapılmıştır. Bu sınıflandırma modeli genel bulut sanallaştırma modellerini ihtiva etmemektedir. DBS ‘nın en önemli avantajı daha hızlı ve etkileşimli

kaynak paylaşımı, düğüm temelinde filtreleme ve veri işleme, sanal makine desteği sunmasıdır. Ancak öte yandan, bir WSN düğümün kısıtlı donanım özelliklerinden dolayı çok sayıda uygulamanın bir düğüm üzerinde çalışmasına izin vermemesi, düğümlere uzaktan yeni kod yüklenme problemi, eski teknolojilerin bu tip sanallaştırmayı desteklememeleri gibi kısıtlar IoT sistemlerde bu tip çalışmayı önemli ölçüde zorlaştırmaktadır. Diğer bir WSN sanallaştırma tekniği olan ABS 'nın başlıca avantajları; sağlayıcı WSN alt yapısı iyi bir şekilde istemciden soyutlaması, ölçeklenebilir olması ve sistem arızalarına karşı daha güvenli olmasıdır [15]. Bununla beraber, P2P iletişim için özel protokol kullanımlarına ihtiyaç duyulması ve bunun tüm WSN teknolojileri tarafından karşılanmasının imkansıza yakın olması en önemli dezavantajıdır [28-30]. ATBS günümüzde IoT sistemler içinde en çok kullanılan tekniklerdendir. Pub-Sub arakatman yazılımları gibi teknolojilerin tesis edildiği bu tip bulut teknolojileri ile çok sayıda kullanıcıya etkin ve hızlı hizmet verilebilmektedir. Bunun yanında istemcilere ait spesifik uygulama algoritmalarının düğümler üzerinde çalıştırılamaması, sunucu tarafta sunulan servis kısıtlarına bağlı kalınması, alt ağlarda filtreleme algoritmalarının çoğu zaman işletilememesi gibi önemli dezavantajları da bulunmaktadır [15].

İkinci problemle alakalı olarak literatürde genellikle yoğun trafikli WSN sistemlerde meydana gelebilecek büyük veri işlemlerinin kotarıldığı sistem modelleri bulunmaktadır. Literatür çalışmalarından bilindiği kadarıyla, farklı WSN sanallaştırma teknikleri kullanan, eklenen verilerin farklı istemci veya uygulamaları tarafından kullanılabilecek şekilde anlamlandırılabildiği iyileştirilmiş bir büyük veri işleyebilen bir sistem modeli bulunmamaktadır. WSN sanallaştırma tekniği olarak genellikle ATBS tekniklerinin kullanılması sistemdeki büyük verilerin farklı disiplinlerdeki istemcilerin veri anlamlandırmalarını kısıtlamaktadır.

Detayları sonraki bölümde verilecek olan literatür araştırmaları sonucunda, WSN sanallaştırmada ve büyük veri işleyebilen IoT/CPS sistemlerde aşağıdaki eksikliklerin olduğu görülmüştür;

- Bir istemci veya istemci WSN ağına ait yönetimsel komutların/sorguların farklı WSN ağlarda genellikle tanımsız olması.
- Donanım, katman ve gömülü işletim sistemi (EOS) farklılıkları, veri iletim hızının düşüklüğü gibi nedenlerden dolayı farklı yerlerdeki farklı teknolojili WSN ağlar arasında kod taşınabilirliğinin neredeyse imkansız olması.

- Kod taşınabilirliği probleminden dolayı bir istemciye ait algoritmaların veya özel filtreleme kurallarının farklı ağlarda çalıştırılamaması
- ABS ve DBS 'nın, çok sayıda istemcinin kullandığı ve büyük verinin gerçekleşebileceği IoT/CPS sistemlerde ve eski tip veya çok kısıtlı WSN düğümler (standalone node) ile kullanılamaması.
- ATBS 'de bulunan sunucu bazlı kısıtların istemci manevralarını azaltması.
- ATBS kullanan sistemlerde kaynak seçme/atama prosedürlerinde hem istemci hem de sağlayıcı alt WSN sistem parametrelerinin aynı anda dikkate alınmaması.
- Alternatif WSN sanallaştırma tekniklerinin kullanıldığı ve büyük verilerin oluşabildiği sistem modellerinin eksikliği.
- Ham veya istemci analizi yapılmış verilerin büyük veri analizlerinde başka kullanıcılar tarafından belli disiplinlere yönelik anlamlandırılması ve paylaştırılması.

1.3. Önerilen Sistemin Katkıları

Bu tezde, üst bölümlerde belirtilen eksikliklere yönelik olarak, günümüz yazılım ve sunucu teknolojilerinin sunduğu avantajlardan faydalanılarak, istemciye özellikle kod/algoritma taşınması açısından avantaj sağlayabilecek bir WSN sanallaştırma platformu önerilmektedir. Özellikle düğüm bazlı sanallaştırmanın kısıtlarını önemli ölçüde kaldıran bu model, yazılımsal bir çerçeve (framework) modeli kullanmaktadır. Bu yazılım modeli ile sağlayıcı/sunucu-istemci alt yapıları birbirinden soyutlanabilir ve sunucu tarafta istemciye özel bir çalışma sağlanabilir. Bu framework istemcinin kendi ağında kullandığı veya istediği uygulama katmanı fonksiyonları ile sunucu tarafta sağlanan sensör/aktuatör kaynaklara kendi belirlediği zamanlarda etkileşebilmesine imkan sağlar. Bunu geliştirilen yazılım çerçevesi yardımıyla oluşturulan ve "sanal düğüm (virtual node)-SD/VNd" olarak adlandırılan yazılımsal komponentler vasıtası ile gerçekleştirmektedir. Önerilen IoT/CPS WSN sağlayıcı mimari ile sisteme yeni dahil olan VNd 'lerin sistem çalışmasına engel olmaksızın çalışır hale getirilmesi, önerilen sistemin bir diğer özgün özelliğidir. Bununla birlikte bu tezde IoT/CPS sistemler için alternatif bir kaynak seçme/atama modeli de önerilmiştir. Genellikle istemci talep ve kaynak tipi parametrelerini baz alan klasik seçme/atama modeli yerine, bu model yardımıyla, hem istemci hem de sağlayıcı alt WSN parametrelerini de dikkate alan dinamik bir seçme/atama işlemi yapılabilmektedir. Büyük

veri işleme modülü ile ilgili olarak, sistem, VNd operasyonları sonucu oluşturulacak sonuçların anlamlandırılarak bu modülde farklı istemciler tarafından kullanılmasına/paylaşılmasına olanak sağlayabilmektedir.

Özet olarak bu tezde sunulan WSN sanallaştırma platformunda kullanılan yöntemlerle, IoT/CPS WSN sağlayıcılar için uygulanabilir önemli özellikler aşağıda sıralanmıştır.

- Belirli amaçlara yönelik tasarlanmış algoritmaların, sunucu tarafta düğüm bazlı sanallaştırmaya ihtiyaç kalmadan çalıştırılması
- Farklı uygulama katmanları çalıştıran heterojen WSN 'ler arasında komut/sorgu geçişkenliğinin sağlanarak, insan faktörü olmaksızın, WSN ağlara ait broadcast alanlarının genişletilmesi
- Filtreleme ve analiz işlemlerinin sunucu tarafta gerçekleştirilerek sunucu kullanıcı arası trafiğin minimize edilmesi
- Sunucu tarafın zaman ve şart kısıtlarının kullanıcı uygulamaları için büyük ölçüde kaldırılması
- Alt WSN sağlayıcı ve istemci parametrelerinin aynı anda dikkate alınarak kaynak seçim/atama işlemlerinin gerçekleştirilmesi
- Oluşacak büyük verilerin sunucu sistem tarafında yönetilmesi, gereksizlik analizlerinin yapılması ve istemcilerle paylaşılması

Tezde önerilen WSN sanallaştırma tekniğini içeren büyük veri teknoloji alt yapılı bir IoT/CPS WSN sağlayıcı modeline literatürde rastlanmamıştır. Dolayısıyla bu tezin, bu konudaki çalışmalara önemli bir katkıda bulunacağı da rahatlıkla ifade edilebilir.

1.4. Tezin Yapısı

Tez, yapısal olarak şu şekilde hazırlanmıştır; Bölüm 1, giriş, problem tanımı ve önerilen modelin literatüre katkısını içermektedir. Bölüm 2, IoT/CPS teknolojileri, büyük veri teknolojileri ve arakatman (middleware) teknolojileri ile ilgili temel bilgiler ele alınacaktır. Bölüm 3 ve 4'de, önerilen sistem mimarisi ve matematiksel model detayları açıklanacaktır. Bölüm 5'de önerilen mimariye göre oluşturulan IoT/CPS sağlayıcı simülasyonu ve önemli sistem parametrelerin optimum değerlerinin belirlendiği çözüm yöntemi detayları verilirken Bölüm 6 'de sisteme ait fiziksel gerçekleştirme detayları verilecektir. Bölüm 7 'da ise sonuçlar ve gelecek çalışmalar sunulmuştur.

2. ÖNERİLEN SİSTEMİN TEMEL KAVRAMLARI

Bu bölüm, tez içeriğinde bahsedilen problemlerin ve sunulan çözüm önerilerinin daha iyi anlaşılabilmesi için gerekli bazı konu detaylarını ve literatür araştırmalarını kapsamaktadır. Bu konulardan ilki geleceğin teknolojileri olarak adlandırılan IoT/CPS ve bu sistemlerin uygulama alanlarıdır. Genellikle birbirine karıştırılan bu teknolojilerin ortak ve ayrılan yanları IoT ve CPS sistemler başlığı altında açıklanacaktır. Diğer önemli konu WSN sistemlerde heterojenlik kavramıdır. Bir IoT/CPS WSN sağlayıcı sisteminin sahip olması gereken en önemli özelliklerden birisi arka plandaki karmaşıklığın istemcilerden soyutlanmasıdır. Bu karmaşıklık, heterojen yapılarda daha yüksektir. Bu maksatla WSN sistemlerde heterojenlik önerilen yöntemlerin irdelenmesinde incelenecek konu başlıklarından biri olacaktır. Bir sonraki konu başlığı ise WSN 'lerde sanallaştırma teknikleridir. Bu konu IoT/CPS sistemlerinin yaygınlaşması ile ayrı bir önem kazanmıştır. WSN 'lerde sanallaştırma teknikleri, avantaj ve dezavantajları "WSN 'lerde Sanallaştırma" başlığı altında verilecektir. Son olarak IoT/CPS sistemler altında heterojen WSN 'ler ve büyük veri teknolojilerinin bir arada yer almasının sağladığı avantajlardan bahsedilecektir. Günümüz büyük veri çözümlerinin, büyük ölçekli IoT/CPS WSN sistemlere entegrasyonu ve gerekli şartlar son bölümde "Büyük Veri Teknolojileri ve IoT/CPS Sistemler" de verilecektir.

2.1. IoT ve CPS

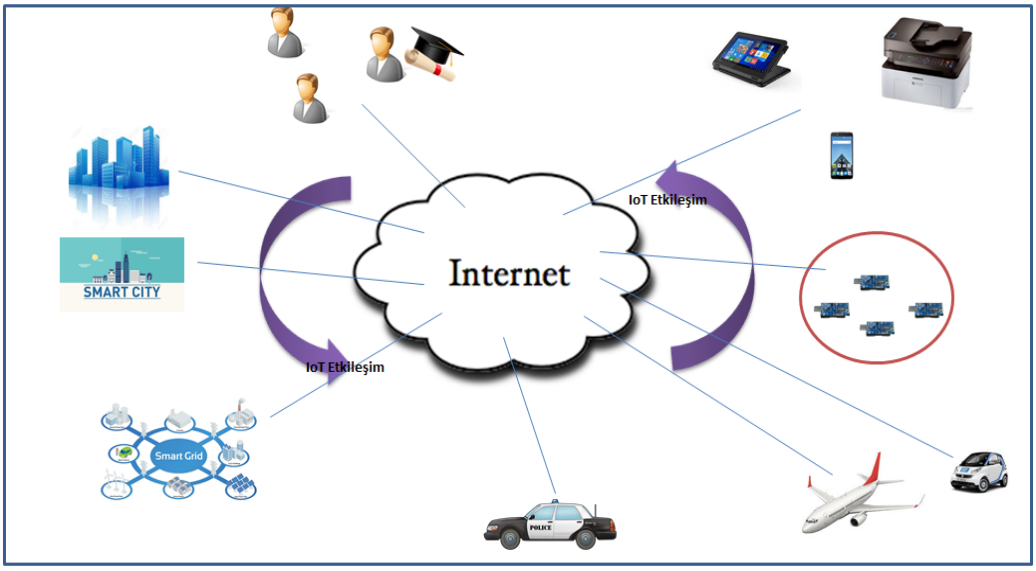
Bu bölümde IoT, CPS 'in genel özellikleri, birbirleri ile ayrılan yönleri ve klasik kontrol sistemlerinden farkları irdelenecektir.

2.1.1. Nesnelerin İnterneti

Şekil 2.1 'de betimlendiği gibi IoT, günlük yaşamda kullanılan aygıtların, gerçek ve tüzel kişilerin internet üzerinden birbirleri ile entegre olabilmelerini, bir ağ oluşturarak belli amaçlara yönelik çözümler üretebilmelerini ve gelecek teknolojileri için yeni bir alt yapı oluşturabilmelerini sağlayan teknolojiler bütünüdür denilebilir. Aslında IoT, spesifik tanımlamalarla net olarak ifade edilemeyecek nitelikte bütünsel bir kavramdır. İnternet teknolojilerindeki hızlı değişim, alt yapı hizmetlerindeki gelişim ve ucuzlama diğer

alanlarındaki farklı sistemlerin bu yönde everilmesine yol açtı. Gelecekte, özellikle endüstriyel ve ticari alanda IoT 'nin pek çok nimetlerinden faydalanılacaktır.

Akıllı cihazlar, RFID, NFC (Yakın Alan İletişimleri), UID (Universal/Unique Identifier), WISP (Wireless Identification and Sensing Platforms), WSN gibi teknolojiler, IoT sistemlerin nesneye yönelik ayaklarını oluşturmaktadır [3,34-47]. Böyle büyük bir yapı içerisinde, nesnelerin sadece sahip oldukları bağlantı yetenekleri veya benzersiz ID 'leri ile yönetilmesi mümkün değildir. Bu nedenle IoT, IPSO (IP for Smart Objects) [38], Web of Things [39], IoT ara katman yazılımları [40], M2M (makine-makine iletişim) ara katman yazılımları ve semantik teknolojileri gibi bileşenleri de kapsamaktadır [41]. Bununla birlikte her tür nesnenin aynı donanımsal ve yazılımsal kabiliyette olamaması sorunu, 6LoPAN, RPL, CoAP ve EXI gibi kısıtlı özellikli protokolleri sayesinde önemli ölçüde aşıldı [42]. Bu teknolojilerin HTML/XML, HTTP, IpV6 gibi geniş özellikli teknolojilerle çalışabilmeleri daha karmaşık heterojen yapıların IoT altında oluşmasına sebep oldu. Bu heterojen yapılar arasında platform bağımsız haberleşmenin yapılmasında, IETF, ETSI, W3C, SENSEI gibi oluşumların desteği ile internet dünyasında önemli bir yere sahip olan web servisleri etkin bir rol oynamaya devam edecektir. SOAP [46] ve ReSTFul [47] temelli bu teknolojiler aynı zamanda M2M ara katman yazılımı olarak da değerlendirilirler. Günümüz sosyal ağ teknolojilerinin de IoT teknolojilerine özellikle sosyal yönelimli analiz projelerinde farklı bir destek sağlayacağı da aşikârdır.



Şekil 2.1. Genel IoT etkileşim

Literatürde IoT ‘ye yönelik oldukça fazla çalışma bulunmaktadır. Genel bir kavram olmasından dolayı bir çok farklı alanda uygulama alanı bulmuştur. Lojistik alanda yapılan çalışmalar, çevresel gözlemleme çalışmaları, sağlık alanındaki uygulamalar bunlardan sadece bazılarıdır [48-50]. En temel problemleri arasında güvenlik ve standartlaşma sayılabilir. Özellikle akıllı şehirler, akıllı enerji sistemleri gibi büyük sistemlerde, sağlıkla alakalı büyük projelerde bu sorunlar daha da önem kazanmaktadır.

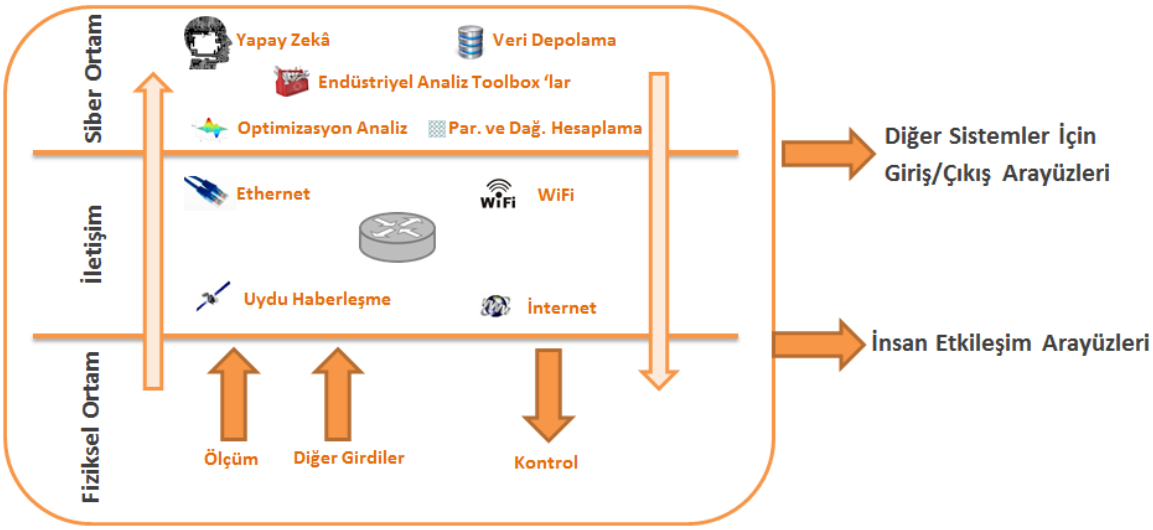
2.1.2. Siber Fiziksel Sistemler

Cyber Physical Systems (CPSs), en temel manada, hesaplama, ağ sistemleri ve fiziksel süreçlerin entegrasyonundan oluşan yapılara verilen genel bir isimdir. Diğer bir tanıma göre ise, CPS bünyesinde bir çok sensör / aktuatör bulunduran ve akıllı karar verme yetisine sahip bir sisteme entegre olmuş yapılardır . CPS ‘lerde, var olan bir fiziksel birim veya büyüklüğün kontrolü, spesifik bir ağ teknolojisi alt yapısı üzerinde çalışan hesaplama birimleri tarafından gerçekleştirilmektedir. 2006 yılında Amerika ‘nın Ulusal Bilim Fonu (The US Natinal Science Foundation–NSF) CPS’leri anahtar araştırma alanı olarak seçmiştir. Tarımdan askeri alanlara kadar düşük veya yüksek bütçeli bir çok CPS projesi teknoloji dünyasında yer almaya başlamıştır [51-54].

CPS’lerin en önemli özellikleri arasında hiç kuşkusuz disiplinler arası olması gelmektedir. CPS’lerde gömülü sistemler, ölçüm sistemleri, ağ teknolojisi üzerinden fiziksel süreci sürekli olarak izler ve gerekli kontrolleri yapar. CPS ‘lerin genel bileşenleri şunlardır;

- Fiziksel büyüklüklere ait dinamiklerinin ölçümü ve takibi
- Modelleme
- Farklı analiz teknikleri üzerinden tümevarım
- Yazılım ve ağ sistemlerinin entegrasyonu
- Kontrol

Sistemde genellikle bir geri besleme süreci, alt siber sistemler ve kablosuz teknolojiler vardır. Kablosuz sensör ağları (WSN) bu sistemlerde oldukça önemli yere sahiptirler. CPS sistemler ölçüm ve kontrol amaçlı olduklarından zaman kritik uygulamalarda sıklıkla tercih edilmektedir.



Şekil 2.2. Bir CPS 'in genel yapısı

Şekil 2.2 'de görüleceği üzere CPS'ler temelde iki faklı ortamdan (domain) oluşur. Bunlar "Fiziksel Ortam (Physical Domain)" ve "Siber Ortam (Cyber Domain)" dir. Bu ortamlar farklı iletişim ve ağ teknolojileri üzerinden birbirlerine bağlıdırlar. Her iki ortam da kendi içerisinde heterojen bir yapıya sahiptirler. Bu ortamlar birbirinden oldukça farklı disiplinler içerebilirler. Öreğin fiziksel ortamda rahatlıkla bir kimyasal ve mekaniksel yapı, birbirleri ile etkileşebilir veya çıktıları ortak bir hesaplama uzayına aktarılabilir. Bu iki ortam doğal olarak bir CPS sistemin karmaşıklığını arttırmaktadır.

Doğası gereği CPS, dağıtık sistemlerle içli dışlıdır. Bu nedenledir ki dağıtık sistemlerin yapısında olan problemler ve zorluklar kendisini CPS'lerde de göstermektedir. Ölçeklendirme ve şeffaflık CPS'lerin de üzerinde araştırmalar yapıldığı alanlardır [51-53].

Bunun yanı sıra günümüzde akıllı telefonların artması ile birlikte mobil CPS 'ler ayrı bir önem kazanmıştır. Taşınabilir akıllı cihazların sahip oldukları hafızalama, hesaplama, görüntüleme ve kayıt sistemleri, GPS teknolojileri, farklı sensör uygulamaları, üzerlerinde farklı ağ teknolojileri bulundurabilmeleri gibi özelliklere sahip olması sayısız CPS uygulama alanına kapı aralamaktadır.

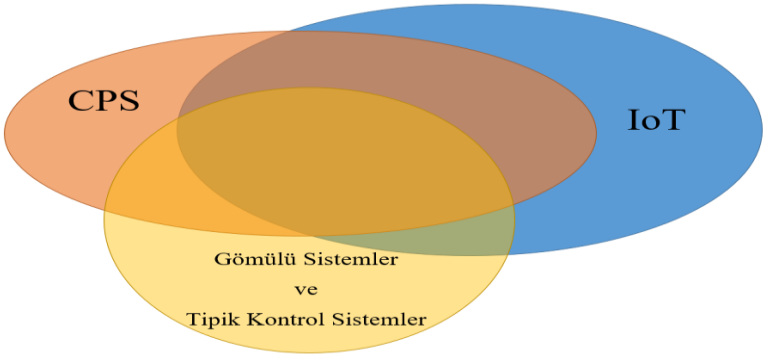
Siber ortamda yapılan çalışmalar genellikle farklı disiplinlere ait önceden denenmiş/ispatlanmış bilgi ve karar havuzlarının ortak bir hesaplama uzayında akıllı hesaplama teknikleri ile işlenmesi üzerinedir. Toplanan verilerin genellikle dağıtık ve heterojen olması siber ortamdaki yükü ve karmaşıklığı arttırmaktadır. CPS temelli çalışmalar günümüzde birçok büyük projenin çekirdeğini oluşturmaktadır. Kullanılan sistemlerin

hassasiyeti ve senkronizasyonu son derece hayati önem arz etmektedir. Siber ortamın iyi tasarlanması ve sürdürülebilir olması CPS 'lerin olmazsa olmazları arasındadır.

2.1.3. IoT, CPS ve Tipik Kontrol Sistemleri Arasındaki Farklar

IoT ile CPS 'ler arasındaki farklar nelerdir sorusu genellikle IoT ile CPS veya CPS ile tipik kontrol sistemleri için sorulmaktadır. Genel bir kanı olarak IoT sistemlerin CPS 'leri kapsadığı ileri sürülse de bu tam manası ile doğru değildir. IoT sistemler alt yapı olarak internet ve internet teknolojilerini kullanan genel bir kavramdır. Öte yandan CPS 'ler alt yapı olarak internet ve internet teknolojilerini kullanmak zorunda değildir. CPS 'ler her hangi bir veya birden fazla ağ teknolojisini kullanabilirler. İki sistem arasında ki en önemli fark budur. Bunun yanında CPS sistemler daha çok mühendislik temelli endüstriyel uygulamalarda kendi göstermektedir. CPS sistemlerin boyutu IoT sistemlerin boyutlarına ulaşamayabilir. Ancak internet ağ alt yapısında interneti kullanması durumunda, bu sistemlerin IoT kapsamı içinde değerlendirilmesi doğaldır.

CPS 'ler, ağ teknolojileri sayesinde, heterojen, genişleyebilir ve daha güçlü hesaplama yeteneklerine sahip kontrol sistemlerdir. Öte yandan tipik kontrol sistemleri daha çok belli bir amaca yönelik belirli alıcı ve aktuatör bileşenlerden oluşur. CPS 'ler ölçeklenebilir teknolojilerdir ve otonom heterojen sistemleri bir ağ teknolojisi üzerinden etkileştirebilmeleridir. Bir diğer deyişle tipik kontrol sistemlerinin bilişim sistemlere evirilmiş şeklidir. Bunun yanında tipik kontrol sistemleri gerekli şartların sağlanması durumunda CPS 'lerle beraber çalışabilir duruma gelebilirler. Bunun olmasında hiç kuşkusuz gömülü sistem teknolojilerinin yadsınamaz bir katkısı bulunmaktadır.

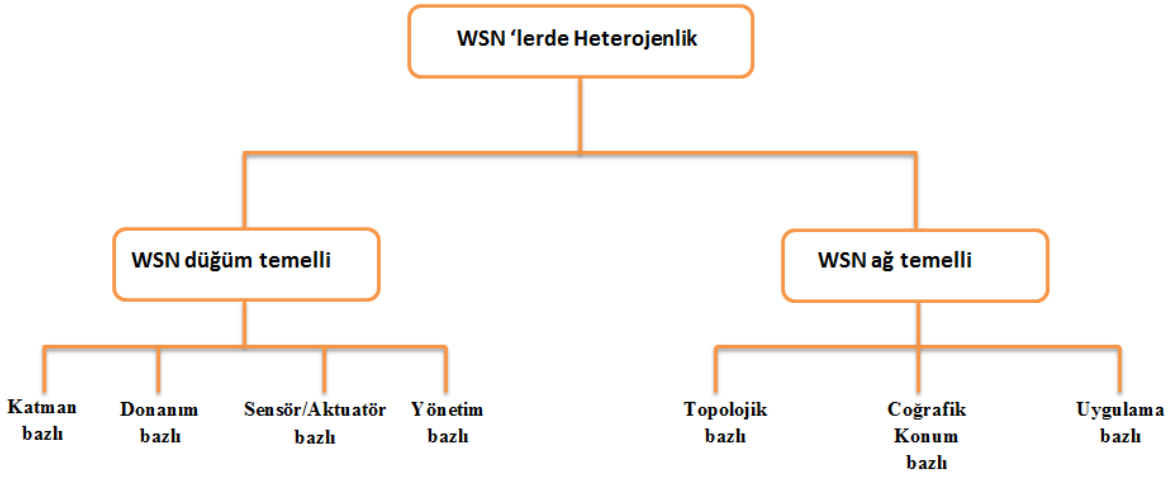


Şekil 2.3. IoT, CPS, gömülü sistemler ve tipik kontrol sistemleri arasındaki ilişki

Şekil 2.3, söz konusu bu teknolojilerin birbirleri ile olan ilişkilerini göstermektedir. Şekilden de anlaşılacağı üzere tüm sistemler farklı kombinasyonlar ile birbirlerinin tanımları arasında geçişken davranabilirler ve birbirlerini kapsayabilirler.

2.2. WSN'lerde Heterojenlik

WSN 'ler hem akademik hem de endüstriyel alanda yaklaşık 20 yıldır aktif bir çalışma alanıdır. WSN sistemlerde heterojenlik genelde WSN düğüm ve bu düğümde kullanılan teknolojiler ile alakalıdır. Bu nedenle bu tez daha genel bir yaklaşım ile WSN 'lerde heterojenliği iki ana başlık altında gruplandırmaktadır [129]. "WSN düğüm temelli heterojenlik" ve "WSN ağ temelli heterojenlik". Buna bağlı genel sınıflandırma yapısı Şekil 2.4 'de verilmiştir. WSN sistemler her iki heterojen yapıyı aynı anda kendi bünyelerinde barındırabileceği gibi ayrı ayrı da tesis edilebilirler.



Şekil 2.4. WSN 'lerde heterojenlik sınıflandırılması

2.2.1.1. WSN düğüm temelli heterojenlik

Düğüm temelli heterojenlik, katman bazlı, donanım bazlı, sensör/aktuatör bazlı, yönetim bazlı olmak üzere 4 alt başlık altında incelenebilir. Bunlardan ilki, katmansal düzeydeki heterojenliktir. WSN sistemler yapısal olarak klasik ağ teknolojilerindeki gibi katmansal bir mimari model ile açıklanırlar. Genel olarak Fiziksel, Ortama Erişim, Ağ, Taşıma ve Uygulama katmanlarından oluşan WSN yapıları, belli bir amaca yönelik WSN tasarımlarında, katmanların sayısı veya nitelikleri bakımından farklılıklar arz edebilir. Ancak temel olarak minimum iki katman her tasarımda bulunmaktadır. Bunlar fiziksel katman (PHY katman) ve ortama erişim katmanıdır (MAC katmanı). Daha sonra amaca yönelik olarak ağ (yönlendirme) katmanı, taşıma katmanı, uygulama katmanı ve bu katmanlarla beraber çalışabilen güvenlik ve mobilite gibi ek katmanlar da eklenebilir.

WSN sistemlerde PHY katmanı frekans seçimi, taşıyıcı sinyal üretimi, sinyal algılama ve modülasyon gibi temel elektriksel operasyonları yürütmektedir. Bu katman kablosuz medya olarak genellikle RF teknolojilerini kullanmaktadır. Neredeyse tüm WSN teknolojileri herhangi bir lisans gerektirmeyen ISM bantlarını tercih etmektedirler. En çok tercih edilen bantları 433, 868, 915 ve 2400 MHz RF bantlarıdır [1,32]. WSN'lerde kablosuz iletişim için genel olarak B-FSK, QPSK ve bunların gelişmiş versiyonları olan modülasyon teknikleri kullanılır. RF sinyal iletişimi için ise en çok tercih edilen teknik "Spread Spektrum- SS" teknolojisidir. Bu teknik RF iletişim için veri hızını ve sinyalin gürültüye direncini arttırmak için tercih edilir. Detayları bu tezin kapsamında olmayan bu tip teknolojiler WSN 'lerde PHY katman karakteristiğini doğrudan değiştirir. IoT/CPS ölçeğinde karşılaşılabilecek katmansal farklılıklarla ilgili detay bilgi [32] 'de sunulmuştur.

Düğüm bazlı heterojenlikte bir diğer önemli farklılık donanımsaldır. WSN düğüm teknolojileri donanımsal bakımdan 3 farklı kategoride incelenebilir, "hesapsal farklılık", "kapsama temelli farklılık" ve "enerji kaynağı temelli farklılık" [55]. WSN düğümler farklı hesapsal kabiliyetlere sahip olabilirler. Örneğin multi-media uygulamalar için kullanılan düğüm teknolojileri (Imote2, Stargate, MeshEye vb.), 40-400 MHz arası işlemci ve 32-64 MB hafıza boyutu gibi değerlere sahip olabilirken, sıcaklık, basınç ölçümü gibi daha az karmaşık senaryolar için kullanılabilen düğüm teknolojileri (Telos, MicaZ vb.), 8-16 MHz işlemci gücü ile 8-128 KB 'lık hafıza boyutlarında olabilmektedir [56-58]. Kapsama temelli farklılıklar düğümün radyo ünitesi teknolojisi ile alakalıdır. Radyo ünitesinin tipi (WiFi, bluetooth, 802.15.4 PHY vb.), transmisyon ve alım gücü, alıcı hassasiyeti gibi özellikleri ağdaki düğüm yerleşimini, enerji tüketimi, MAC ve yönlendirme katmanlarındaki işleyişi doğrudan

etkilemektedir. Enerji kaynağı temelli farklılıklar daha çok düğümün kısıtlı enerji kaynağı kullanıp kullanmaması ile alakalıdır. Bazı hayati düğümler daha güçlü donanımsal özelliklere sahip olabilir ve 802.11 gibi daha fazla enerji tüketimi gerektiren geniş bant RF teknolojilerini kullanabilirler. Bu tip düğümler genellikle WSN ağ içinde ek özel görevlere sahiptirler (ZigBee koordinatör ve yönlendirici düğümler vb.). Bunun dışında genellikle WSN düğümler batarya kaynaklarına sahiptirler ve uyu-uyan-işlem yap-uyu mantığı ile çalışırlar. Bunun dışında solar enerji gibi faklı enerji kaynaklarını kullanan düğüm teknolojileri de mevcuttur. Donanım bazlı farklılıklar, WSN sanallaştırmada en önemli karmaşıklıklardan biridir. Kaynak seçimi/atama prosedürleri, ne tür bir çalışma modeli kullanılacağı gibi meseleler bu farklılıklarla yakından ilişiklidir. WSN 'lerde kullanılan sanallaştırma teknikleri donanımsal farklılıklar açısından farklılık gösterebilmektedir. Özellikle yeterli hafıza ve işlem gücüne sahip olmayan düğümlerde kaynak paylaşımı için aracı donanımlara/servislere ihtiyaç duyulabilmektedir.

Yine benzer şekilde düğümün sahip olduğu sensör/aktuatör tipide heterojenliği etkilemektedir. WSN düğüm teknolojileri genellikle farklı tip sensör/ aktuatör kaynaklar ile çalışabilmeleri için tasarlanırlar. Bu nedenle birçok WSN düğüm teknolojisi on-board sensör/aktuatör kaynaklarının yanında farklı tip sensör/aktuatör kaynaklar için genişleme yuvalarına sahiptir. Bir sensör düğümün sahip olduğu kaynaklar zaman içinde değiştirilebilir. Bu değişim, düğümün işlem yükünde ve enerji tüketiminde de önemli değişimlere sebep olabilir. Aynı fiziksel fenomen için geliştirilmiş sensör/aktuatör kaynaklar farklı spesifikasyonlara ve erişim şekillerine sahip olabilmektedir. Özellikle aktuatör içeren WSN ağlardaki kaynak paylaşımı güvenlik ve çoklu erişim kısıtları gibi etkenlerden dolayı sadece sensör içeren ağlardaki kaynak paylaşımından farklı olacaktır.

Düğüm üzerindeki yönetimsel şekil farklı uygulamaların aynı düğüm üzerinde kaynak paylaşımı yapabilmesini doğrudan etkiler. Her ne kadar gömülü bir işletim sistemi (GİS/EOS) yerine amaca yönelik geliştirilmiş küçük yönetimsel yazılımları kullanan WSN düğümler bulunsa da gelişmiş WSN sistemler genellikle düğümlerinde gömülü işletim sistemlerini kullanmayı tercih ederler. Kendi yönetimsel yazılımları kullanan WSN düğümler genelde "modül" olarak adlandırılırlar. Bu modüller kendi başlarında çalışabilme yeteneğine sahip (standalone) düşük konfigürasyonlu ve karmaşık operasyonlarda kullanılmayan elemanlardır (Digi XBee vb.). Farklı MCU 'lar ile de çalışabilen bu modüllerin en önemli avantajları hızlı kurulum ve maliyettir. Hem periyodik çalışma modunda hem de ön tanımlı komut seti sayesinde, sorgu-cevap (query-answer) modunda işlem yapabilirler. Ayrıca bu modüller bir MCU ile beraber kullanılmadıklarında amaca yönelik algoritmaları çalıştıramazlar.

Öte yandan günümüzde daha gelişmiş düğüm teknolojilerine yüklenebilen TinyOS, Contiki, RIOT gibi farklı EOS ‘lar, WSN sistem tasarımcılarının kendi algoritmalarını kullanarak düğümler üzerinde programlama yapmalarına ve düğüm kaynaklarını yönetmelerine imkan vermektedir [59-61]. Gerçek zamanlı uygulamalar veya kompleks algoritmaların kullanılacağı senaryolar kullanılacak EOS ‘u belirlemede bir kriter olabilmektedir. Aynı tip düğümlerin farklı tip EOS ‘lar ile yönetilmesi (örneğin farklı Telos düğümlerin Contiki ve TinyOS ‘u kullanması gibi) bir ağ içinde tam bir heterojen yapı oluşturmayabilir ancak bu durum WSN sanallaştırma uygulamalarının kullanıldığı bir ortamda kod taşınabilirliği açısından bir engel oluşturacaktır.

2.2.1.2. WSN ağ temelli heterojenlik

Diğer bir heterojenlik sınıfı WSN ağın özelliklerine göre belirlenmektedir. Bu durumda ağ topolojisi, coğrafi lokasyonu ve uygulama tipi belirleyici faktörlerdir. Topolojik açıdan WSN sistemler hiyerarşik ve mesh tipi bir topoloji kullanabilmektedir [1,32]. Bunlar bir küme (cluster) içerisinde tanımlı olabileceği gibi farklı özellikteki dağıtık bir modeli de kullanabilirler. Kümesel yapılandırma ön tanımlı veya dinamik olabilir. Biraz daha geniş ölçekli bir kısım WSN sistemler bir birinden farklı topolojileri aynı anda ihtiva edebilirler. Kullanılan topoloji, düğüm adresleme ve yönlendirme katmanı çalışma prensiplerini doğrudan etkileyen faktörlerdir.

Öte yandan, WSN sistemler coğrafi olarak farklı lokasyonlarda bulunabilir. Bu özellikle zaman senkronizasyonu ve gecikme gibi iki önemli kavram açısından oldukça önemlidir. Farklı konumlardaki WSN sistemlerin etkileşmesinde, düğümlere/kaynaklara ait lokasyon bilgileri önemli bir parametredir. WSN sanallaştırma zamanlamanın önemli olduğu durumlarda bu parametreler mutlaka dikkate alınmalıdır.

Farklı disiplinler veya ticari uygulamalar çalıştıkları senaryo gereği farklı amaçlar güdebilirler ve aynı çatı altında bulunabilirler. Bu durum IoT/CPS sistemler için her zaman olası bir durumdur. Farklı uygulama katmanlarına sahip sistemlerin kendi iç işleyişlerinde kullandıkları komut veya sorgular bir diğeri için genellikle tanımsızdır. Dahası bir WSN sistemde kullanılan algoritma/algoritmalar bir diğer uygulamada yürütülemeyebilir. Bu durumda beraber çalışabilirlik şartlarının minimum ortak paydada sağlanması gerekmektedir. Özellikle bu farklılık, IoT/CPS sitemlerde insan müdahalesiz beraber çalışabilirlik için soyutlanması gereken ana faktörlerden birini teşkil eder.

2.3 WSN'lerde Sanallaştırma

WSN sanallaştırma, WSN 'lerin doğası gereği sahip oldukları temel kısıtları da dikkate alarak bir veya birden fazla farklı WSN sistemdeki kaynaklar üzerinde, farklı amaçlara hizmet eden çoklu uygulamaların belirli teknikler yardımıyla sorunsuzca yürütülebilmesine veya kaynak paylaşımı yapılabilmesine imkân verilmesidir. WSN 'lerde sanallaştırma kavramını diğer teknolojilerdeki sanallaştırma kavramlarından ayıran temel özellikler;

- Kısıtlı donanımsal ve yazılımsal yetiler
- Farklı çalışma modelleri (farklı uyuma periyotları vb.)
- Dağıtık yapıları
- Heterojen içeriğe sahip olabilmeleri

olarak ifade edilebilir.

Bilindiği kadarıyla literatürde şu ana kadar yapılan WSN sanallaştırma tekniklerinin çok büyük bir bölümü homojen sistemler üzerine yapılmıştır. Bu çalışmalar genellikle belirli EOS 'lar ve protokol yapıları kullanılarak gerçekleştirilmiştir. Ancak IoT/CPS yapılar altında farklı teknolojili WSN sistemler bulunabilir ve bu sistemler birbirlerinin kaynaklarını kullanmak isteyebilir. Bu nedenle WSN 'lerde sanallaştırma bu alanda biraz daha detaylanmaktadır. Bu bağlamda son zamanlarda ortaya çıkan WSN bulut teknolojileri, kendi alt yapılarında sağladıkları çözümlerle akıllı şehir, akıllı enerji, akıllı tarım sistemleri gibi projelerin gerçekleşmesinde büyük pay sahibi oldular [62]. Ancak bu sistemler genellikle belli bir alana yönelik kapalı sistemler olup, harici istemcilerin taleplerini ya kısıtlı bir şekilde karşılamakta ya da hiçbir şekilde karşılamamaktadır. Öte yandan kamusal ve ticari alandaki WSN bulut sistemleri genellikle sensör veri paylaşımına dayalı olup harici WSN kaynakların doğrudan etkileşimine izin vermemektedir. Bu özellikle bir insan faktörü olmadan farklı ağlar arası dinamik bir WSN düğüm – WSN düğüm etkileşimini imkânsızlaştırmaktadır.

IoT/CPS 'ler altında bir WSN sistem başka bir WSN sistemin/sistemlerin sadece belli kaynaklarına ihtiyaç duyabilir veya sadece belli düğümleri ile entegre olmak isteyebilir. Bu entegrasyon iki şekilde gerçekleşebilir. Birincisi bir sistem diğer sistemden sadece ölçüm verilerini spesifik zamanlarda talep eder ve bu talep kaynak tarafından genellikle publish/subscribe teknolojiler gibi MW aracılığı ile sunulur. Biz bu tip entegrasyonları basit entegrasyon (BE) olarak adlandıracağız. Kompleks entegrasyon (KE) olarak adlandırdığımız ikinci entegrasyon tipi ise bir WSN sisteminin tüm uygulamasının (komut setleri, sorgu formatları vb.) bir başka sistem üzerinde çalıştırılmasına izin verilmesi ile gerçekleşir. BE sistemler basit ve genel bir amaca yönelik olduğundan bu sistemlerin günümüzde pek çok

başarılı uygulamasını bulmak mümkündür [65-67]. BE sistemler sunucu tarafın sunduğu hizmetler ile sınırlıdır. KE sistemde ise istemci kendi çalışma prensiplerini sunucu tarafa iletip genel bir anlaşma sağlandıktan sonra daha esnek bir beraber çalışma ortamı sağlanır. KE yapılarda bir WSN 'nin broadcast alanı diğer ağların içlerine genişletilebilir. He ne kadar bu tezde önerilen mimaride BE yapı destekleniyor olsa da asıl odak noktası KE sistem yapısıdır. KE sistemler daha çok profesyonel yapıda olup pahalı akıllı sistemlerde ön plana çıkar ve M2M sistemler için daha uygundur.

Genel detaylarına geçmeden önce, KE entegrasyonun genel kullanım amaçlarının ve hangi sorunlara çözüm bulunacağının daha net bir şekilde izahı için birkaç uygulama senaryosunu sunulacaktır;

Senaryo 1; Belirli bir lokasyonda kendi belirlediği zaman ve şartlarda atmosferik gözlem yapmak isteyen bir kuruluş ilgili alanda başka bir kuruluşa ait bir başka WSN sistemin olduğunu öğrenmiştir. Birinci kuruluş tamamen farklı WSN donanım ve yazılım teknolojileri kullanmakla birlikte ikinci kurumun istediği ölçümleri yapabildiği bilinmektedir. İkinci kuruluş kendi uygulama komut ve sorgularını, birinci kuruluşun WSN yapısındaki bazı düğümler üzerinde çalıştırmak isteyebilir. Bu durumda mevcut uygulamanın çalışmasına zarar vermeden, öneden hiç bilinmeyen bu uygulamanın sisteme sorunsuz entegre edilmesi problemi KE 'de kendine cevap arayacaktır.

Senaryo 2; Birbirinden farklı özel (private) uygulamlara sahip ZigBee ağlar, farklı Profile ID, Endpoint ve Cluster ID 'lere sahip olabilir. Bu ZigBee ağlar kısmi bir oranda veya bütün olarak broadcast alanlarını diğer ağlara genişletmek isteyebilir. Kendi Cluster Library 'sini diğer ağlarda kendi isteği doğrultusunda kullanmak isteyebilir. Her ne kadar bir ZigBee düğüm üzerinde birden fazla profile çalıştırabilse de uzak lokasyonlarda, mevcut yapıyı kesintiye uğratmadan, başka profillerin ilgili düğümlere yüklenmesi işi aşılması zor bir problemdir. Bu problemde KE sistemlerin kapsamı alanına girer.

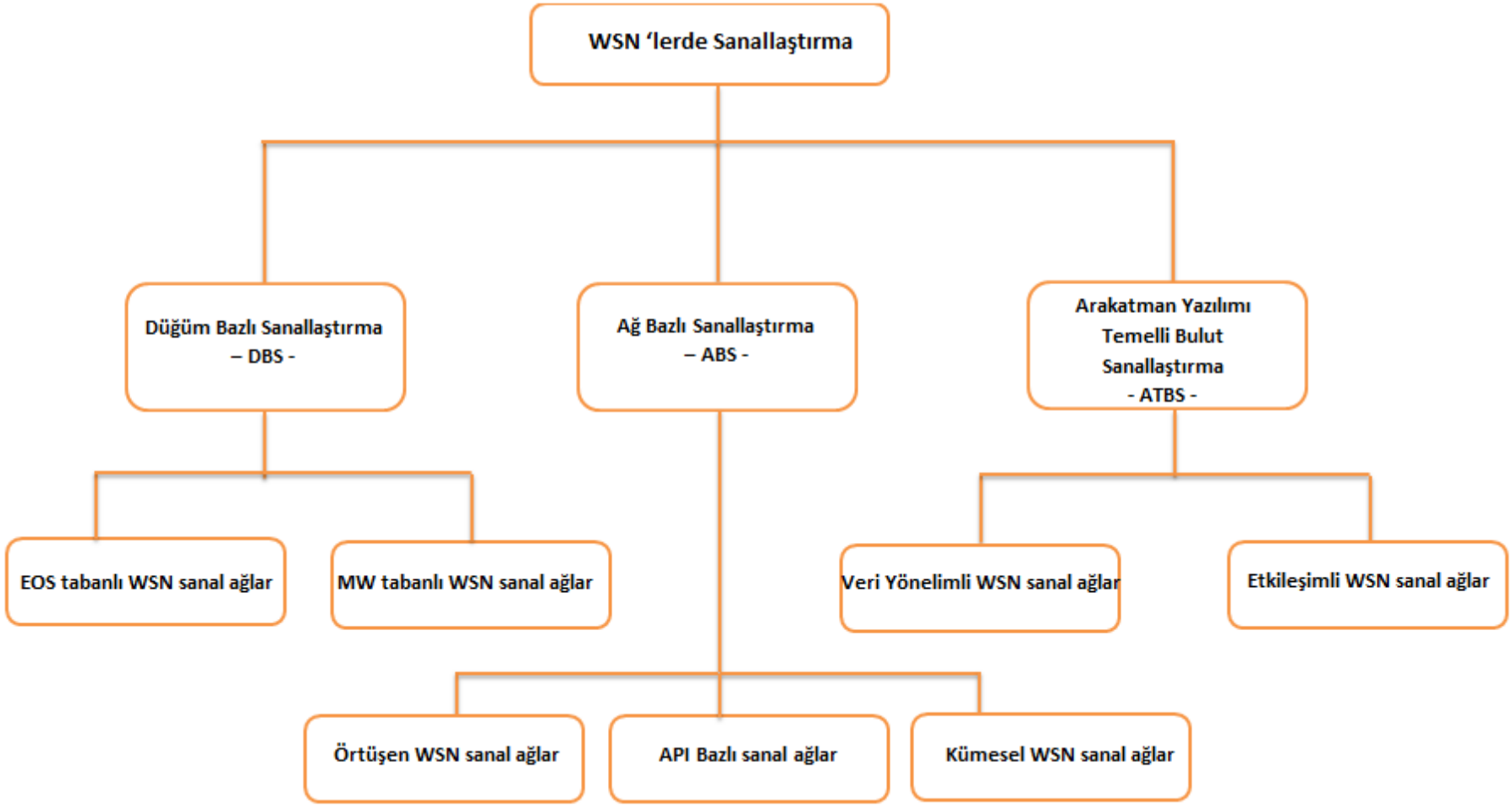
Senaryo 3; Akıllı şehir (Smart City) projesi üzerine çalışan bir belediye, şehrin belirli bölgelerindeki mevcut farklı WSN 'leri kendi uygulama broadcast 'i altında toplamak isteyebilir. Bu durumda diğer amaçlara yönelik kurulmuş olan bu sistemlerin doğal işleyişini bozmadan, en ekonomik şekilde kaynak paylaşımı istemesi olası bir durumdur. Bu proje altında çalışan farklı WSN 'lere ait düğümler, kendi uygulama komut/sorgu setlerini diğer WSN üzerinde kaynak ve hizmet keşfi yapmada kullanmak isteyebilir. Bu yazılımsal tasarım problemleri yine KE sistemlerin sunduğu çözümlere ihtiyaç duyar.

Senaryo 4; Birbirinde bulunmayan farklı sensörlere/aktuatörlere sahip deney platformları içeren laboratuvarlar (Kimya, biyoloji vb.) ortak bir deney platformunu mevcut WSN 'leri

üzerinden kurabilirler. Zaman ve deney şartlarının sağlanması durumunda bu M2M platformun, her iki tarafta belirli aktuatörleri devreye alıp çıkarması gerekebilir. Bu durumda platform iki farklı WSN teknolojisinin kullandığı kaynak tanımlamalarını ve çalışma parametrelerini çözümleyebilmelidir.

Çoğaltılması mümkün olan bu senaryolarda ortaya çıkacak temel problemlerin çözümünde sanallaştırma etkin bir rol oynamaktadır. Kaynak paylaşımı özellikle IoT sistemlerin önemli özelliklerinden birisidir. Bu hem maliyet hem de zaman tasarrufu açısından oldukça önemlidir. Doğal olarak heterojen sistemlere ait farklı kaynakların çoklu uygulamalar tarafından kullanılması belli kurallar çerçevesinde olmak zorundadır. Bu kurallar WSN sanallaştırmada daha katıdır. Uygulanacak sanallaştırma yöntemleri kullanılacak sisteme ve heterojen yapıya göre farklılık arz edebilir. Donanımsal bazda homojen, yönetimsel bazda heterojen olan bir sistemde uygulanan sanallaştırma tekniği ile konumsal bazda heterojen olan sistemlerde uygulanacak sanallaştırma teknikleri birbirinden farklı olabilir. IoT/CPS sistemler hemen hemen tüm heterojen sınıflandırmaları içerebileceğinden bu durum farklı problemlerin aynı anda ortaya çıkmasına neden olabilir. Amaca yönelik olarak hem BE hem de KE yapılar sanallaştırma teknikleri ile bu problemlere cevaplar sunmaktadırlar.

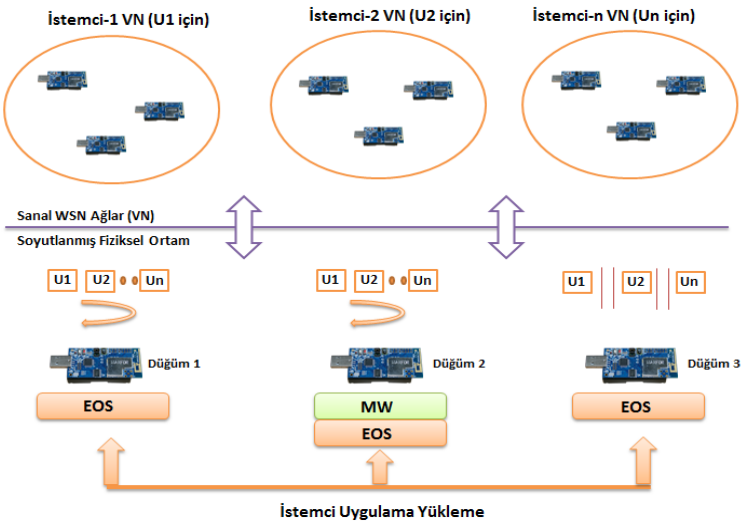
Bu tez, Şekil 2.5'te de görüleceği üzere, WSN 'lerde sanallaştırma tekniklerini üç başlık altında sınıflandırmaktadır. Bunlar, "Düğüm Bazlı Sanallaştırma (DBS)", "Ağ Bazlı Sanallaştırma (ABS)" ve "Arakatman Yazılımı Temelli Bulut Sanallaştırma (ATBS)" dır. WSN 'lerde sanallaştırma literatürde genellikle DBS ve ABS olarak iki şekilde sınıflandırılmaktadır [15,62]. Bu sınıflandırmalar, yapılan çalışmaların sadece WSN araştırma konuları içerisinde kalmasından dolayı yapılmıştır. Bu tezde kabul edilen sınıflandırmada DBS ve ABS için [15] ve [62] 'ındaki sınıflandırma esas alınmıştır. Günümüz IoT/CPS sistemlerde kullanılan teknolojileri de dikkate alındığında bu sınıflandırma yetersiz kalmaktadır.



Şekil 2.5. WSN 'lerde sanallaştırma sınıfları

2.3.1. Düğüm Bazlı Sanallaştırma

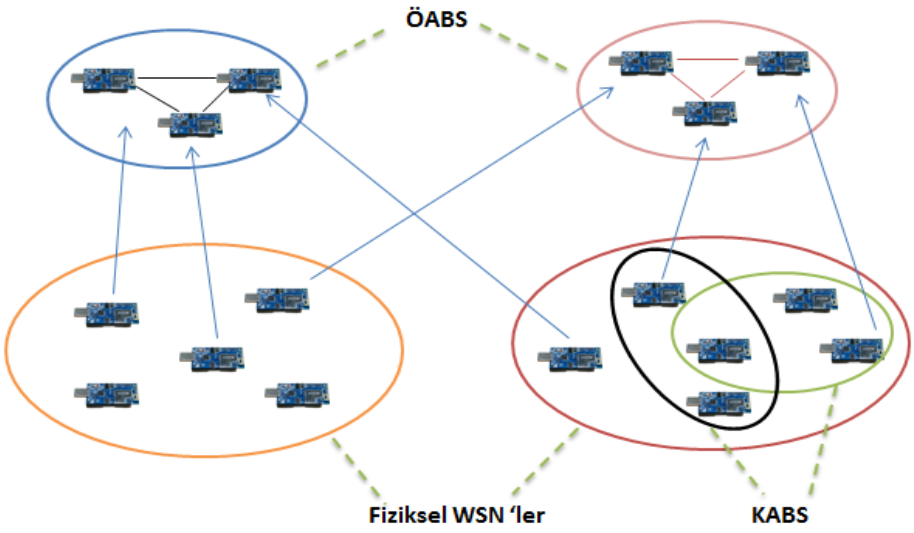
Düğüm bazlı sanallaşitırma (DBS) tek bir düğüm üzerinde farklı uygulamaların çalıştırılması amacını güder. Bu sanallaştırma tekniği tamamıyla düğüm özelliklerine bağlıdır. Alt grup olarak iki başlıkta incelenebilir [15]. Bunlar "EOS tabanlı" ve "Küçük boyutlu ara katman yazılımları (MW) tabanlı" sanallaştırmadır. Her iki yöntemde de işletim sistemi özellikleri ilk öne çıkan özelliktir. Bundan sonraki kriterler düğümün donanımsal ve hesapsal yetenekleridir. Her iki yöntemde de hem olay tabanlı hem de çoklu iş parçacığı (multi-thread) program modellerinde uygulamaları mevcuttur. Olay tabanlı programlamada sonlu durum makineleri destekli kodlama teknikleri kullanılırken, çoklu iş parçacığı temelli programlamada eş zamanlı yürütüm gerçekleştirilmektedir. DBS 'de dikkat edilmesi gereken noktalardan biri düğüm teknolojilerinde genellikle hafıza koruma olmadığından ortaya çıkacak hafıza yönetiminin çok iyi yapılması gerekliliğidir. Bu nedenle her bir uygulamanın çalışma anında kullanacağı yığın (stack) bölgesi, çalışma zaman dilimi, DBS için gerçekleştirilecek programlamada hayati önem taşır. Düğüm üzerinde çalışan EOS, uygulamalar için mantıksal adreslemeler kullanarak bunları daha sonra fiziksel adresler üzerine uygun bir şekilde haritalar. Şekil 2.6 genel olarak DBS 'nin çalışma yapısını göstermektedir. Şekilde DBS 'de kullanılan ardışık (Düğüm1 ve 2) ve çoklu iş parçacığı bazlı (Düğüm 3), MW bazlı (Düğüm 2) tekniklerin temel işleyişleri bir arada gösterilmiştir.



Şekil 2.6. DBS genel işleyiş prensipleri

2.3.2. Ağ Bazlı Sanallaştırma

Ağ bazlı sanallaştırmada (ABS) ise amaç, WSN ağlardaki belirli düğümlerden oluşan alt ağlar oluşturarak sanal ağlar kurmaktır. Bu tip alt sanal ağlar tek bir WSN içerisinde olabileceği gibi farklı ağlardan seçilen düğümlerden de oluşabilmektedir. Her ne kadar literatürde ABS, örtüşen-ÖABS [68] ve kümesel -KABS [69] WSN sanal ağlar olmak üzere iki alt başlıkta incelense de sunulan API 'ler vasıtasıyla özel bir iletişim protokolüne ihtiyaç duymayan, farklı ağ düğümlerini aynı uygulama altına toplayabilen API bazlı sanallaştırma teknikleri de bu sınıfa dahil edilmelidir [70]. ÖABS 'de oluşan mantıksal ağlar fiziksel ağların üzerinde oluşmaktadır. Pratikte bu ağlar, belli bir ağ içerisinde aynı kaynaklar üzerinden farklı uygulama senaryolarını gerçekleştirmek için veya farklı yönetimsel yapılara sahip uygulamalar arasında etkileşim için kullanırlar. Şekil 2.7 'de genel yapıları verilen ağ bazlı bu teknikler genellikle donanımsal homojenlik, yönetimsel heterojenlik veya tersi senaryolarda işletilirler. Az gelişmiş düğümlerin bulunduğu ağlarda amaca yönelik gateway 'ler ile bu sanallaştırmaların gerçekleştirilmesi mümkün olabilmektedir. Bunun yanında kullanılacak protokol yapısı en önemli kriterdir. Kullanılacak P2P protokolün güvenilirliği ve sürdürülebilirliği başarımı birinci elden etkilemektedir. KABS 'da ise genellikle bir fiziksel WSN ağ üzerindeki sensör düğümler farklı uygulamalara hizmet etmek üzere gruplanmaktadırlar. Bu gruplanmalar genellikle kendi kendini organize edebilecek şekilde (self-organizing) tasarlanırlar.

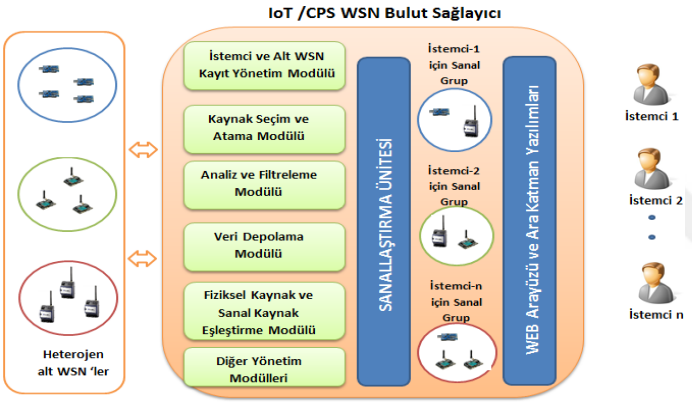


Şekil 2.7. Ağ bazlı sanallaştırma

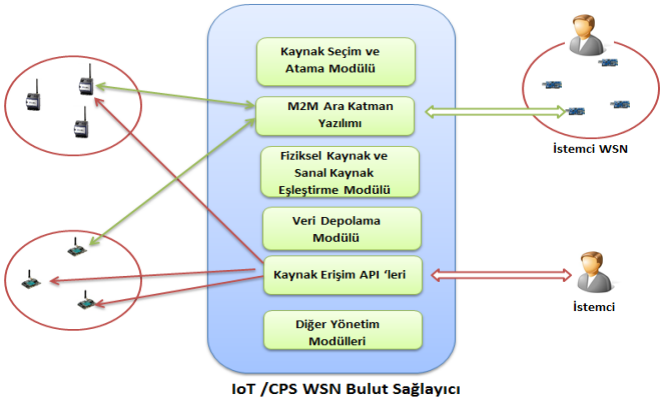
2.3.3. Arakatman Yazılımı Temelli Bulut Sanallaştırma

Gerek DBS teknikleri gerekse de ABS teknikleri daha çok spesifik uygulamalara yönelik önerilmiştir. Bu teknikler, WSN 'lerin genellikle yaklaşık son on yıllık gelişme sürecinde, kısmi heterojenlik içeren veya tamamıyla homojen olan sistemler üzerinde uygulanmışlardır. Bu nedenle bir çoğu IoT desteğini doğrudan sunamamaktadır. Her ne kadar son yıllarda 6LoPAN, RPL, CoAP gibi teknolojiler ortaya çıkmışsa da DBS/ABS tekniklerinin bu teknolojiler ile entegrasyonunu sağlayan bir literatür çalışması; bilindiği kadarı ile; yoktur. Günümüz IoT WSN uygulamaları genellikle bulut teknolojileri üzerinden sağlanmaktadır [71-74]. Özellikle Publish/Subscribe ara katman yazılımları ve RESTFul web servisleri bu bağlamda oldukça esnek bir konfigürasyon yapısı sağlamaktadır. Bunun dışında bir çok IoT ara katman yazılımı bu sistemlere entegre olabilmektedir [75]. Bu tür sistemler farklı istemcilere hizmet verebilen ticari projeler olabileceği gibi kendi içerisinde kapalı ve sadece belli bir amaca yönelik uygulamalar da olabilirler. Bu sistemlerde sanallaştırma teknikleri, genelde arka plan karmaşıklığını soyutlamada ve tek yönlü veri paylaşımı gerçekleştirmede kullanılmaktadırlar. Bu tez, ara katman yazılımı temelli bulut sanallaştırmayı (ATBS) hizmet prensiplerine iki alt sınıfa ayırmaktadır. Bunlar "Veri yönelimli" ve "Etkileşimli" WSN sanal ağlardır. Veri yönelimli sanal WSN ağlar istemci taleplerine göre sağlayıcı sistemin içerdiği farklı alt WSN ağlardaki kaynaklardan oluşan mantıksal gruplardır. Bu tip sanallaştırmada genelde veri akışı alt WSN ağlardan sağlayıcı sisteme ve sağlayıcı sistemden MW 'ler yardımıyla istemcilere olacak şekildedir. Kaynaklar arası etkileşim, bu tip sistem uygulama senaryoları muhtevasında yer almamaktadır. Veri yönelimli ağlarda alt WSN sağlayıcılardan elde edilen ölçüm verilerinin diğer istemcilerle paylaşılması zorunluluğu yoktur. İstemciye özel analiz, değerlendirme ve depolama hizmetleri de sunulabilmektedir. Bunun yanında diğer alt WSN 'lerde seçilen kaynaklardan oluşturulan sanal ağ üzerinden veri alan istemciler, ilk olarak alt WSN sağlayıcı daha sonrada IoT/CPS-WSN sağlayıcının belirlediği sınırlar içerisinde sunulan hizmetlerden faydalanabilmektedirler. Etkileşimli WSN sanal ağlarda ise istemci sağlayıcı tarafta seçmiş olduğu kaynaklarla dinamik bir şekilde etkileşebilmektedir. Bu yöntemlerde genellikle sunucu-istemci tanımlı API 'ler veya M2M ara katman yazılımları tercih edilmektedir. Aslında bu yöntem klasik WSN sistemlerin etkileşmelerinde de kullanılan bir yöntemdir ancak daha önceki klasik yöntemler IoT/CPS kapsamında ölçeklenebilir bir özelliğe sahip değildirler. Aynı zamanda

kısıtlı sayıda farklı WSN ile M2M ara katman yazılımıyla yapılan etkileşimlerde tam olarak WSN heterojenliği giderilmemekte ve IoT ölçeğine getirilememektedir. Yapılan literatür araştırmalarından görüldüğü kadarı ile, tam bir soyutlama sağlayan, ölçeklenebilir, etkileşimli WSN sanal ağlar içeren IoT/CPS WSN sistem tasarımları konusunda ciddi bir boşluk vardır. Bu amaçla bu tezde önerilen sanallaştırma tekniği, bu sınıftaki boşluğu doldurmaya yöneliktir. Önerilen yazılımsal çerçeve sayesinde, özellikle DBS 'de önemli bir kısıt olan algoritma, komut ve sorgu taşınması bu sınıfta IoT/CPS ölçeğinde giderilebilmektedir. Öte yandan ABS tekniklerinde gerekli olan özel P2P protokol ve algoritmalarına da gereksinim bu sınıfta kullanılan teknolojilerle ortadan kaldırılmış olacaktır. Şekil 2.8 a ve b 'de ABTS tekniklerinin genel yapıları ve çalışma prensipleri verilmiştir.



a) Veri yönelimli IoT/CPS WSN sağlayıcı



b) Etkileşimli IoT/CPS WSN sağlayıcı

Şekil 2.8. IoT/CPS WSN sağlayıcı tipleri

Bunlardan VITRO [76], MW tabanlı bir sanallaştırma destekli servis yapılandırma frameworku sunuyor. Hizmet ve uygulamaların birbirinden ayrıldığı, genel bir sunucu mimarisi önermektedir. Kullanıcılar, WSN adaları ve kayıt işlemleri için arayüzler sunan bir sanallaştırma yöneticisi barındırmaktadır. Servis keşfi, kaynak keşfi gibi pek çok katman içeren MW ile birlikte publis/subscribe yapı kullanan sanallaştırma yöneticisi esnek bir model sunmaktadır. VITRO ikinci olarak bir sanal ağ uyumlu bir düğüm modeli de önermektedir. Bu düğüm modelinde her düğüm bir MW içermekte ve bu MW ile servis keşfi yapılabilmektedir. Ancak bu modelde kullanıcılar sağlayıcı tarafında sunulan hizmetlerle sınırlı olup kendi uygulama ve kodlarını hedef WSN içine nasıl dahil edecekleri ile ilgili pratik bir detay sunulmamıştır. Kullanılacak programlama modeli ile ilgili her hangi bir açıklama bulunmamaktadır. Diğer önemli bir çalışmada [77], kısaca MENO olarak adlandırılan "Managed Ecosystems of Networked Objects" bir kavram önerilmektedir. Bu kavram, ağ temelli sanallaştırma tekniği üzerine kurulmuştur. Amacı IP-enabled cihazların end-to-end olarak birbirleri ile haberleşebilecekleri akıllı bir mimari sunmaktır. MENO katmansal bir model üzerine çalışmaktadır. En altta WiFi, Ethernet vb. internet altyapısı üzerine kurulu nesneleri içeren fiziksel katman, fiziksel katmandan seçilen ve belli bir amaca hizmet etmek için bir araya gelen nesnelerin oluşturduğu mantıksal katman ve son olarak end to end iletişimi kotaran end-to-end communication katmanı bulunmaktadır. Ancak bu çalışmada da pratik açıdan genel gerçekleme detayları net olarak sunulmamıştır. Ayrıca kaynak keşfinin nasıl yapılacağı, sürdürülebilirliğin nasıl sağlanacağı, bu sistem içerisindeki mekanizmal yapının ne olduğu detaylıca belirtilmemiştir. Ancak [78] çalışması MENO kavramı, [79] ve [80] mimarisi üzerine kurulu IoT-VN yaklaşımı önerilmiştir. IoT –VN, hem kaynak kısıtlı hem de kaynak kısıtı olmayan aygıtları end-to-end iletişim ile entegre edebilen bir yapıdır. Kısıtlı kaynaklar için IDRA framework 'unu, diğerleri için ClickRouter frameworkunu kullanmaktadır. Ancak bu başarılı çalışmada kullanılan yöntemler günümüz mevcut WSN sistemlerine doğrudan uygulanması zaman ve teknik olarak zorluklar içermektedir. Dahası bu çalışmada WSN sağlayıcıların ihtiyaç duyduğu diğer alt bileşenler (kaynak, servis keşfi, sürdürülebilirlik, loglama vb.) hakkında detaylı bilgi mevcut değildir. Bilindiği üzere iletişim protokollerindeki heterojenlik, adresleme ve isimlendirmedeki karmaşık yapılar farklı ağların beraber çalışmaları önündeki büyük engellerden biridir. Dahası pek çok kısıtlı kaynaklı düğüm IP desteğinden mahrumdur. [81] deki çalışmada tüm fiziksel sensörleri sanal sensörler olarak sunan SensWrap adında bir MW yaklaşımı sunularak VSN 'ler için

bu sıkıntıların giderilmesi amaçlanmıştır. Bu MW, ZeroConf adında bir protokol ile desteklenmiştir. Bu protokol ile sensör keşfi yapılmakta ve standart bir protokol arayüzü sunulmaktadır. SensWrap hem request/reply hem de publish/subscribe yapıları desteklemektedir. Arayüzler TCP/UDP tabanlıdır. [82] deki çalışma ile her ne kadar çok detay verilmese de sensör ağlarda bir P2P overlay alt yapısı sunulmuştur. Bu çalışmada TChord adı verilen protokol ile P2P ağların internete entegrasyonu sağlanacağı ifade edilmiştir. Ancak çoklu uygulamaların çalışma modelleri sunulmamıştır. Sürdürülebilirlik ve performans ile ilgili genel bir açıklama sunulmamıştır. Overlay yapılar ile ilgili bir diğer çalışma ise [83] dır. Ancak bu çalışma "ring overlay" lerde yönlendirme üzerine odaklanmıştır. Sensör düğüm temelli sanallaştırmalarda literatürde kendine belli bir ağırlık kazandırmıştır. SenseEye [84] çalışması kamera sensörlü düğümler içeren uygulamalarda kullanılmıştır. Three-tier temelli bir yapıya sahip olan SenseEye, çoklu uygulamaları bu katmanlar üzerinde çalıştırmakta ve hareketli nesnelerin takibini gerçekleştirmektedir. Bu çalışma belli bir amaca yönelik olup sistemde kullanılan düğümlerin tam bir heterojen yapıya sahip oldukları söylenemez. SenQ [85] nesC ve Java temelli bir başka sanallaştırma çalışmasıdır. Heterojen yapılar için esnek bir sorgulama sistemi amaçlanmıştır. Ancak bu sorgulamalar kısıtlı olup sistemin global bir çözüm sunmamaktadır.

Her ne kadar IoT 'ler için uygulaması ve gerçeklenmesi zor kabul edilse de düğüm bazlı sanallaştırmalar spesifik uygulamalarda etkin çözümler olabilirler. Düğüm bazlı sanallaştırmalar tamamen düğümün donanımsal ve yazılımsal özelliklerine bağlıdır. Kullanılan işletim sisteminin birden fazla farklı uygulamanın düğüm üzerinde çalışmasına olanak sağlayabilmesi veya küçük MW 'ler ve sanal makine yazılımları desteği ile bu yetiyi kazanması temel noktadır. Tabii düğüm kısıtları en büyük çıkmazdır. WSN Gömülü işletim sistemleri (EOS) deyince akla ilk olarak TinyOS, MANTIS ve Contiki gelmektedir. Ancak PAVNET ve LiteOS gibi pek çok farklı EOS 'u literatürde görmek mümkündür[86-87]. TinyOS event-driven bir işletim sistemidir ve WSN 'lerde sıklıkla kullanılmaktadır. Her ne kadar TinyOS son versiyonu ile multi-thread desteği sunuyor olsa da eski versiyonları bunun için TOSTHread gibi yardımcı paketlere ihtiyaç duyuyordu. Event-driven temelli Contiki düğüm bazlı sanallaştırmada kullanılan güçlü bir gömülü işletim sistemdir. Preemtive multithread desteği sunmaktadır. Böylece çoklu görevler daha etkin bir şekilde işlenebilmektedir. Contiki düğümlere kablosuz olarak upload işlemine destek vermektedir. İşletim sistemi bakış açısından homojen sistemler için daha çok uygundurlar. Çünkü farklı WSN 'lerden veya farklı düğümlerden oluşan ağ sistemlerde

beraber çalışabilirlik zorlaşacaktır. MANTIS ve RIOT ise doğrudan multi-thread desteği sunmaktadırlar. Agilla middleware [88] bu TinyOS sistemlerde düğüm sanallaştırmada iyi bilinen bir teknolojidir. Mobile-agent temelli bir yazılımdır ve TinyOS üzerinde çalışır. Uygulamalar round-robin yöntemine göre çalıştırılır. Yazılan agent 'lar kendilerini düğümler arasında taşıtabilir. Ancak bu taşınma ve sisteme dahil olma sürecinin uzun olduğu [32] 'de yapılan deneylerde uzun olduğu belirtilmiştir. Bu alanda yine adından sö ettirmeyi başarmış bir diğer çalışma küçük bir sanal makine olan Mate [89] dir. TinyOS üzerinde çalışır ve amacı minimum enerji harcaması ile kod taşınabilirliğinin sağlanmasıdır. Bunun için küçük kod kapsülleri kullanılmıştır. Ancak daha önce bahsettiğimiz kısıtlar bu düğüm bazlı sanallaştırma teknolojisi içinde geçerliliğini korumaktadır. Bunların dışında VMSTAR, Squawk gibi Java destekli çalışmalarda düğüm bazlı sanallaştırma alanında yapılmış diğer güzel çalışmalardır [90,91].

Bulut WSN sistemler son yıllarda farklı alanlarda teknoloji dünyasında yer almaya başladı. Bunlardan Sentilo projesi, Barcelona 'da belediye hizmetleri için tesis edilmiş açık kaynak sensör platformudur [92]. Özellikle akıllı şehir projelerine yönelik geliştirilmiş bu proje son kullanıcıya yönelik değildir. WSN teknolojileri konusunda girişimci adımlar atan Libelium firması Sensor to Cloud projesiyle genel amaçlı bir WSN bulut sistemi sağlamıştır [93]. Ancak bu IoT WSN projesi markaya yönelik servis hizmetleri sunmaktadır. Bir başka akıllı şehir projesi olan IOTSense, akıllı şehir hizmetleri için genel çevresel ölçümlerde kullanılmak üzere gerçekleştirilmiştir [94]. Atık ölçümü, park bahçe gözetleme gibi servisleri sunan bu proje farklı kaynak heterojen bir alt yapı üzerine kurulmuştur. Kapalı bir sistemdir ve son kullanıcılara hizmet vermemektedir. Windows Azure ve Amazon gibi iyi bilinen bulut sistemler ise WSN bulut hizmetlerini bir alt servis olarak verirler [95,96]. Genel amaçlı bulut sistemler olduklarından spesifik WSN uygulamalarına yönelik özel çözümler sunmamaktadırlar. Veri yönelimli IoT WSN sınıfındadırlar. RESTFul tabanlı web servisler aracılığı ile gerek sabit gerekse mobil kullanıcılarına esnek bir erişim imkanı sağlayan SensorRush bir başka IoT WSN sistemdir [97]. Veri toplama, veri paylaşım ve depolama hizmetleri vermektedir. Bunun yanında farklı heterojen WSN sistemler arasında bir etkileşime destek sağlamamaktadır. Etkileşimli bulut sistemlerinden olan Digi Device Cloud provided by Digi, güçlü bir M2M hizmeti vermektedir [98]. Her ne kadar genel amaçlı bir servis olsa da ZigBee destekli ürünleri için bu etkileşimi sağlayabilmektedir. Servis farklı kullanıcılar ve farklı heterojen sistemler arasında değil, ürün sahibi müşterilerin uzaktan kendi sistemlerini kontrol edebilmelerini

sağlamak için geliştirilmiştir. Bunu Digi Cloud Connector olarak adlandırılan bir yazılım ve kenar cihazlar (edge device) ile gerçekleştirmektedir. Bu projelerin yanında Nimbits, ThingSpeak ve Pachube gibi WSN bulut sağlayıcıları da veri yönelimli hizmetler vermekte ancak heterojen alt WSN sistemler arasında tam kaynak paylaşımlı temelli etkileşim sunamamaktadır [99-101].

2.3.4. WSN Sanallaştırma Tekniklerinin Avantaj ve Dezavantajları

WSN sanallaştırma tekniklerinin uygulanabilirliği çalıştırılacağı senaryo tipiyle alakadır. DBS, ABS ve ATBS 'nin genel özellikleri bunların IoT/CPS WSN sistemler için ne denli uygun olup olmadıklarını göstermektedir. Bu nedenle bu bölümde WSN sanallaştırma tekniklerinin avantaj ve dezavantajları özetlenecektir.

DBS 'nin avantajları:

+ Komutlar düğüm üzerinde çalıştırıldığından hızlıdırlar.
+ Zaman-kritik kaynak paylaşımlı uygulamalarda diğer tekniklere göre daha başarılıdırlar.

DBS 'nin dezavantajları:

− Düğümler kısıtlı donanımsal özelliklere sahip olduklarından dolayı çok sayıda uygulamayı üzerlerinde barındırmaları hem enerji verimliliği hem de sistem performansı açısından sağlıklı değildir.
− Pek çok düğüm hafıza koruma birimlerine sahip olmadığından kodlama çok dikkatli bir şekilde yapılmalıdır.
− Eski tip düğümler kullanan sistemlerde bu tip sanallaştırma kullanılamaz
− Düğümlere çalışma anında yeni uygulama kodlarını, sistemi kesintiye uğratmadan yüklemek zordur. Çoğu düğüm buna destek sunmaz.
− Farklı düğüm teknolojileri ve yazılımları içeren heterojen ağlarda tek tip kodlama yapılamaz. Pek çok düğüm teknolojisi ve gömülü işletim sisteminin varlığı göz önüne alındığında bu ciddi bir öğrenme ve zaman maliyeti ortaya çıkacaktır
− Kod taşınabilirliği ciddi bir problemdir.

DBS daha çok spesifik uygulamalar için uygun olmakla birlikte WSN sağlayıcılar ve bulut sistemler için uygun değildir.

ABS 'nin avantajları:

+ Daha esnek bir kaynak paylaşım ortamı sunar

+ İyi tasarlanmış bir protokol ve gateway ile yönetim bazlı heterojen WSN sistemler arasında etkileşim sağlanabilir. Bu durumda ATBS sistemlerle kıyasla daha iyi bir zaman gecikmesi ve yürütüm zamanı kısalığı sağlayabilirler.
+ Bir WSN ağ içerisinde kümesel sanal ağ oluşturmada verimlidirler.

ABS 'nin dezavantajları:

– ÖABS ağlarda P2P iletişim için özel protokol kullanımlarına ihtiyaç duyulması,
– EOS 'lara ve bazı küçük ara katman yazılımlarına ihtiyaç duyması,
– Eski tip gelişmemiş düğümlerde uygulanamaması veya spesifik API desteğe ihtiyaç duyulması,
– Kullanılan protokollerin tüm WSN teknolojiler tarafından desteklenmemesidir.

ABS teknikleri de genellikle amaca yönelik ve büyük ölçekli olmayan sistemlerde etkinlileri yüksektir. Her ne kadar günümüzde 6LoWPAN, CoAP gibi teknolojiler bu sistemlerin IoT projelerinde kullanılmasına olanak sağlasa da söz konusu bu teknolojilerin başlık yükleri, standart olamayan WSN düğüm teknolojileri tarafından desteklenmemeleri, onları büyük ölçekli IoT/CPS sağlayıcılar için ikinci planda bırakmaktadır.

ATBS 'nin avantajları:

+ Ölçeklenebilir olmaları, veri depolama ve paylaşma gibi ek özellikleri, büyük ölçekli IoT/CPS sistemler için daha uygun yapmaktadır.
+ Sadece veri yönelimli hizmetlere ihtiyaç duyan projeler için kurulum ve işletme maliyetlerini en alt seviyeye indirir.
+ WEB ve ara katman yazılım teknolojileri ile farklı heterojen sistemler için en iyi soyutlamayı sağlar.

ATBS 'nin dezavantajları:

– Gerçek zamanlı uygulamalar için uygun değildirler
– Özellikle veri yönelimli sanal WSN ağlar sınıfında hizmet sunan sistemlerde, sağlayıcı taraflı kısıtlar istemcilere esnek kaynak paylaşım imkanı sunmaz
– İstemciye özel algoritma ve filtreleme işlemleri için özel etkileşimli sanal WSN ağ tasarımlarına ihtiyaç duyar.

Yukarıda açıklandığı üzere bu tez, DBS ve ABS sistemlerin önemli avantajlarını etkileşimli sanal WSN ağ oluşturabilen ATBS'ler taşıyabilecek bir sistem mimarisi önermektedir. Özellikle algoritma taşıma probleminin çözüldüğü ve özel protokollere

ihtiyaç duymayan heterojen kaynak paylaşımlı IoT/CPS WSN sistemler bu tezin ana amacını oluşturmaktadır.

2.4 Büyük Veri Teknolojileri ve WSN 'ler

Günümüzde internet teknolojisinin hemen hemen her yeri kapsaması, milyarlarca cihazın bu teknoloji alt yapısını kullanmaya başlaması, sosyal ağların yaygınlaşması gibi daha pek çok nedenden dolayı depolanacak sayısal veri boyutları devasa boyutlara ulaşmıştır. Facebook, Twiter, dünya borsaları gibi daha pek çok kuruluşun depoladığı veri boyutları artık petabyte 'lar ile ifade edilmektedir. Bu verilerin depolanması ve işlenmesi doğal olarak büyük veri teknolojilerin geliştirilmesine ön ayak olmuştur. Yüksek teknolojili depolama donanımlarının maliyeti, spesifik paralel ve dağıtık sistem teknolojileri bu büyük verilerin depolanmasında ve işlenmesinde yüksek maliyetlere sebep olabilmektedir. Bu nedenle son 10-15 yılda sıradan donanımlar üzerinde büyük veri işlemlerinin gerçekleştirilmesine yönelik çalışmalar önem kazanmıştır. Özellikle maliyet ve veri güvenliği bağlamında son derece etkin çözümler bu süreç içerisinde hayat bulmuştur. Günümüzde Hadoop, Spark, Pachyderm, Google BigQuery gibi büyük veri teknolojileri bu alanda oldukça iddialı teknolojilerdir [102-15]. Özellikle bunlardan açık kaynak kod bir yazılım çerçevesi (framework) olan Hadoop, şu an için büyük veri teknolojileri arasında en etkin olanıdır. Bu framework, pahalı olmayan sıradan donanımlar üzerinde veri depolama ve veri işleme işlemlerini gerçekleştirmek için kullanılır. Gelişmiş özellikleri, geniş kaynak dokümantasyonu ve açık kaynak kodlu olmasından dolayı da bu tez kapsamındaki çalışmalarda, Hadoop frameworku ve desteklediği teknolojiler tercih edilmiştir. Bu maksatla bu bölümde ağırlık olarak bu teknoloji ve bu teknolojiyi tamamlayan/beraber çalışabilen bileşenler hakkında genel bilgiler sunulacaktır.

2.4.1. Büyük Veri ve WSN Entegrasyonu

Klasik sensör ağ uygulamalarında elde edilen sensör verileri kaynak tipine ve düğüm özelliklerine bağlı olarak üç şekilde depolanır [106];

a) WSN üzerinde dağıtık olarak
b) WSN dışında merkezi olarak
c) Hibrid olarak

Bunlardan ilkinde, toplanan veriler sensör düğümler üzerindeki kaynaklarda (hafıza, hafıza kartları vb.) tutulur ve gerekli sorgular/komutlar alındığında bu bilgiler paylaşılır. Bu daha çok sensör sayısının çok olduğu uygulamalarda kullanılan bir yöntemdir. İkinci yöntem genellikle az sayıda sensör içeren sistemlerde tercih edilmektedir ve toplanan veriler merkezi bir noktada (genelde WSN 'nin kontrol biriminde) depolanmaktadır. Son yöntem ise ilk iki yöntemin kombinasyonudur.

WSN 'lerde kullanılan bu veri saklama yöntemleri, enerji sarfiyatı, ve WSN kaynakları sayısı bakımından büyük ölçekli IoT/CPS sistemlerde kullanılmaz. Şöyle ki, düğüm sayısı "n" adet olan ve periyodik çalışma modeli ile çalışan bir "W" WSN sistemi ele alınsın,

$$W = \{d_1, d_2, .. d_n\}, \quad 1 \leq i \leq n, \tag{2.1}$$

$$T = \{t_1, t_2, t_3 .. t_j\}, \quad j \geq 1, \tag{2.2}$$

Bununla birlikte bu ağdaki sensör kaynak tipleri "T" kümesi ile temsil edilmektedir. Örnek olarak "t_1" analog LM35 sıcaklık sensörünü, "t_2" dijital DS18B20 sıcaklık sensörünü, "t_3" ise ADXL345 3-axis accelerometer kaynaklarını göstersin. "j" indeksi "W" deki tanımlı tüm kaynak tiplerinin sayısını göstermektedir. "W" 'deki kaynak varlık matrisi "Ë" ile ifade edilirse bu durumda;

$$Ë = \begin{bmatrix} k_1^{d_1} & k_2^{d_1} & .. & k_j^{d_1} \\ k_1^{d_2} & k_2^{d_2} & .. & k_j^{d_2} \\ .. & .. & .. & .. \\ k_1^{d_n} & k_2^{d_2} & .. & k_j^{d_n} \end{bmatrix} \tag{2.3}$$

Şeklinde gösterilecektir. Bu matrisin her bir satırı bir "W" de tanımlı bir düğümü gösterirken, her bir sütun ağda tanımlı kaynak tiplerinin ilgili düğüm üzerindeki varlık durumunu göstermektedir. Bu matriste;

$$k_j^{d_i} = \begin{cases} 1 & , \text{ eğer } t_j \in d_i \\ 0 & , \text{ aksi durum} \end{cases} \quad (2.4)$$

Olarak tanımlı olacaktır. Öte yandan her bir kaynak tipinin ürettiği veri boyutunu (b_j) gösteren ön tanımlı "B" matrisi ise şöyle gösterilmektedir;

$$B = [b_1 \quad b_2 \quad .. \quad b_j] \quad (2.5)$$

Bununla birlikte ağdaki düğüm aktiflik pasiflik durumunu gösteren düğüm durum matrisi ise şöyle ifade edilir;

$$N_s = [s_{d1} \quad s_{d2} \quad s_{di} .. \quad s_{dn}] \text{ , } \begin{matrix} s_{di} = 0 \text{ , Eğer } d_i \text{ uyku veya kapalı durumda ise} \\ s_{di} = 1, \quad \text{Eğer } d_i \text{ aktif durumda ise} \end{matrix} \quad (2.6)$$

Bu durumda bu sistemdeki tüm düğümlerin bir " P " sürelik zaman diliminde (f) çalışma oranı ile tüm kaynaklarından veri topladığı ve saatlerinin senkron olduğu kabul edildiğinde, toplanan veri boyutu;

$$D = fx\left[N_s \text{ x } [\ddot{E} \text{ x } B^T]\right] \quad (2.7)$$

olacaktır. WSN düğümlerde enerji tüketiminin en yüksek olduğu işlem radyo operasyonlarında gerçekleşmektedir. [107] 'deki enerji tüketimi modeli ele alındığında, iletim esnasında enerji tüketimi ve alım esnasındaki tüketimE^{Rx} için,

$$E^{Tx} = b_i \text{ x } (\text{J}N + \text{Ɛ x } d^2) \quad (2.8)$$

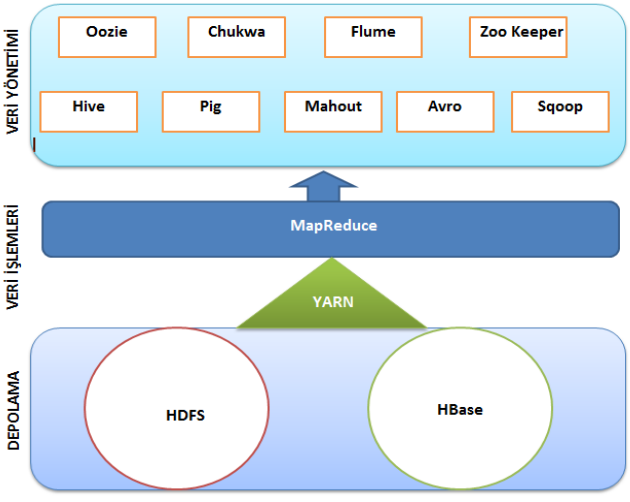
$$E^{Rx} = b_i \text{ x J}N \quad (2.9)$$

Burada Ɲ ve Ɛ , sırasıyla radyo devresi ve yükselteci sabitesidir. Düğümlerin gönderip alacağı veri boyutu B_{ih} ve transmisyon mesafesi "d" temel değişkenlerdir.

Görüleceği üzere n, j ve f değerlerinin büyük boyutlara ulaşması, veri boyutunu ve dolayısıyla sistemdeki enerji sarfiyatını arttıracaktır. Bu nedenle klasik WSN 'lerde kullanılan veri depolama yöntemlerinin IoT/CPS WSN sistemlerinde kullanılması elverişsiz olacaktır. IoT/CPS ölçekli büyük WSN sağlayıcı sistemlerde içerilen kaynak sayısı ve veri akışı dikkate alındığında büyük veri teknolojilerinin kullanılması gerekecektir. Yukarıda açıklık getirilen nedenlerden dolayı bu tez çalışmasında sunulan ATBS sistem, günümüzün önde gelen büyük veri teknolojilerinden olan Hadoop 'u kullanmaktadır.

2.4.1. Hadoop

Hadoop, sponsorluğu Apache Software Foundation (ASF) tarafından sağlanan Apache projesinin bir parçasıdır. İlk olarak 2000 'li yılların başlarında geliştirilmeye başlanan bu proje, bugün birçok modül, komponent ve yazılım paketiyle büyük veri teknolojilerinde oldukça önemli bir yere sahip olmuştur. Hadoop, kernel olarak "Hadoop Common" u kullanmaktadır. Bu framework 'un en temel kütüphaneleri buradan sağlanmaktadır. Diğer özellikler genellikle ek modüller ve Hadoop projeleri tarafından sağlanmaktadır. İçerdiği bu çözüm paketlerinden dolayı genellikle bir bilişim teknolojileri ekosistemi olarak lanse edilir. Şekil 2.9, bir bölümü tez çalışmalarında da kullanılan bu ekosistemin genel bileşenlerini göstermektedir.



Şekil 2.9. Hadoop Ekosistem

Apache Oozie, Haddop işlerinin (Jobs) yönetilebilmesine olanak sağlayan bir iş akışı düzenleme sistemidir. Birden fazla işin birleştirilerek belli bir düzen içerisinde çalıştırılmasına imkan verir. Bu görev sırasında birdan fala iş birbirleri ile paralelel çalışmaları için programlanabilirler. En önemli avantajı, Hadoop ekosisteminin diğer modülleri/projeleri ile entegre olarak çalışabildiklerinden daha verimli bir çalışma ortamı sağlayabilmeleridir. Bir Hadoop işi ile ilgili tüm süreçleri callback (geri çağrım) ve polling (sıralı bekletme) ile gerçekleştirir. Gerçek zamanlı çalışmada cluster 'lar ile ilgili durum bilgilerine ulaşmanın imkansız olmasından dolayı Apache Hadoop 'un operasyonel işlemleri ile ilgili takip işlemlerini Hadoop MapReduce ile yapmak oldukça elverişsizdir. Bunun amaçla Chuckwa projesi Hadoop loglarının toplanması ve analiz edilmesi için geliştirilmiştir. Chuckwa bunun için bazı alt araçlar içermektedir ve alt yapı olarak HDFS 'i kullanmaktadır. Sadece Hadoop değil diğer dağıtık sistemlerle de çalışabilen ölçeklenebilir bir projedir.

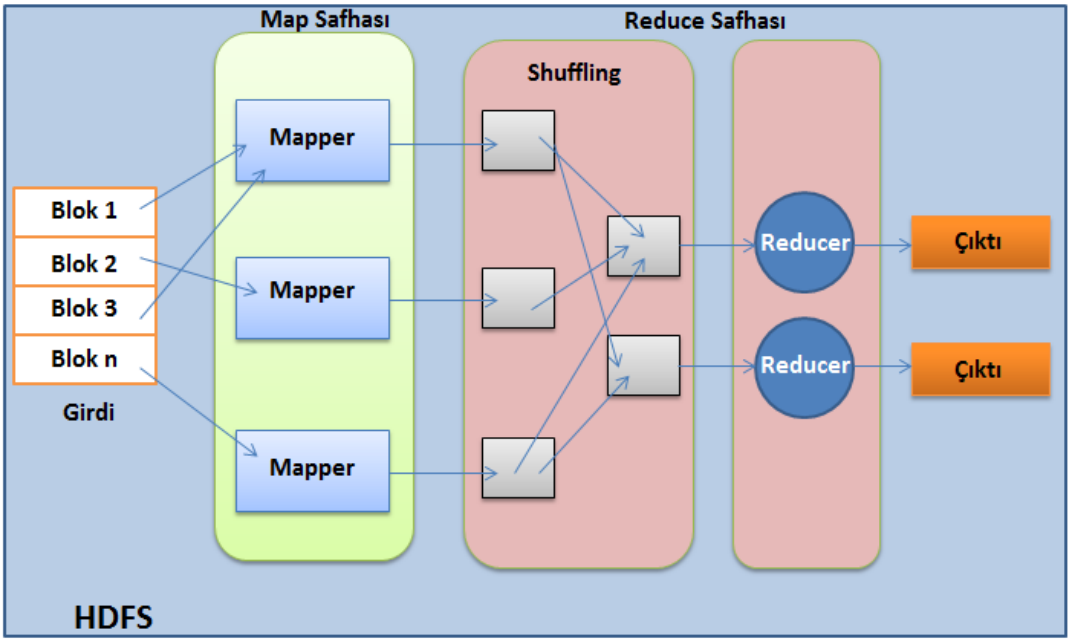
Hadoop ekosistem içerisinde log yönetim için geliştirilmiş dağıtık sistem mekanizması Flume 'dur [108]. Flume, farklı kaynaklardan log verilerini toplar, gereksiz log verilerini ayıklar ve HDFS 'e yazar. Data akışını takip edebilme yeteneğine sahip, esnek ve kolay bir programlama modeli sunar. Temelde Chuckwa ile benzer amaçlara sahiptir. Ancak Flume data akışı için merkezi bir liste kullanırken, Chuckwa bu bilgiyi genellikle kendi servisleri arasında dağıtır. Dağıtık senkronizasyonu ZooKeeper, grup servisleri, adresleme ve konfigürasyon bilgileri yürütümü içi geliştirilmiş merkezi bir sistemdir [109]. Bu servislerin farklı uygulamalarda bulunmaları ve çalışma anlarında oluşacak bug veya yarışma temelli problemleri ortaya çıkarması kaçınılmaz olduğundan bu tip bir merkezi yönetim sistemi projesi geliştirilmiştir. Bu sistem sayesinde dağıtık uygulamalar arasında gerekli cluster yönetimi, senkronizasyon ve yürütüm sağlanabilmektedir.

Apache Hive SQL benzeri bir arayüz ile depolanmış veriler üzerinde sorgulama yapılmasına olanak sağlayan bir projedir [110]. İlk olarak Facebook tarafından kullanılan bu proje daha sonra ASF tarafından daha da geliştirildi. Şu an Amazon gibi birçok teknoloji devi tarafından kullanılmaktadır. Geliştirilen sorgu dili HiveQL veya HQL oldukça hızlı ve kolay bir çalışma ortamı sağlamaktadır.

Pig, Hadoop 'un programlama modeli olan MapReduce üzerine bir soyutlama sağlayarak büyük veri setlerinin daha ayrıntılı analiz edilmesini sağlar [111]. Bu tip analiz programlarını yazmak için "Pig Latin" adı verilen yüksek seviye bir dil kullanmaktadır. Bu

dil yardımıyla oluşturulan scriptler daha sonra Map ve Reduce görevlerine dönüştürülerek analiz işlemi gerçekleştirilir. Bu operasyonları gerçekleştiren ise "Pig Engine" adı verilen bir komponentdir. Özellikle çoklu sorgulamaya izin verdiğinden uzun java kodları ile gerçekleştirilecek olan analiz işlemleri Pig Latin ile daha az sayıda ve karmaşık olmayan kodlamalar ile hızlıca yapılabilmektedir. Makine öğrenme algoritmalarının büyük veri setleri üzerinde kullanılmasına imkan veren açık kaynak kod Apache projesi ise Mahout 'dur. Mahout ile Hadoop kütüphanelerini kullanarak ölçeklenebilir uygulamalar gerçekleştirmek mümkündür. Mean-Shift, k-means, Fuzzy k-means gibi bazı kümeleme ve Dağıtık Naive Bayes gibi sınıflandırma uygulamaları dahili olarak yüklenir. Bunun yanında gelişmiş matris ve vektör kütüphaneleri de içermektedir. Apache Avro bir veri serileştirme sistemidir. Şema tabanlı bir sistem olup, veriyi ön tanımlı bir şema üzerine serileştirmektedir. Serileştirme işlemi binary formatındadır ve her hangi bir uygulama ile tekrardan ters serileştirme işlemi (deserialization) gerçekleştirilebilir. Java, C,C++, Python gibi dil desteği sunan Avro, veri yapılarının deklarasyonunda JSON formatını kullanır. Apache Sqoop yığınsal verilerin transferini verimli bir şekilde gerçekleştirmek için geliştirilmiş bir modüldür [112]. Bu araç sayesinde Hadoop ile ilişkisel veri tabanları arasında veri transferleri gerçekleşebilir. Sqoop verilerin içe veya dışa aktarılmasında MapReduce operasyonları kullanmaktadır. Bu işlemlerin paralel ve yüksek hata toleranslı olarak gerçekleştirilmesini sağlar.

MapReduce dağıtık hesaplama için bir programla modelidir. Hadoop 'un en önemli bileşenleri arasında yer alır [113]. MapReduce algoritmaları iki önemli görev (task) içermektedir. Bunlar "Map" ve "Reduce" dur. Map, veri setlerini alarak başka bir veri setine dönüştürür. Bunu yaparken her bir anlamlı veriyi parçalayarak, <**anahtar** (key), **değer** (value)>, değişkenler grubu (tuple) olarak saklamaktadır. Daha sonra Reduce task 'i Map 'in çıktısını kendisine girdi olarak alarak bu değişken gruplarından ve daha küçük değişken grubu setleri elde eder. Reduce görevleri daima Map 'in icrasından sonra işletilir. Yazılan bir MapReduce uygulaması bir cluster içerisindeki binlerce makinede çalıştırılabilir şekilde ölçeklenebilir. Şekil 2.10 'da görüldüğü üzere bir MapReduce uygulaması yürütümü gerçekleştirilirken üç aşama yürütülür, Map, Shuffle ve Reduce. Bunlardan Shuffle aşaması Reduce safhasının bir bölümüdür ve Map 'den gelen verilerin Reduce 'lara dağıtılmasında yürütülürler.



Şekil 2.10. MapReduce genel çalışma prensibi

YARN (Yet Another Resource Negotiator), Hadoop 2 ile birlikte gelen ve MapReduce operasyonlarının geliştirilmesini sağlayan bir cluster kaynak yönetim sistemidir [114]. YARN 'nın sağlamış olduğu API 'ler doğrudan kullanıcı kodları ile kullanılmazlar. Bunun yerine programcılar, daha üst seviyede yer alan modüllerin sağlamış olduğu API 'leri kullanırlar. Özellikle Spark ve Storm gibi güçlü teknolojiler YARN 'nın sağlamış olduğu API 'ler ile doğrudan etkileşirler. Apache HBase dağıtık, ölçeklenebilir Hadoop veri tabanıdır [115]. Büyük boyutlu veri yapılarına rastgele hızlı erişime imkan verebilmektedir. Google Big Table ile benzer bir yapıya sahiptir. HDFS üzerine inşa edilmiştir ve sütun (column) bazlı bir veri tabanıdır. Bir veri, HDFS 'e direk veya HBase üzerinden veri yüklenebilir ve HDFS ile direk yapılan işlemlere göre daha hızlı işlevlere sahiptir. Hadoop Distributed File System (HDFS) sıradan donanımlar üzerinde çalışabilen ve Hadoop ekosisteminin en temel bileşenlerinden olan dağıtık dosya sistemidir. Ekonomik ve yüksek hata toleransına sahip olması onu diğer dağıtık dosya sistemlerine göre daha avantajlı hale getirmektedir. HDFS, büyük verilerin farklı makineler üzerinde depolanmasına imkan sağlar ve paralel hesaplamaya destek verir.

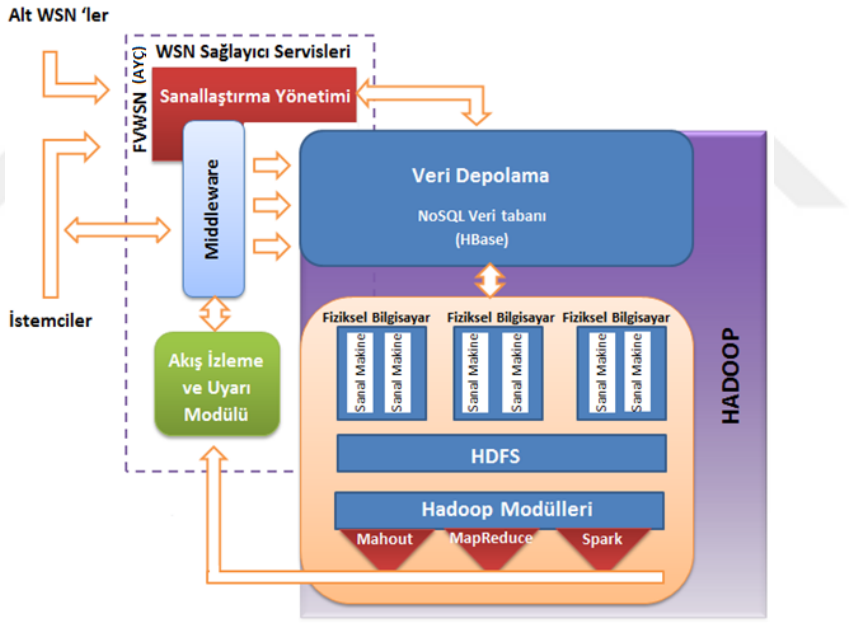
3. ÖNERİLEN IoT/CPS WSN SAĞLAYICI MİMARİ MODELİ

Bu tez iki kriter üzerine odaklanmaktadır. Bunlardan ilki farklı heterojenlik sınıflarındaki WSN ağları içeren IoT/CPS WSN sağlayıcı sistemlerde istemciye dinamik etkileşimli bir kaynak paylaşımı seçimi imkanı sunmak ve istemci tarafından bunun tek bir sistem olarak görülmesini sağlamaktır. Bu problemin çözümü için WSN sanallaştırma teknikleri baz alınmaktadır. Bölüm 2 'de de bahsedilen DBS (Düğüm Bazlı Sanallaştırma) ve ABS (Ağ Bazlı Sanallaştırma) temelli çözümler IoT/CPS WSN 'ler için elverişli değildirler. ATBS temelli teknikler bu amaç için daha uygundur. Ancak ATBS tiplerinde bulunan en önemli eksiklik, DBS ve ABS 'deki hızlı veya doğrudan etkileşimin daha zayıf olmasıdır. İstemcilerin veya istemcilerin ağındaki düğümlerin, hiçbir bilgi sahibi olmadıkları farklı ağlardaki kaynaklarla, istemci taleplerine uygun bir şekilde haberleşebilmesine yönelik çözümler konusunda literatürde bir boşluk vardır. Bu problemin çözümüne yönelik olarak, sunucu taraflı çalışabilen ve istemci talepleri doğrultusunda altyapısı soyutlanmış kaynaklarla etkileşim imkanı sunabilen bir yazılım framework 'ünün kullanılabileceği önerilmektedir. Bu tip bir framework bu tez kapsamında "Adaptör Yazılım Çerçevesi -AYÇ" olarak adlandırılmaktadır. AYÇ 'nin üç ana görevi vardır. Birincisi, istemci tarafında programlamaya olanak tanıyarak, istemci WSN ağındaki veya istemcinin kendi belirlediği yönetimsel/algoritmik işlevleri sağlayıcı üzerinde çalıştırılabilmesine imkan sağlamaktır. Böylelikle istemci veya istemci WSN ağındaki her hangi bir düğümden gelecek komutlar/sorgular, bir insan müdahalesi olmadan, sanki sağlayıcı tarafta istemci ağına ait bir düğüm varmışçasına (istemci sanal düğüm-VNd) değerlendirilebilecektir. AYÇ 'nin ikinci görevi, istemci VNd 'lerinin sunucu sistemin normal işleyişini kesintiye uğratmaksızın devreye alınmasını, yönetilmesini ve oluşacak VNd hatalarından kaynaklanan problemlerin diğer VNd operasyonlarını etkilememsini sağlamaktadır. Bu tip AYÇ kullanan IoT/CPS WSN sağlayıcılarda, istemci sayısı ve tipi dinamik olduğundan işletilen VNd 'lerin kontrollü çalışması hayati bir öneme sahiptir. Bu nedenle AYÇ modülü sağlayıcı sistemde bulunan izleme/kontrol modülleri ile uyumlu çalışmalıdır. Buna ek olarak VNd 'ler diğer VNd 'ler ile sistem şartları altında haberleşebilmelidirler. Önerilen sistem mimarisinin üçüncü görevi ise sağlayıcı alt WSN 'lerdeki alt yapı karmaşıklığını istemciden tamamen gizleyerek kaynak bazlı çalışmaya imkan vermesidir. Özellikle, sisteme yeni eklenecek/çıkacak alt WSN ağların

(veya bunlara ait kaynakların) istemci VNd tarafından sorgulanabilmesine kolaylık sağlamalıdır.

Tezin odaklandığı ikinci ktiter ise bu tip bir AYÇ kullanan IoT/CPS WSN sağlayıcılarda oluşabilecek büyük verilerin nasıl işleneceğidir. Buna yönelik olarak, günümüz dağıtık veri teknolojilerinin, önerilen sağlayıcı sisteme verimli bir şekilde entegrasyonu amaçlanmaktadır. Bu entegrasyonu gerçekleştirirken, AYÇ 'nin sağladığı avantajlarla, gereksiz artık verilerin sistemde depolanmaması, sağlayıcıdan beklenen bir özelliktir. Kullanılan büyük veri teknolojisinin ölçeklenebilir, ekonomik ve analiz kabiliyetlerinin olması, önerilen sağlayıcı sistemi daha uygulanabilir yapacaktır.

Tezde önerilen IoT/CPS WSN sağlayıcı sistemin genel yapısı Şekil 3.1 'de verilmiştir. Mimari, WSN Sağlayıcı servisleri (WSNSS) ve büyük veri işlem servisleri (Hadoop) olarak iki ana bölümden meydana gelmektedir. Bundan sonraki alt başlıklarda bu bölümlere ait genel çalışma yapıları, geliştirilen AYÇ ve sanallaştırma yönetimi açıklanacaktır.

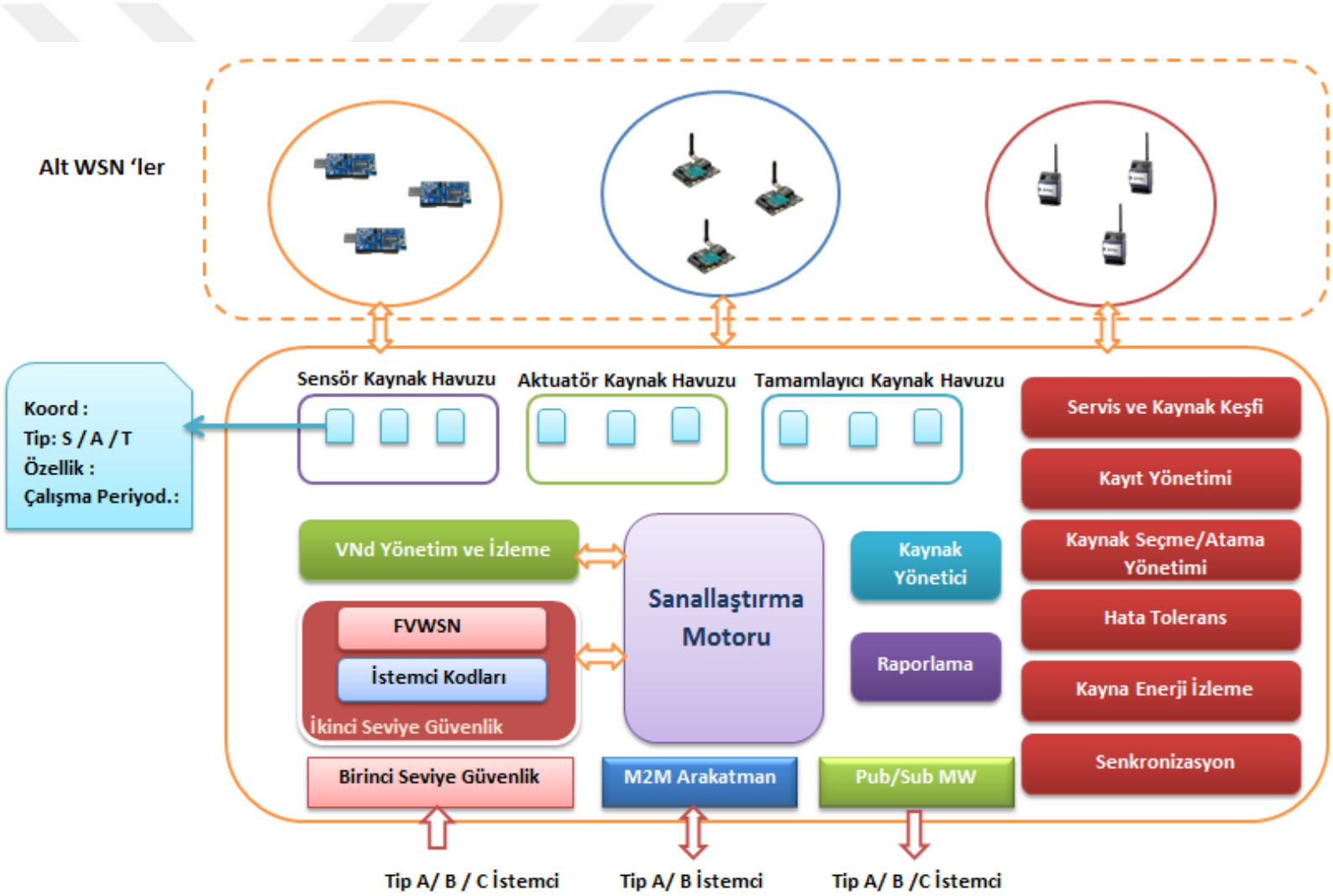


Şekil 3.1. Önerilen mimarinin genel yapısı

Bu bölümde ilk olarak WSNSS bölümü açıklanacaktır. Bu bölüm, IoT/CPS WSN sağlayıcı sistemin temel fonksiyonlarını ve hizmetlerini içermektedir. Ayrıca heterojen WSN'ler arasında beraber çalışabilirliği sağlamak için geliştirilen AYÇ'ye ait detayları da içermektedir. Önerilen sistemin ikinci ana parçası olan büyük veri işlem bölümü Bölüm 6 'da açıklanacaktır.

3.1 WSN Sağlayıcı Servisleri

Şekil 3.2 ‘de detay yapısı verilen WSNSS, bir WSN sağlayıcıdan beklenen temel servisleri ve geliştirilen sanallaştırma yönetim modüllerini içermektedir. WSNSS bölümünün üç farklı etkileşim yüzü bulunmaktadır. Bunlardan biri alt WSN sistemlere, diğeri istemcilere ve sonuncusu ise veri yönetim birimine bakan yüzüdür. Alt WSN sağlayıcılar sisteme dinamik olarak dahil olup ayrılabilen otonom varlıklardır. Alt WSN sistemlere ait operasyonel çalışma şartları, sağlayıcı ve alt WSN sağlayıcı tarafından önceden belirlenmek zorundadır. Alt WSN sağlayıcılar, sahip oldukları kaynak tiplerini, ağlarındaki düğümlerin koordinatlarını, ölçüm mesafe çaplarını, hangi kaynaklarını paylaşıma açacaklarını, hangi şart ve zamanda buna izin vereceklerini ve gerekli API ‘leri IoT/CPS WSN sisteme sağlamaktadırlar. Ayrıca önerilen modelde, sağlayıcı sistem, alt WSN sağlayıcılardan kendi kaynakları ile ilgili ek spesifik bilgiler (kaynağın ürettiği veri büyüklüğü, MIPS değeri vs.) talep edebilir veya bu bilgileri ön tanımlı veriler kullanarak, alt WSN sağlayıcı bağımsız set edebilir. Alt WSN sağlayıcılar ticari veya gönüllü varlıklar olabildikleri gibi doğrudan sağlayıcı sisteme de ait olabilmektedir. Alt WSN sağlayıcılara ait sistem karmaşasının soyutlanmasında en önemli aşamalardan biri, alt WSN ağlara ait düğümler üzerinde bulunan kaynakların, tiplerine göre kimliklendirilerek kaynak havuzunda saklanmasıdır. IoT/CPS WSN sağlayıcı sistem envanterindeki kaynakları üç ayrı havuzda sınıflandırmaktadır. Bunlar “Sensör Kaynak Havuzu”, “Aktuatör Havuzu” ve “Tamamlayıcı Kaynaklar Havuzu”. Bu havuzlamanın bir diğer sebebi de birbiri yerine kullanılabilecek kaynakların belirlenerek olası bir hata durumunda birbirlerinin yerine kullanılabilir olmasını sağlamaktır. Bu özellik hata tolerans modülünde ayrıca açıklanacaktır. Sunucu sistem, envanterdeki kaynakların takibinden birinci elden sorumludur. Alt WSN ‘ler ile sağlayıcı sistem arasında “kalp atışı” fonksiyonlarına sahiptir. Bu fonksiyonlar özellikle sanallaştırma bölümünde yürütümü geciken veya başarılmayan kaynakların kontrolünde kullanılmaktadır. Bu fonksiyonlar istemcilerden tamamıyla soyutlanmış olup AYÇ kapsamında değildirler.



Şekil 3.2. WSNSS (WSN sağlayıcı servisleri) genel yapısı

Her bir kaynak kendi spesifik bilgilerinin bulunduğu bir tanımlama dokümanına (Kaynak Tanımlama Dokümanı- KTD/RDD) sahiptir. Dokümandaki spesifik bilgiler; koordinat bilgisi, sensör veya aktuatör tipi, hassasiyet, sürekli zamanlı ve kısmi çalışma durumu, çalışma periyotları gibi bilgileri içermektedir. Bunlar alt WSN sistemlerin, sunucu sisteme kaydolurken oluşturdukları dokümanlardır. SensorML [116] veya farklı web teknolojileri vasıtasıyla bu dokümanlar istemcilere internet üzerinden sunulmaktadır. Kullanıcılar kaynak havuzunda bulunan kaynaklardan seçtikleri kaynakları, FVWSN yardımıyla oluşturacakları VNd 'ler ile yönetmektedirler. Kaynağın özellikleri, istemci uygulaması ile doğrudan örtüşmek zorundadır. Şöyle ki, kısmi süreli çalışma modeli ile çalışan bir kaynağı yöneten bir VNd ile etkileşecek bir istemci veya istemci WSN 'nindeki bir düğüm, kendi haberleşme düzenini, RDD dokümanlarındaki zaman bilgilerine göre gerçekleştirmelidir. Kaynak tanımlamaları iki tür bilgi içermektedir;

- Kaynak fiziksel özellikleri
- Kaynak-uygulama özellikleri

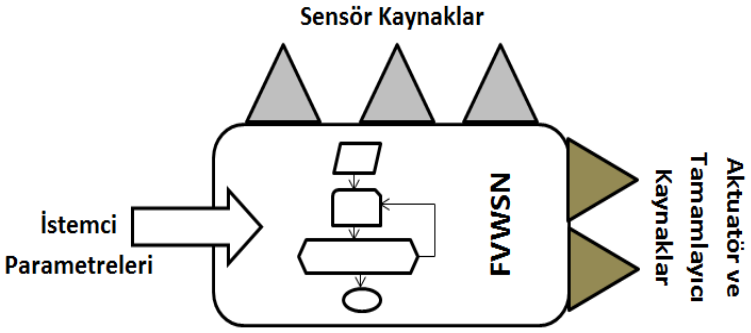
Kaynak fiziksel özellikleri, alt WSN sağlayıcının deklere ettiği verilere göre oluşturulurken, kaynak-uygulama özellikleri hem alt WSN hem de istemci kriterlerine göre oluşturulur. Kaynak fiziksel özelliklerinin tamamı istemcilerle doğrudan paylaşılmaz. İstemci uygulama senaryoları ve sağlayıcı sistemin istemci parametrelerini irdelemesi sonucunda hangi kaynağın VNd 'ler ile kullanımına uygun olacağı belirlenmektedir. Kaynak notasyonu ve değerlendirilmesi sistemde kullanılan kayıt/ kaynak yönetimi ilgili teknik ve matematiksel detaylar Bölüm 4.3 'de verilecektir. Kaynaklar sensör, aktuatör, hafıza kartı gibi çeşitlilik gösterebilmektedir. RDD 'lerede, özellikle aktuatör veya hafıza kartı gibi kaynaklar için gerekli izin ve kullanım şartları da bildirilmektedir. Farklı tip kullanıcıların, aynı aktuatör veya veri kayıt kaynaklarını kullanması belli bir denetim içerisinde yapılmaktadır. Bu işlemlerle ilgili detay Aktuatör/Veri Kayıt Kaynakları Yönetimi bölümünde verilecektir. Sisteme dinamik olarak eklenen kaynaklar, sistemin özellikle kaynak keşfi işlemlerini doğrudan etkilemektedir. FVWSN 'nin en önemli avantajlarından birisi de off-line olarak yazılan VNd kodlarında yapılacak kaynak keşfi işlemlerini, ilgili VNd 'nin sanallaştırma motoru tarafından yürütülmesi esnasında otomatik olarak güncel verilere göre gerçekleştirebilmesidir. Kaynak envanterindeki değişimler ve düzenlemeler, sistemin VNd yürütüm şekline göre önbellekleme (caching) ve doğrudan çağrım (calling) metotları ile yürütülmektedir.

3.1.1. Sanal Düğümler

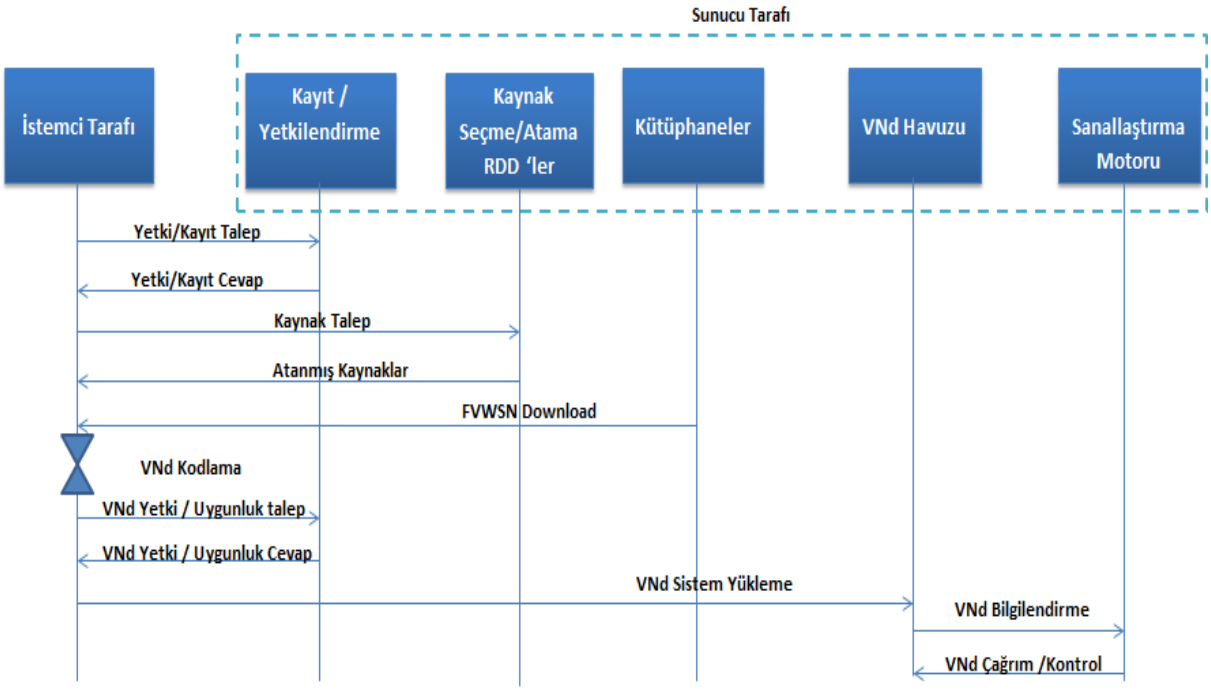
İstemci tarafından RDD verilerine göre seçilen kaynaklara sanal linkler vasıtasıyla bağlı olan, bu kaynakların yönetimi için istemci kodlarını içeren mantıksal düğümlerdir. Bu düğümlerin yürütümleri fiziksel makinalar üzerinde gerçekleştirilirken sanal görüntüleri ise kaynakların yönetildiği lokasyona düşmektedir. Bir sanal düğüm (VNd), sistem tanımlı AYÇ (FVWSN) yardımıyla oluşturulur. Bir VNd kendi istemcisi veya istemci taraflı düğümlerinden gelen komut/sorguları tanımlayabilir, kendisine bağlı kaynakları yönetebilir ve istemci tanımlı algoritmaları işletebilir. Oluşturulan VNd yardımıyla bir istemci kendi WSN broadcast alanını minimum maliyetle genişletme imkanına sahip olabilmektedir. Bir VNd 'nin mantıksal yapısı Şekil 3.3 a 'da verilmiştir. İstemciler tarafından oluşturulan VNd 'ler kaynak havuzundaki kaynakları yönetebilecek kabiliyetlere FVWSN sayesinde haizdir. Sensör kaynakların yönetimi ile aktuatör veya tamamlayıcı kaynakların yönetimi birbirinden farklı işleyişlere sahiptir. Bun nedenle RDD 'lerde kaynak yönetimi için gerekli özellikler sağlanmalıdır. VNd 'ler temelde iki işi bir arada veya ayrı ayrı gerçekleştirebilir. Bunlardan ilki istemciden alınan kodları / sorguları çözümlemek ve bu komutlarla kaynakları yönetmek, ikincisi istemci tarafından kodlanmış algoritmaları gerçekleştirmektir. VNd 'ler kullanıcı tipine göre iki şekilde oluşturulur. Sonraki bölümlerde açıklanacağı üzere Tip A,B ve C kullanıcılar farklı yetkilere sahip istemcilerdir. Bunlardan Tip A ve B sanal düğüm oluşturma ve kullanma yetkisine sahip istemcilerdir. Tip A kullanıcılar tam yetkili iken Tip B kullanıcılar sınırlı VNd yetenekleri ile sistemden faydalanırlar. Şekil 3.3 b ve c, Tip A ve B kullanıcılar için VNd 'lerin oluşturulma sürecini göstermektedir.

Tip A kullanıcılar için VNd oluşturmada ilk süreç kullanıcının yetki tanımının belirlenmesi ile başlar. Gerekli yetkiye sahip kullanıcı, seçmiş olduğu kaynakları ve kendi uygulama parametrelerini sunucu sisteme ileterek kaynak talebinde bulunur. Bir VNd sunucu tarafından belirlenen sayıda ve kriterde kaynak yönetebilmektedir. Bu nedenle fazla sayıda kaynak kullanımı fazla sayıda VNd oluşturulmasını gerektirmektedir. Bununla birlikte bir birinden çok uzak lokasyonlardaki kaynakların tek bir VNd tarafından yönetilmesi, sağlayıcı sistem tarafından sınırlandırılmıştır. Sağlayıcı sistem, farklı lokasyonlardaki kaynaklar için çalışma yarı çapları ve uygulama tipine göre ayrı ayrı VNd ler tarafından yönetilmesini istemcilerden istemektedir. Bu işlem, sistem performansı ve güvenlikle ilgili bir sınırlama olup detayları sistemin matematiksel modeli bölümünde verilecektir. VNd düğümün kodlanması için istemcinin AYÇ (FVWSN) dosyalarını sunucudan alması ve kendi yazılım geliştirme ortamına (IDE) yüklemesi gerekmektedir. Sonraki bölümde açıklandığı şekilde kodlamaları

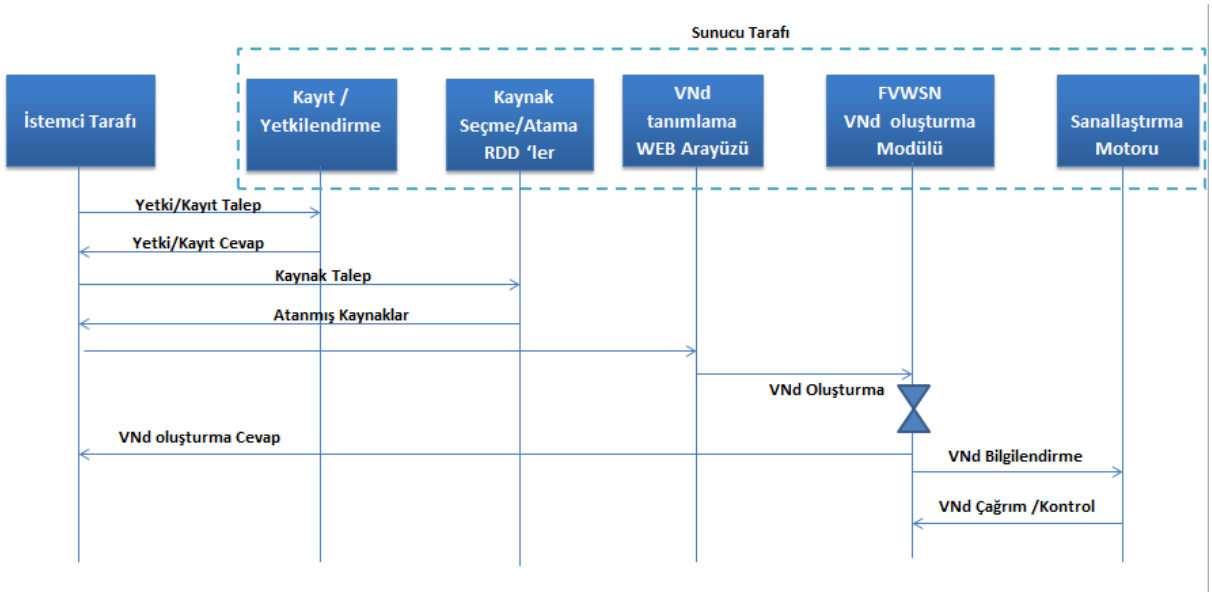
yapılan VNd ‘ler, ikinci seviye güvenlik kontrolleri için sağlayıcı sistem yetkilendirme bölümüne sunulurlar. Gerekli kontrolleri geçen VNd ‘ler, sistem VNd havuzuna yüklenerek gerekli sunucu bildirimleri yapılır. Sanallaştırma motoru (VEngine) diğer sistem operasyonlarını kesintiye uğratmaksızın istemci VNd lerini devreye alır.



a) Sanal düğüm (VNd) mantıksal yapısı



b) Tip-A kullanıcı için VNd oluşturma süreci



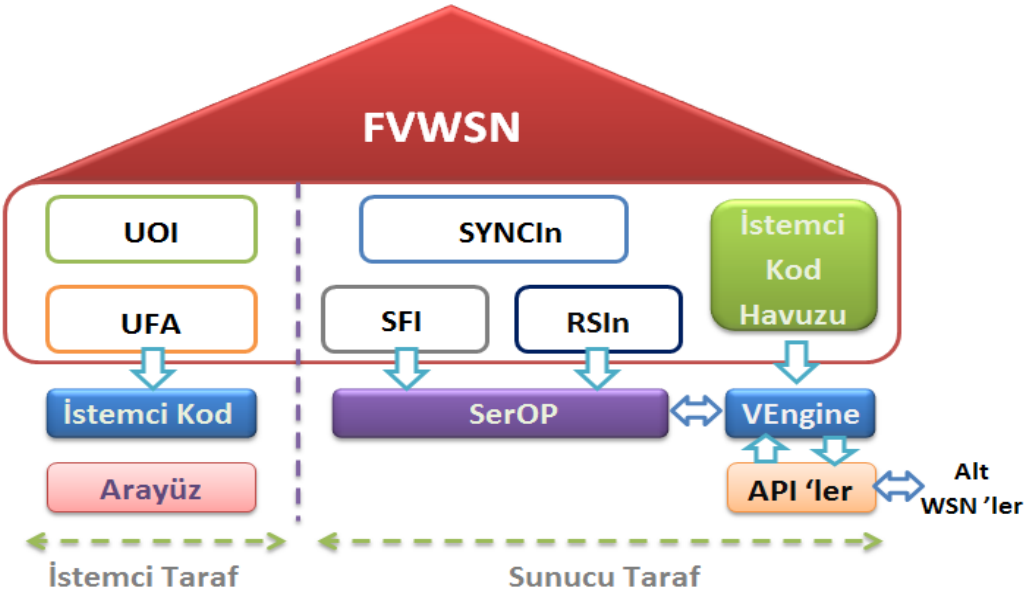
c) Tip-B kullanıcı için VNd oluşturma süreci

Şekil 3.3. VNd mantıksal yapısı ve farklı kullanıcı tipleri için VNd oluşturma süreci

Tip B kullanıcıları kendi VNd 'lerini sahip oldukları güvenlik sınıfından dolayı sistem web hizmetleri üzerinden oluşturabilmektedir. Bu süreçte sağlayıcı sistem, istemci parametrelerini VNd oluşturma modülüne göndererek istemci VNd 'sinin otomatik olarak oluşturulmasını sağlamaktadır. Bu VNd oluşturmada ikinci seviye güvenlik denetimlerine ihtiyaç duyulmamaktadır. Her iki tip VNd arasındaki çalışma farkları Güvenlik ve Kullanıcı Tipleri bölümünde açıklanacaktır. Bu süreçlerin detayları EK-2 de örneklendirilmiştir.

3.1.2. Fırat Sanal WSN Çerçevesi

Kullanıcıya ait algoritma, kod/komutların sunucuda çalışabilir ve ulaşılabilir olması her iki tarafın yazılımsal bir çatı altında toplanmasını gerektirmektedir. Bu özellik, farklı teknolojilere sahip sistemlerin birbirleri ile etkileşmesine önemli bir avantaj sağlamaktadır. Bu tezde önerilen temel yeniliklerden biri, hem sağlayıcı tarafı hem de istemci tarafı aynı yazılım çerçevesi altında etkileştirmek için JAVA 'da geliştirilen "Fırat Sanal WNS çerçevesi (FVWSN)" dir. Önerilen sistemin AYÇ 'si olan FVWSN 'in genel sınıf yapısı Şekil 3.4 'de verilmiştir.

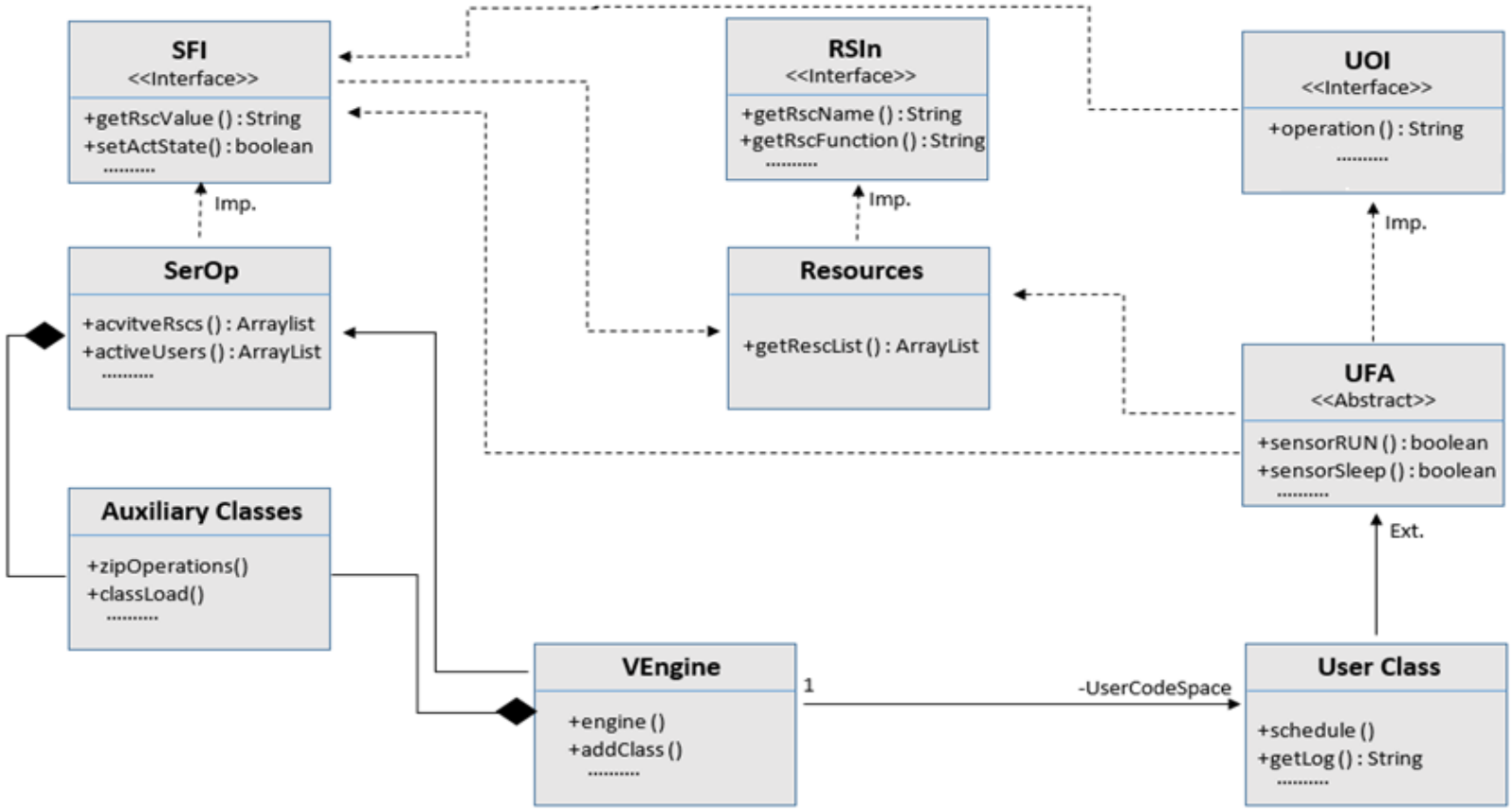


Şekil 3.4. FVWSN yapısı

Genişletilebilir yapıda olan FVWSN, genel olarak iki kısımdan meydana gelmektedir. Bunlar istemci taraf ve sunucu taraftır. Her iki taraf da farklı Interface (Arayüz) ve Abstract (Soyut) sınıflardan oluşmaktadır. Bu sınıflar istemci ve sunucu taraflı fonksiyonları tanımlayan sınıflar olup içerikleri bir birlerinden soyutlanmıştır. Başka bir değişle bir birleri için "black box-kara kutu" yordamlardır. FVWSN 'deki bu ilişkisel yapıyı gösteren UML diyagramı Şekil 3.5 'de gösterilmektedir. Her ne kadar bu AYÇ, JAVA 'da geliştirilmişse de bu model farklı nesnel tabanlı dil teknolojileri ile de rahatlıkla gerçekleştirilebilmektedir.

3.1.2.1 FVWSN İstemci Taraf Yapısı ve Fonksiyonları

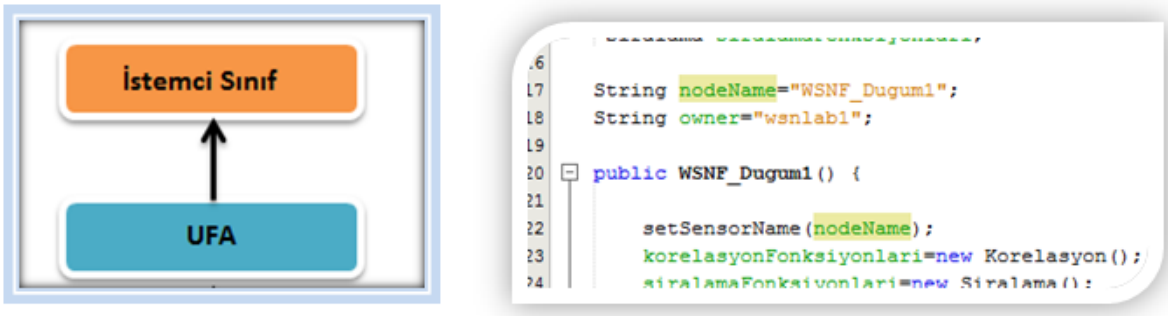
FVWSN 'nin istemci tarafı, iki temel sınıftan meydana gelmektedir. Bunlar "Kullanıcı Operasyon Arayüz Sınıfı (User Operations Interface Class) – KOA/UOI" ve "Kullanıcı Fonksiyonları Soyut Sınıfı (User Functions Abstract Class) – KFS/UFA". UOI temel yapısal fonksiyon tanımlarını deklere ederken, UFA, temel fonksiyonların implementasyonunu sağlamaktadır. İstemci tarafta tanımlı olan fonksiyonların amacı, istemciye sağlayıcı tarafta sunulan temel fonksiyonları kullanarak, istemci kod/komutlarını sağlayıcı sistem için tanımlanabilir yapmak ve istemcinin kullanacağı algoritmaları ilgili sunucu sistem fonksiyonlarıyla gerçekleştirmektir. UFA aynı zamanda Thread sınıfından miras alan bir Abstract sınıf olduğundan VNd 'ler Thread bazlı çalışan varlıklardır.



Şekil 3.5. FVWSN 'nin UML diyagramı

3.1.2.1.1. setSensorName(String isim) fonksiyonu

Bir VNd sağlayıcı tarafta istemci ID ve istemci tarafından belirlenmiş sanal düğüm adı ile ayırt edilmektedir. İstemci ID, istemcinin sağlayıcı sisteme kaydolduğu zaman sağlayıcı tarafından verilen benzersiz kimliktir. VNd ismi ise sağlayıcı tarafından belirlenen kimliktir. Bu kimliğin istemcinin sahip olduğu diğer VNd 'ler içinde benzersiz olması beklenmektir. Aksi durumda sağlayıcı sistem, "Hata 201: İsimlendirme Hatası (Error: 201 Failed Naming)" mesajı gönderecektir. Bir VNd 'nin istemci tarafından kimliklendirilmesi iki şekilde yapılabilir. Birincisi VNd yapılandırıcısında "setSensorName(String isim)" in kullanılmasıdır. İkincisi ise aşağıda açıklanan "getSensorName()" fonksiyonun "override" edilmesi ile isimlendirilmesidir. İlk durumda yapılan isimlendirme sunucu tarafından yeniden isimlendirmeye engel olmamaktadır. Bu yöntem ile isimlendirmeden kaynaklanan hatalar sağlayıcı hata tolerans modülü tarafından düzeltilebilir. Ancak bu durumda aynı istemcilere ait VNd lerin haberleşmesinde veya VNd işlem sonucunda elde edilen verinin yazılmasında sunucu tarafından atanan sensor ismi kullanılacaktır. İkinci yol ile isimlendirmede ise istemci tarafından verilen isim, tüm sağlayıcı operasyonları tarafından değiştirilmeksizin kullanılacaktır. Bu durumda oluşacak hatalar istemcilere bildirilecek ve raporlanacaktır. Bu fonksiyona ait hiyerarşi ve örnek kullanım Şekil 3.6 a ve b ' de verilmiştir.



a) Sınıf hiyerarşisi b) Örnek kodlama

Şekil 3.6. setSensorName(String) fonksiyonu

3.1.2.1.2. getSensorName()

Bu yordam vasıtasıyla sağlayıcı sistem, istemci VNd 'den, atanmış olan sensör isimlendirmesini elde edebilmektedir. İstemci tarafında, VNd isimlendirme için "setSensorName(String)" yordamını kullanmak yerine doğrudan bu yordamın override

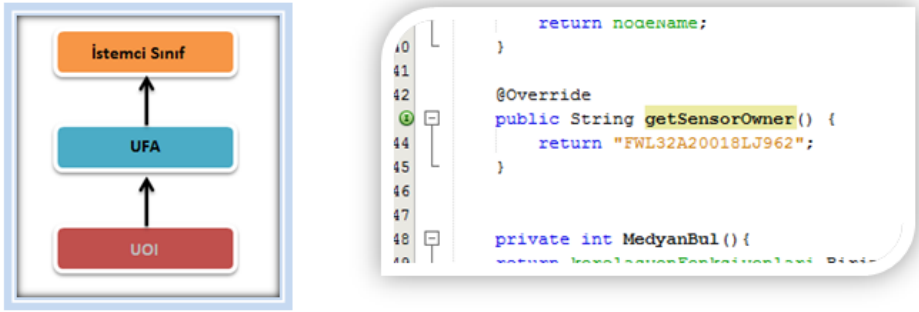
edilmesiyle de söz konusu işlem gerçekleştirilebilir. Ancak bu durum sabittir ve sağlayıcı Hata Tolerans modülü oluşacak hataları sadece rapor eder. Bu yordam da UFA tanımlı bir yordamdır ancak "setSensorName(String)" gibi "final" deklarasyonu ile deklere edilmemiştir. Dönüş değeri String 'dir. FVWSN aksi belirtilmetikçe tüm kimliklendirme yordamlarında String tipini kullanmaktadır. Fonksiyon sınıf hiyerarşisi ve kodlama içerisinde örnek kullanımı Şekil 3.7 a ve b de verilmştir.

a) Sınıf hiyerarşisi b) Örnek kodlama

Şekil 3.7. getSensorName(String) fonksiyonu

3.1.2.1.3. getSensorOwner()

Tüm Tip A ve B istemciler kayıt esnasında sağlayıcı sistem tarafından benzersiz bir ID ile kimliklendirilirler. Bu kimlikler oluşturulacak tüm VNd 'lerde deklere edilmek zorundadır. Sağlayıcı sistem, her istemci için yürütülen işlemleri "getSensorOwner()" ve "getSensorName()" ile elde ettiği özel kimliklendirme ile raporlamaktadır. Bu nedenle bu yordam UOI tanımlıdır. Yordam sınıf hiyerarşisi ve kodlama örneği Şekil 3.8 'de verilmiştir.



a) Sınıf hiyerarşisi b) Örnek kodlama

Şekil 3.8. getSensorOwner fonksiyonu

3.1.2.1.4. SenorRun () / SensorSleep() / SensorStop ()

Bir VNd nin aktif edilmesini, bekletilmesini ve durdurulmasını sağlayan yordamlardır. Her VNd bir thread bazlı varlık olduğundan bu işlevler doğrudan thread fonksiyonları ile ilişkilendirilmiştir. Bunun işlevler için "wait()" ve "notify()" fonksiyonları kullanılmaktadır. Multithread çalışma söz konusu olduğundan bu operasyonlar UFA 'da, "synchronized" kontrolü ile gerçekleştirilmektedir. Bu yordamların Thread sınıfı yordamları ile ilişkilendirilmesinin bir diğer avantajı, thread bazlı oluşacak hatalardan ve problemlerden yönetilmesinin daha kolay olmasıdır. Örneğin bir VNd 'nin canlı olup olmamasının belirlenmesi ve raporlanması (isAlive()), oluşan problemin yürütüm bazlımı yoksa VEngine bazlımı teşhisinde kolaylık sağlamaktadır.

3.1.2.1.5. isRunning () / isBusy() / isSleeping ()

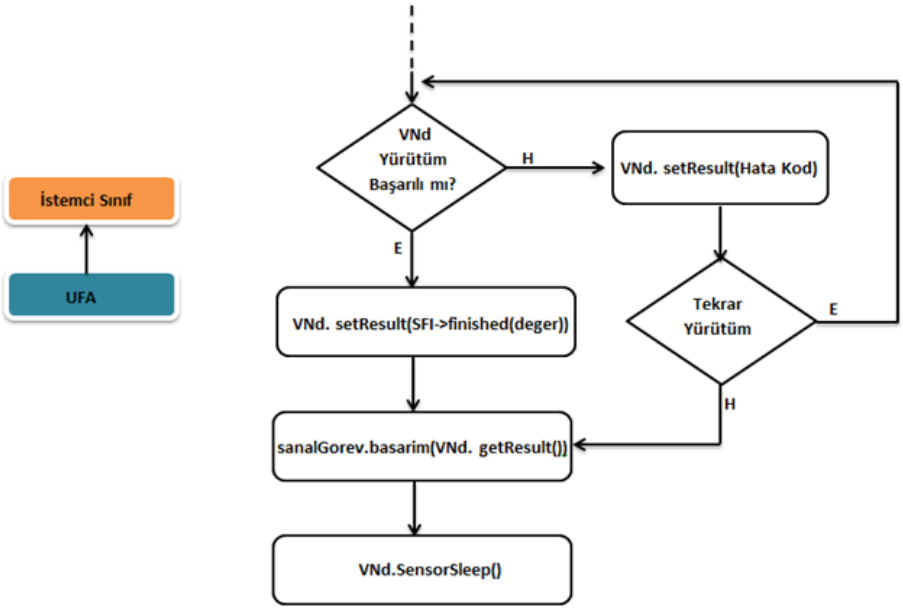
Bir VNd, bir durumdan diğer bir duruma geçiş yaptığında UFA 'da tanımlı ilgili boolean değişken uygun şekilde set edilmektedir. Bu değişkenlerin durumlarının kontrolü sadece kodlama içerisinde değil sağlayıcı sistem modüllerinde de kullanılmaktadır. Aktif VNd 'lerin tespitinde, çalışma zamanı ayarlanmış VNd 'lerin yürütümünde " isRunning()", "isSleeping()" ve "isBusy()" boolean fonksiyonları sıklıkla istemci VNd 'sinden çağrılan yordamlardır. Bunlardan sadece "isBusy()" fonksiyonu sağlayıcı sistem tarafından set edilen bir yordamdır. Sistem performansı açısından bir VNd 'nin yürütümü esnasında istemciden belirli bir süre içerisinde gelecek komutlar değerlendirme dışı bırakılmaktadır. Bu nedenle sağlayıcı sistem bir VNd 'nin ne kadar süre için meşgul göstereceğini optimum şartlar için kendisi belirlemektedir. Hem SenorRun () / SensorSleep() / SensorStop () yordamları hem de bu yordamlara ait sınıf hiyerarşisi ve örnek kod blokları Şekil 3.9 'da verilmiştir.

a) Sınıf hiyerarşisi b) Örnek kodlama

Şekil 3.9. isRunning () / isBusy() / isSleeping ()

3.1.2.1.6. setResult () / getResult()

Bir VNd ‘nin sağlayıcı sistemde yürütümünün tamamlanması ile geri bilgilendirme ve diğer işlemlerin yapılmasına geçiş süreci başlamaktadır. Bu nedenle bir VNd ‘nin, kendisine sunulan şartlar altında başarılı veya başarısız bir yürütüm gerçekleştirdiğinin sistem tarafından anlaşılması gerekmektedir. Bunun için bir “Listener” işlemi yürütülmektedir. Yürütüm sürecini tamamlayan VNd ‘ler SFI arayüz tanımlı “finished()” fonksiyonunu kullanırlar. Bu fonksiyonun VNd için varsayılan sonuç değerini değiştirmemesi durumunda sistem tarafından VNd yürütümünün başarısız olduğu sonuç için “setResult()” üzerinden ilgili hata koduyla etiketlenmektedir. Bir VNd ‘nin hata koduna bağlı olarak tekrardan yürütüm hakkı olabildiğinden aynı süreçler birkaç kez devam edebilmektedir. Başarılı yürütüm ile bu fonksiyon, “finished()” ile belirtilen değer ile set edilecek ve yürütüm başarılı olarak etiketlenecektir. Bu durumda istemcinin sadece “finished()” fonksiyonunu kullanması yeterlidir, “setResult()” kullanıcı VNd ‘sinin yürütümü esnasında sistem tarafından kullanılacaktır. Bir VNd ‘nin her hangi bir anda sonuç değerinin alınması (örneğin sanal görev sınıf özelliklerinin set edilmesi vb.) “getResult()” yordamı ile kotarılmaktadır. Şekil 3.10 ‘da bu fonksiyonlara ait sınıf hiyerarşisi ve çalışma yöntemini gösteren akış diyagramı verilmiştir.



Şekil 3.10. setResult() / getResult() hiyerarşik sınıf yapısı ve çalışma yöntemi akış diyagramı

3.1.2.1.7. operation(SFI, String)

FVWSN ‘nin istemci tarafındaki en önemli yordamıdır. İstemciler bu yordamı implement ederek kendi ağlarında kullandıkları veya belirledikleri komut/sorgu veya algoritmalarını VNd üzerinde tanımlayabilirler. Bu yordam sağlayıcı taraflı SFI (Sunucu Fonksiyonları Arayüzü) sınıfından bir parametre almaktadır. Bu parametre ile sağlayıcı sistem tarafından VNd ‘ler için sağlanan servislere ulaşılabilmektedir. Sağlayıcı sistem tarafında ilgili VNd için belirlenmiş kaynaklara ulaşım, aktuatör kontrol, veri kaydetme gibi tüm temel işlevler istemci tarafından bu yordam içinde gerçekleştirilmektedir. İstemci tarafından tanımlanan komutlar bir ArrayList gibi bir veri yapısı içerisinde tutulabilmektedir. Programlama kolaylığı ve genellemesi açısından istemci komutları String parametre olarak bu yordama iletilmektedir. İstemci taraflı kod/komut ve sorgular istemci tarafından “operation(SFI, String)” yordamı içerisinde çözümlenir. Çözümlenen komut/sorgu sonucunda kullanılacak olan sunucu taraflı kaynaklar SFI arayüzü değişkeni ile çağrılmaktadır. Daha sonra açıklanacağı üzere SFI arayüzü ile spesifik sensör bilgilerine ulaşılabilmekte ve farklı komutlar ile farklı operasyonlar bir VNd üzerinden gerçekleştirilebilmektedir. Bu özellikle FVWSN ‘nin istemci-sunucu arası veri trafiğini azaltmada ön plana çıkan bir özelliğidir. Örnek olarak bir sensörün verileri sistem tarafından tutulduğundan o sensöre ait verilerin belli bir bölümüne (misal son 15 okuma gibi) analiz maksatlı olarak SFI parametresi ile kolaylıkla ulaşılabilmektedir. Üstelik önceki verilerin istemci tarafından evvelinden alınmasına gerek kalmadan. Bu durum eğer bir klasik WSN bulut sağlayıcı ile gerçekleştirilmiş olsaydı ilgili sensör verilerin sürekli stream şeklinde önceden istemci tarafından alınması ve yine istemci tarafta değerlendirilmesi gerekmekteydi. Bu hem maliyet hem de zaman gerektiren bir süreçtir. Öte yandan önerilen sistem ile istemciler, bir kaynak ile ilgili sunulan ön hizmetlere, bir maliyet ve zaman kaybı olmadan ulaşabilmekte ve değerlendirmelerini sunucu tarafta yapabilmektedir. FVWSN ‘nin sağladığı avantaj aşağıdaki senaryo ile daha iyi açıklanabilir.

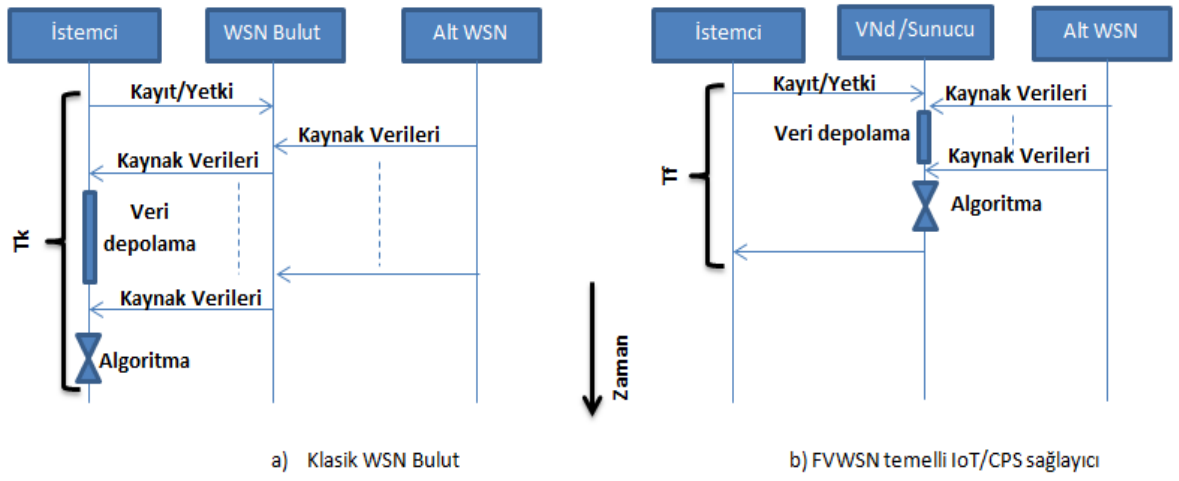
```
VNd 'de İstemci Algoritmaları Gerçeklemesi ve Komut/Sorgu Geçişkenliği
1 public class sanalDugum implements UFA
2 ArrayList <ZigBee Kod Tanım Sınıfı zkt>
3 sahibi ← İstemci ID
4 dugum ← Dugum ID
5 k← En son kaç okuma isteniyor
6 Yapılandırıcı sanalDugum()
7     zkt.ProfileID ← "2323", zkt.ClusterID← "0101",zkt.ClusterID← "AA"
8     setSensorName(dugum)
8 Override getSensorOwner()
9     return sahibi
10 Override operation (SFI sf, String komut)
11    profil_ID ← StringParsing(komut), cluster_ID← StringParsing(komut), command_ID←StringParsing(komut)
12    if (zkt.ProfileID =profil_ID && zkt.ClusterID= cluster_ID && zkt.ClusterID =command_ID)
13          ArrayList<Double> nemDegerleri = sf.getSensorRecords("Nem Kaynak ID", k)
14          ArrayList<Double> sıcaklıkDegerleri = sf.getSensorRecords("Sıcaklık Kaynak ID", k)
15          concordant, discordant←0
15          for i=0 : k-1
16                for j=i+1 : k
17                if (nemDegerleri(i)-nemDegerleri(j) ve sıcaklıkDegerleri(i)-sıcaklıkDegerleri(j) aynı işaretli )
18                      concordant ++
19                else
20                      discordant ++
21
22          KTau Faktör←  Böl ((concordant-discordant), k(k-1)/2)
23          sf.finished(KTau Faktör)
```

Şekil 3.11. Bir VNd 'de örnek istemci algoritması ve kod/sorgu tanımlaması

Senaryo; Bir tarım tesisinde toprak nem (Watermark soil moisture vb.) ve toprak altı sıcaklık seviyesi (PT1000 temperature sensor vb.) sensörlerinden belli bir çalışma periyodunda eş zamanlı olarak veri akışı vardır. Toprak nem sensörü ve toprak sıcaklık sensörlerinin sulanacak bitki/ağacın sulama yarıçapı içerisinde farklı noktalara yerleştirilmiştir. Bu durumda toprak nem seviyesi ile toprak sıcaklığı arasındaki ilişki "Kendall's Tau Faktörü" ile hesaplanabilir. Bir "Private Profile" kullanan ZigBee ağa sahip farklı lokasyondaki istemci, "Profile ID=2323", "Cluster ID=0101" ve "Command ID=AA" ile "Kendall's Tau Faktörü" algoritmasını ve değerlendirmesini son 15 kaynak değerleri için gerçekleştirmektedir. Bunun için sunucu sistemdeki VNd 'sine ait pseudo kod örneği Şekil 3.11 'deki gibi olacaktır. Bu şekilde istemci sadece kaynak bilgilerini kullanarak alt yapısı hakkında hiçbir şey bilmediği başka bir WSN sistemden elde ettiği bilgilerle değerlendirme yapabilmektedir. Dahası bunu yaparken yine kendi tanımladığı veya kendi WSN ağında hali hazırda kullanmış olduğu komutaları kullanmaktadır. Ayrıca bu komutlar doğrudan

istemciden gönderilebildiği gibi istemci WSN 'nindeki herhangi bir düğümden de gönderilebilmektedir. Bunlara ek olarak sonuçta elde edilen veri bir analiz verisi olup diğer istemcilerin faydalanacağı şekilde anlamlandırılarak sağlayıcıda depolanabilir veya pub/sub ara katman yoluyla ilgili kişilere servis edilebilmektedir. Bununla ilgili detaylar büyük veri işlemleri bölümünde sunulacaktır. VNd 'lerin sağladığı avantajlar önerilen sistemin en özgün özellikleri arasındadırlar.

Şekil 3.12 a ve b, önerilen sistemin klasik WSN bulut sağlayıcılara göre sağladığı avantajları daha iyi ifade etmektedir. Aynı senaryo klasik WSN bulut sistemler ile gerçekleştirilmiş olsaydı istemciler, ilgili sensör kaynakları için veri talep edecekler ve bu verileri kendi ortamlarında saklamak durumunda kalacaktır. Dahası gerçekleştirilecek olan algoritma yine istemci kaynaklar üzerinde gerçekleştirilecektir. Elde edilen veriler ile ilgilenen başka kişilerin/kuruluşların olup olmadığı veya bunlara nasıl servis edileceği ayrı bir soru olacaktır. Eğer istemci, sağlayıcı tarafından gönderilen veri miktarına göre fiyatlandırılıyorsa bu zaman kaybının yanında fazladan maliyet artışına da sebep olacaktır. Bunlara ek olarak klasik WSN bulut sağlayıcı sistemler genellikle web tabanlı filtreleme hizmetleri sağladığından kısıtlı sayıda filtreleme işlemi gerçekleştirilebilir. Oysa VNd 'ler üzerinde gerçekleştirilecek algoritmalar ile istemci tanımlı farklı filtrelemeler de mümkün olabilecektir. Dahası klasik WSN bulut sistemlerde (SOAP, RESTFul tabanlı internet servisleri sunanlar hariç), istemci taraflı cihazların belli komutlarla/sorgularla sağlayıcı sistem ile etkileşmesi zordur ve çoğu zaman sistem tarafından sağlanmayan bir durumdur.



Şekil 3.12. Önerilen sistemin klasik bulut WSN sağlayıcılara göre zaman ve maliyet kazancı

Şekilden de anlaşılacağı üzere hem istemci- sağlayıcı arasındaki iletişim gecikmesi hem de istemci tarafta gerçekleştirilen diğer işlemlerden (veri depolama, algoritma yürütümü) dolayı zaman karşılaştırması $\boldsymbol{T_k} > \boldsymbol{T_f}$ olacaktır.

3.1.2.2. FVWSN Sağlayıcı Taraf Yapısı ve Fonksiyonları

Sağlayıcı taraf fonksiyonları istemciler için her zaman "black box" olup implementasyonları sağlayıcı tarafta gerçekleştirilmiştir. Sağlayıcı sistem sanallaştırma motoru VEngine ve diğer bileşenlerle ortak çalışabilmektedir. En temel iki interface sınıfı SFI ve RSIn 'dir. Bunlardan SFI istemci kodu ile sağlayıcı sistem arasındaki ilişkiyi sağlayan yordamları deklere ederken RSIn sistemde tanımlı kaynakların nesnel olarak yürütümde kullanılması için gerekli yordamları tanımlamaktadır.

3.1.2.2.1. setResourceID() / getResourceID()

Sağlayıcı tarafta kaynaklar için oluşturulan nesnelerin gerekli bilgiyi sağlayıcıda kayıtlı olan "Kaynaklar" isimli bir XML dosyasından okumaktadırlar. Bu dosya RDD dosyalarının genişletilmiş ve bir araya getirilmiş halidir. Bir kaynağa ait ayırt edici özellik olan 10 karakterli kaynak kimliği (ID) bu fonksiyonlar tarafından ayarlanır ve okunur. Kaynak ID 'nin ilk üç karakteri alt WSN kodunu geri kalan karakterler ise kaynak numarasını vermektedir. Oluşturulan kaynak sınıfı, RSIn interface sınıfının implement etmektedir.

3.1.2.2.2. setResourceName() / getResourceName()

Kaynaklar sahip oldukları ID 'lerin yanında alt WSN sağlayıcılar tarafından sağlanan ayrı bir isimlendirmede kullanabilirler. Ayrıca sağlayıcı, sistem kullanılan kaynağı tam ürün adını bu alana set edebilir. Bu özelliğin set edilmesi aynı tip kaynakların ayırt edilmesinde ve detaylı arama işlemlerinde faydalı olmaktadır ancak bir kullanılma zorunluluğu yoktur, null değer döndürebilir. Bu fonksiyon RSIn arayüz deklarasyonudur.

3.1.2.2.3. setResourceFunction() / getResourceFunction()

Alt WSN 'lerdeki kaynaklar daha önce de belirtildiği gibi, sensör, aktuatör ve tamamlayıcı olmak üzere üç sınıfta toplanmaktadır. Bir kaynağın tipi, setResourceFunction() / getResourceFunction() yordamları ile set edilir ve elde edilir. Kaynak tiplerinin belirleyici özelliği kendisini kaynak filtreleme de göstermektedir. Normal VEngine ve SerOp işlemlerinde etkin bir rolü bulunmamaktadır. Bu yordamlar RSIn tanımlıdır.

3.1.2.2.4. setOwner() / getOwner()

RSIn tanımlı bu yordamlar, kaynağın sahibi olan alt WSN sağlayıcı kimliğini vermektedir. Her alt WSN sisteme kaydolurken benzersiz bir "owner ID" ye sahip olmaktadır. Bu kimlik kaynakların ayırt edici ikincil parametrelerinden de olsa özellikle fiyatlandırma ve raporlamalarda sıklıkla kullanılan yordamlardır. Bir owner ID, üç karakterden oluşur ve kaynak ID 'nin oluşturulmasında da kullanılır.

3.1.2.2.5. setAddress() / getAdress()

Bu yordamlar alt WSN 'nin her şeyi ile sağlayıcı sisteme ait olan kaynaklar için geçerlidir. Sağlayıcı sistem altında hizmet veren alt ağlara ait düğümler üzerinde çalışan kaynakları kontrol etmede kullanılır. Bu adresler düğüm adreslerini göstermekte olup ilgili alt WSN ağı API 'sinde kullanılmak için çağrılmaktadır. Farklı WSN 'lere ait API lerin kullanılmasında sadece kaynak ID kullanılmaktadır. Gerekli dönüşüm alt WSN ağlarda gerçekleştirilebilir. Ancak bu durum sağlayıcı ve alt WSN sahibinin anlaşması durumunda farklı işletilebilir. Anlaşma sağlandığı durumda doğrudan kaynağın takılı olduğu düğümün adresi kullanılabilir. RSIn tanımlı bir diğer yordamdır.

3.1.2.2.6. setAPI() / getAPI()

İlgili kaynağın bulunduğu alt WSN sahibinin sağladığı API 'nin yolunu gösteren yordamlardır. Bir alt WSN 'nin kaynaklarına ulaşmak için uzaktan çağrım yöntemleri kullanılması durumda bu yordamlara ihtiyaç duyulmaz. Daha çok sisteme doğrudan entegre olan alt WSN 'lerin kullanılması durumunda bu yordamlar kullanılır.

3.1.2.2.5.7. setPin() / getPin()

Bu yordamlarda tıpkı setAPI() /getAPI() yordamları gibi sağlayıcı tarafından oluşturulan alt WSN ağlara ait kaynakların kullanılması durumunda çağrılmaktadır. Daha çok aktuatör kaynakların set edilmesinde kullanılırlar. Bazı düğüm teknolojilerinin sağladığı API lerde pin numarası üzerinden etkileşim gerçekleştiğinden bu yordamlar kullanılacak ağ ve düğüm teknolojilerine bağlı olarak tercih edilirler. Bu fonksiyonlarda RSIn ara yüzünde deklere edilmiş yordamlar olup kullanılmamaları durumunda null değer döndürmektedirler.

3.1.2.2.5.8. getResourceValue() /setActuator(String,boolean)

SFI tanımlı bu fonksiyonlar kaynak değerlerinin elde edilmesini ve aktuatör tipi kaynakların set edilmesini sağlamaktadır. String dönen değerler gerekli dönüştürme işlemine tabi tutulması istemcinin sorumluluğundadır.

3.1.2.2.5.9. getResourceRecords(String,int) / getResourceRecordsAdv(String,int) / getHighest(String) / getLowest(String) / getHighDuration() / getLowDuration()

Bir kaynağa ait bazı arşiv verilerinin sistemden alınmasını sağlayan kullanışlı fonksiyonlardır. Bunlardan "getResourceRecords(String,int)" sadece ilgili kaynağa ait "k" adet son okunan veriyi ArrayList<String> olarak döndürmektedir. Benzer yordam "getResourceRecordsAdv()" ise okuma zamanı ile beraber bu verileri sağlamaktadır. Bu durumda parsing işleminin yapılabilmesi için bilgiler arası "!" karakteri ile ayrılmıştır. Tahmin edileceği üzere getHighest() / getLowest() yordamları arşiv verileri içerisinde okunan en yüksek ve en düşük değeri elde etmek için kullanılmaktadır. Son olarak "getHighDuration()" ve "getLowDuration()" yordamları ise genellikle aktuatör kaynakların açık/true ve kapalı/false durumda kalma sürelerini öğrenmek için kullanılmaktadırlar.

3.1.2.2.5.10. isResourceActive() / isResourceValid()

Bu boolean fonksiyonlar kodlama içerisinde kaynakların sağlamlığın kontrol etmek ve gerekiyorsa raporlama yapmak için kullanılmaktadır. Bu yordamlar bir "kalp atışı " sinyali gönderirler ve cevabı beklerler. Bir VNd 'ye ayrılan yürütüm zamanının kısıtlı olmasından dolayı bu fonksiyonların istemciler tarafından kodlamada kullanılması tavsiye edilmemektedir. Genellikle sistem tarafında çalıştırılan SFI tanımlı fonksiyonlardır.

3.1.2.2.5.11. writeToResource(String,String) / writeToSystem(String)

VNd yürütümleri esnasında elde edilen sonuçların yazılabilir bir kaynak üzerine veya sistem üzerindeki veri depolama sistemine yazılmasını "writeToResource(String,String)" ve "writeToSystem(String)" sağlar. Son yordam "publish(String)" ise verilerin diğer istemciler için paylaşılabilir olmasını sağlar. Bu yordamlar arka planda bazı SerOp ve VEngine işlevlerini kullanmaktadır.

3.1.2.2.5.12. finished(String)

Bir VNd operasyonunun sonlandırıldığını bildiren yordamdır. İstemci kodunda kullanılır. Bu yordamın kullanılmasından sonra sistem VNd 'yi tekrardan uyku moduna alır. Bu yordam da SFI deklarasyonudur.

3.1.2.2.5.13. VTask (Sanal Görev) Sınıfı

Sağlayıcı tarafta yürütülen görevleri tanımlayan sınıftır. Bir istemci talebi üzerine bu sınıfa ait bir instance oluşturulur ve istemci, kaynak, öncelik derecesi ve yürütümle ilgili tüm detay bilgileri barındırır. İstemciye cevap verisi döndürüldüğünde ise sistemden kaldırılır. VEngine 'den raporlama işlemlerine kadar her operasyonda kullanılan bir sınıftır. Bu sınıfın işlev yapısını gösteren Tablo 3.1 aşağıda sunulmuştur.

Tablo 3.1. Sanal Görev sınıfı yapısı

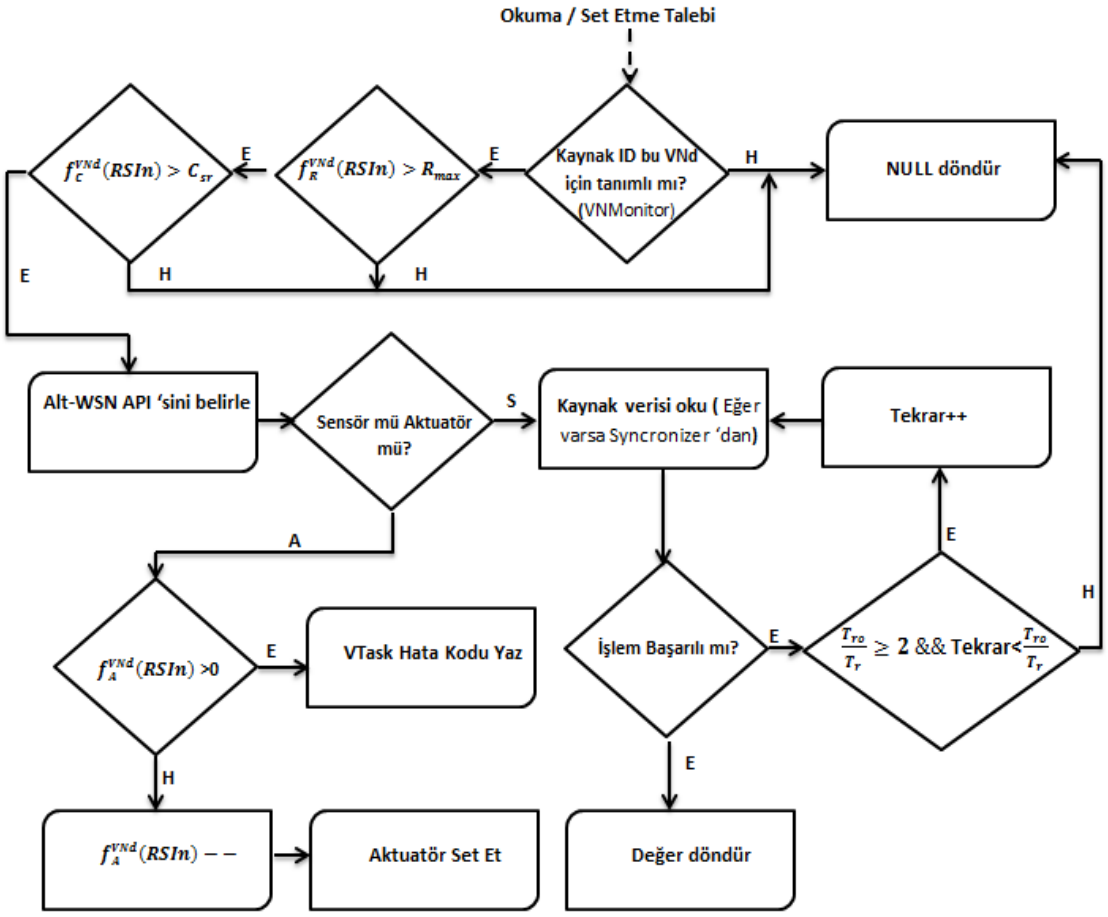
Fonksiyon	Parametre	Dönüş Tipi	Açıklama
setFinished ()	Boolean	void	Sanal görevin bitmesi durumunda set edilir
isFinished()	-	Boolean	Görevin bitip bitmediğini sorgular
setVTaskID()	String	Boolean	Verilen benzersiz kimliği set eder
getVTaskID()	String	void	Görev kimliğini verir
setOwner()	String	Void	VNd sahibinin kimliğini set eder
getOwner()	-	String	VNd sahibinin kimliğini verir
setNode()	String	Boolean	VNd kimliğini set eder
getNode()	-	String	VNd kimliğini verir
setPriorty()	String	void	Görev önceliğini ayarlar.(VH,H,M,L,VL)
getPriority()	-	String	Görev önceliğini verir
setPacketNo()	integer	void	İlgili VNd için yapılan talep sayısını set eder
getPacketNo()	-	integer	VNd için yapılan talep sayısını verir
setHowLongLasted()	Double	void	Sanal görev toplam yürütüm süresini set eder
getHowLongLasted()	-	Double	Sanal görev toplam yürütüm süresini verir
setResources()	ArrayList<RSIn>	void	Kullanılan kaynakları set eder
getResources()	-	ArrayList<RSIn>	VNd 'nin kullandığı kaynakları verir
setCommand()	String	Boolean	M2M arakatman yazılımı vasıtasıyla alınan istemci komut veya sorgusunu set eder
getCommand()	-	String	M2M arakatman yazılımı vasıtasıyla alınan istemci komut veya sorgusunu verir

setResult()	String	-	VTask sonuç bilgisinin set edilmesini sağlar
getResult()	-	String	VTask yürütüm sonucu
setRequestTime()	TarihZamanVeri	void	İstemci talep zamanı
getRequestTime()		TarihZamanVeri *	İstemci talep zamanı
setResponseTime()	TarihZamanVeri	void	İstemciye cevap zamanını set eder
getResponseTime()	void	TarihZamanVeri *	Cevap zamanını verir

* VH=En yüksek öncelik H=Yüksek M=Orta L=Düşük VL=En düşük

3.1.2.2.5.14. SerOP Sınıfı

SFI ara yüzünü sağlayıcı tarafta implement eden sınıf olup sistemin en önemli bileşenlerindendir. İstemci VNd 'si "operation()" yordamında parametre olarak kullanılan SFI referansının sağladığı işlevleri sağlayıcı tarafta gerçekleştirmektedir. SFI implementasyonunun dışında sistem yönetimine ait bazı yardımcı fonksiyonları da içermektedir (Sanal Göreve erişim, VEngine yordamlarına erişim vb.). Bu sınıfa ait instance, VEngine içerisinde VNd yürütümü için çağrılan OperationRunner 'a gönderilir. SerOp aynı zamanda ilgili kaynaklara ulaşılan API 'lerle de etkileşimdedir. Sensör kaynakların okunması veya aktuatör kaynakların set edilmesi gibi işlemler SerOP üzerinden gerçekleşmektedir. Bu işlemler SFI deklarasyonu olan "getResourceValue()" ve "setActuator(String,boolean)" yordamlarının implement edilmesi ile yapılmaktadır. Bu yordamlarda, kaynak-API çözümlemesi gerçekleştirilir ve ilgili alt WSN gateway 'ine talebin iletilmesi sağlanır. SerOP yapılandırıcısına aynı zamanda kullanılan semaphore değişkeni de parametre olarak gönderilmektedir. Bu değişken çoklu iş parçacığı işlemlerinde eş zamanlı ulaşımları engellemek için kullanılmaktadır. Alt WSN sistemde ki her bir kaynak için maksimum bir okuma/set-etme süresi Bölüm 4.4 'de belirtildiği gibi $(\boldsymbol{T_r})$ sistemde tanımlıdır. $\boldsymbol{T_r}$, alt-WSN ile sağlayıcı sistem arasındaki gecikmeye, alt-WSN çalışma modeline ve işlem yüküne bağlıdır. Bazı durumlarda sistem optimum $\boldsymbol{T_r}$, değeri atayabilir ($\boldsymbol{T_{ro}}$). $\boldsymbol{T_{ro}}$, belli bir eşik değerinin altında bulunulmaz ($\boldsymbol{T_{ro}} > \boldsymbol{T_r} \geq \varepsilon_{\boldsymbol{r}}$). $\boldsymbol{T_{ro}}$ değerinin $\boldsymbol{T_r}$ değerinin iki katından fazla olması durumunda SerOp aynı işlemi tekrardan deneyebilir. $\boldsymbol{T_{ro}} veya\, \boldsymbol{T_r}$ değerlerinin aşılması durumunda OperationRunner gelecek her türlü bilgiyi yok sayacak ve tekrardan okuma izni kadar kaynak verisi yeniden istenecektir.



Şekil 3.13. SerOP çalışma akış diyagramı

Güvenlik ve performans bakımından bir VNd belli sayıda kaynağı yönetebilmektedir. Bu değer sistem ön tanımlıdır $\boldsymbol{R_{max}}$ ile ifade edilmektedir. Buna ek olarak bir kaynak bir yürütümde belli sayıda ($\boldsymbol{C_{sr}}$) okunabilmektedir. Bu değerin nasıl elde edildiği Bölüm 4.4 'de Denklem 4.23 'de verilmiştir. Bu değerlerin aşılması durumunda VNd yürütümü kesilecek ve VTask 'e ilgili hata kodu eklenecektir. Yürütüm esnasında $\boldsymbol{R_{max}}$ değeri $\boldsymbol{f_R^{VNd}(RSIn)}$ ile $\boldsymbol{C_{sr}}$ değeri ise $\boldsymbol{f_C^{VNd}(RSIn)}$ fonksiyonları ile elde edilmektedir. Öte yandan bir aktuatör kaynağın çoklu kullanıcılar tarafından paylaşılması dikkat edilmesi gereken bir durumdur. Kontrolsüz aktuatör işlemleri güvenlik, sağlık gibi uygulamalarda ciddi sonuçlara yol açabilir. Bu amaçla bu biriminde, eşzamanlı çalışma kontrol yönetimlerinden olan semaphore kullanımına benzer bir mekanizma kullanılmaktadır. Bu mekanizmada her bir aktuatör, "key" olarak adlandırılan bir tam sayı değere sahiptir ve çalışma esnasında $\boldsymbol{f_A^{VNd}(RSIn)}$ ile elde edilebilmektedir. Bu değer alt WSN ve sistem tarafından aktuatör özelliğine ve önemine göre 1 veya daha büyük seçilebilir. Yetkili bir istemci tarafından kullanımı talep edilen aktuatör için bu değer bir

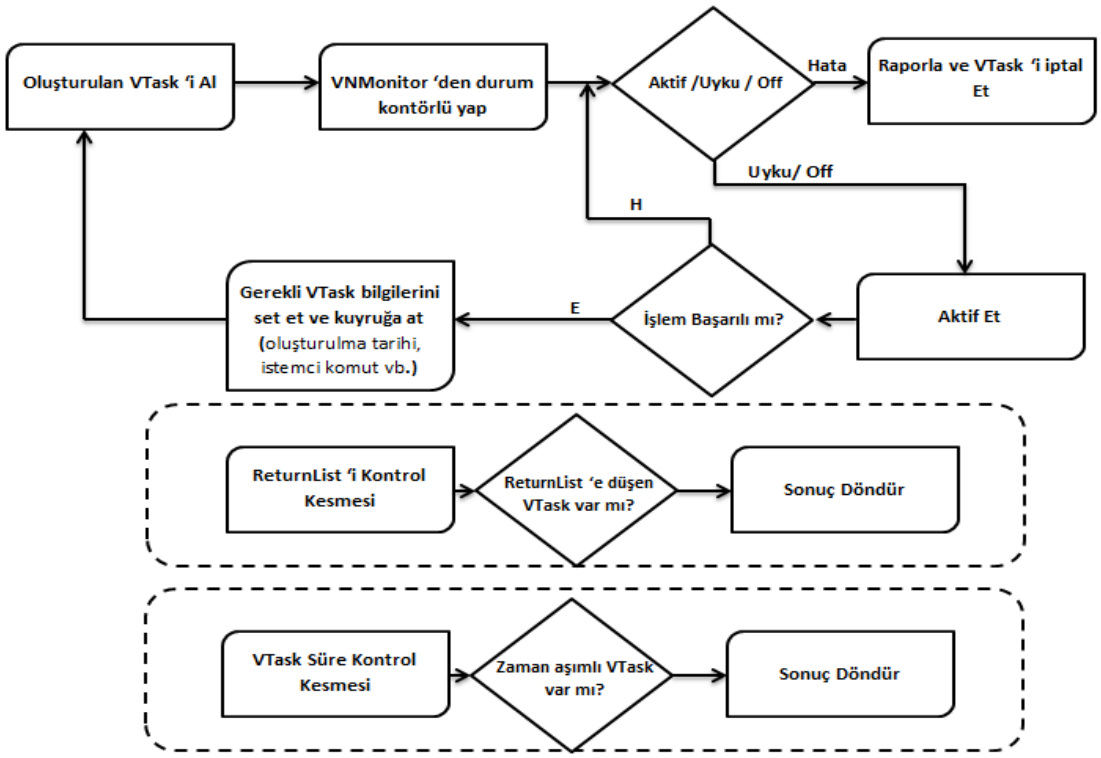
azaltılır. Değeri “0” olan aktuatöre, maksimum öncelikli sistem uygulamaları hariç diğer hiçbir istemci müdahale edemez. Bu yöntem ile aktuatör kullanımlarında karasızlık ortadan kaldırılmış olmaktadır. SerOp modülüne ait çalışma akış diyagramı Şekil 3.13 ‘deki gibidir. Tablo 3.2 ‘de SerOP sınıfının SFI implementasyonlarının dışında kalan fonksiyonları verilmiştir.

Tablo 3.2. SFI fonksiyonları dışında kalan SerOP fonksiyonları

Fonksiyon	Parametre	Dönüş Tipi	Açıklama
setVTask()	SanalGorev	void	VNd yürütümü için atanan sanal görevi set eder
getVTask()	-	SanalGorev	VNd sanal görevini verir
getSynchronizingList()	-	ArrayList	Senkronlama zaman süresi içerisinde kaynaklara ait en son okunan değer ve VNd leri verir
activeResources()	-	ArrayList<RSIn>	Aktif Kaynakların listesini verir
activeUsers()	-	ArrayList<String>	Aktif kullanıcıları verir

3.1.2.2.5.15. AddAndFollow Sınıfı

İstemciden M2M ara katman yazılımı ile alınan talep doğrultusunda ilk önce bir görev oluşturucu yardamı ile sanal bir görev (VTask) oluşturulmaktadır. Oluşturulan bu görev AddAndFollow sınıfından oluşturulan instance modülüne gönderilir. Bu modül gerekli verileri set ederek görev kuyruğuna (TaskQueue) göndermektedir. Gönderilen bir VTask ‘in yürütümü tamamlandığında dönen görev listesinden (ReturnList) yine bu modül tarafından kaldırılarak istemciye gönderilir. Sistem tarafından belirlenmiş ön tanımlı bir süre içerisinde ReturnList ‘e düşmeyen VTask ‘ler başarısız olarak etiketlenecek ve tekrardan gönderim yapılması için istemciye döndürülecektir. Bu işlemleri yaparken ilgili sanal göreve ait set edilmesi gereken fonksiyonlar bu modül fonksiyonları içerisinde çağrılmaktadır. Diğer bir görevi ise oluşturulan VTask ‘in çalıştıracağı VNd ‘nin durumunun kontrolünü VNMonitor ‘den sağlamaktır. Bir VNd nin aktif edilmesi veya bir sorunla karşılaşması durumunda raporlanması bu modül tarafından yürütülmektedir. AddAndFollw sınıfına ait akış diyagramı Şekil 3.14 ‘de ve fonksiyonları ise Tablo 3.3 ‘de sunulmuştur.



Şekil 3.14. AddAndFollow çalışma akış diyagramı

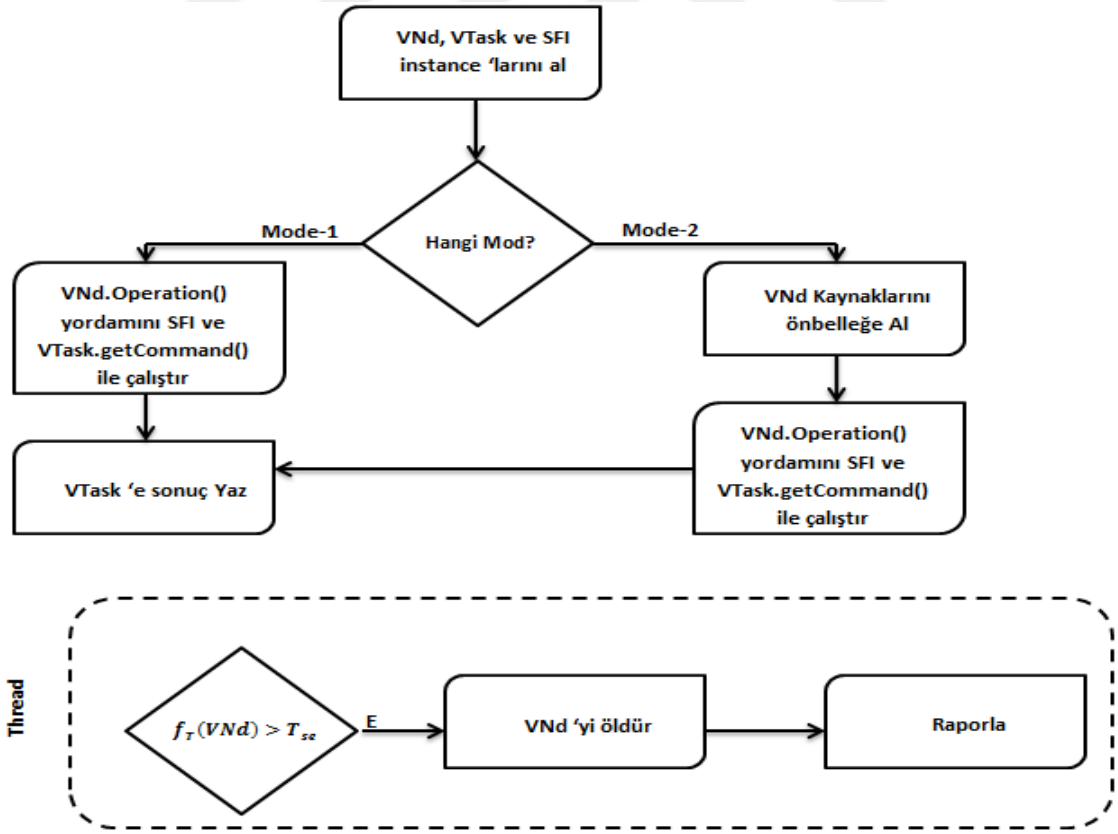
Tablo 3.3. AddAndFollow sınıfı fonksiyonları

Fonksiyon	Parametre	Dönüş Tipi	Açıklama
addTask()	VTask	Boolean	Oluşturlan VTask 'i TaskQueue 'ya ekler
removeTask()	VTask	Boolean	Belirtilen VTask 'i ReturnList 'den çıkartır ve sonucu istemciye gönderir
activateVNd()	String	String	VTask 'in yürütüleceği VNd 'yi aktif eder. Dönen kod tipi bu işlemin başarılı veya hatalı olduğunu gösterir
declineTask()	String	Boolean	Belirtilen VNd 'lerin VTask 'lerini TaskQueue 'ya göndermez.

3.1.2.2.5.16. OperationRunner Sınıfı

Sağlayıcı sistemde VNd 'ler, OperationRunner sınıfından oluşturulan instance tarafından yürütülmektedir. Bu modül kendisine gelen sanal görev bilgilerine göre VNd instance 'larını çalıştırmaktadır. Bu modülün en önemli fonksiyonu "execute(String,SFI)" 'dur. Bu fonksiyon VNd ID ve gönderilen SerOp instance 'larını söz konusu VNd 'nin "operation()" yordamına aktarmaktadır. Her bir VNd 'nin sistem tarafından önceden belirlenen ön tanımlı bir

maksimum yürütüm süresine (T_{se}) sahiptir. Bölüm 4.4 'de verilen Denklem 4.22 'deki şartın aşılması durumunda, VNd 'yürütümünün başarısız olduğu ve bu başarısızlığının nedeni ilgili VTask 'e işlenerek ReturnList modülüne gönderilmektedir. Bu modül iki yürütüm modunda çalışabilmektedir. Bunlardan ilki düşük işlem yükü durumunda her bir VNd 'ye ait kaynakların, "operation()" fonksiyonunun yürütümü esnasında okunması ve ayarlanmasını sağlayan "Mode-1" dir. İkinci mod ise "Mode-2" olarak adlandırılır ve yüksek işlem yükü olması durumunda VEngine tarafından otomatik olarak aktif edilir. Bu modda bir VNd yürütümü gerçekleştirilmeden önce kullanacağı kaynaklara ait veriler sistem performansı açısından önbelleklenir. SFI 'ın "getResourceValue()" işlevi çalıştırıldığında bu değer önbellekten elde edilir. OperationRunner modülü aynı zamanda VNMonitor modülüyle de etkileşim halindedir. İstemci VNd 'leri ile ilgili bir problem olması durumunda, bu modül, ilgili VTask 'i söz konusu problemi gösteren hata kodu ile set ederek ReturnList 'e gönderir. Bu sınıfa ait çalışma akış diyagramı Şekil 3.15 'de ve fonksiyon tanımları Tablo 3.4 'de verilmiştir.



Şekil 3.15. OperationRunner çalışma akış diyagramı

Tablo 3.4. OperationRunner sınıfı fonksiyonları

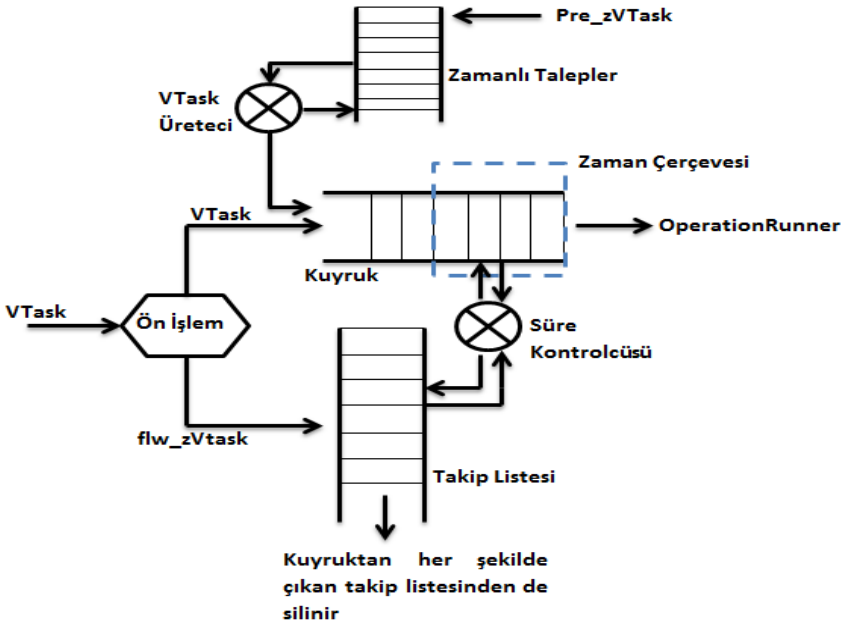
Fonksiyon	Parametre	Dönüş Tipi	Açıklama
getTask()	-	void	TaskQueue 'dan yeni bir VTask talep eder
execute()	String,SFI	void	VNd yürütüm yordamı
sendToReturnList()	VTask	void	Yürütümü tamamlanan VTask 'i dönen görevler modülüne gönderir
getVNdStatus()	String	void	VNMonitorden ilgili VNd durum bilgisi alır

3.1.2.2.5.17. TaskQueue (Sanal Görev Kuyruğu) Sınıfı

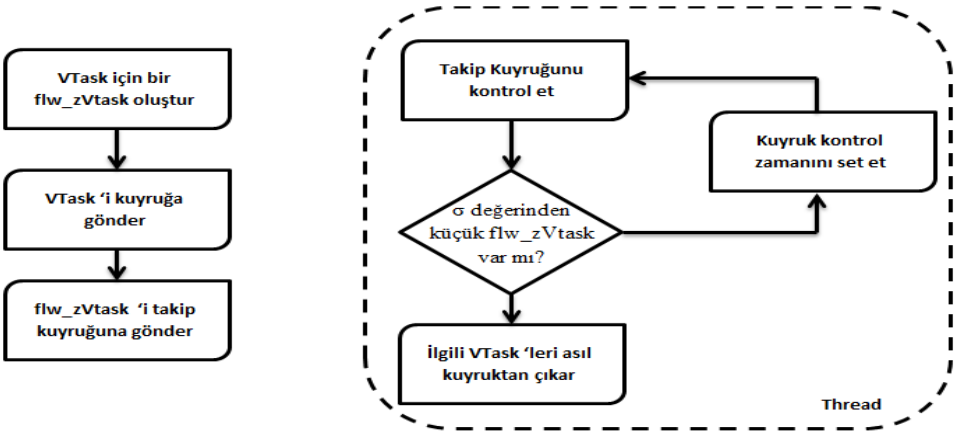
Talep üzerine oluşturulmuş VTask 'ler FIFO düzeni ile işletilmektedir. Bu nedenle VTask'ler çalışma modeli Bölüm 4.4 'de Denklem 4.26-4.29 'a göre çalışan bir kuyruk veri yapısında tutulmaktadırlar. TaskQueue sınıfı içerisinde "Queue<Vtask>" veri yapısı ve diğer yardımcı sınıflar bulunmaktadır. TaskQueue 'da oluşabilecek tıkanıklar için tıkanıklık önleme yerine tıkanıklık kontrol teknikleri tercih edilmiştir. Bunu nedeni bu tür bir WSN sağlayıcı için genel dağıtım modelinin pub/sub olması ve Tip-A/B sınıfı kullanıcıların sayısının TCP ağlardaki kadar çok olamayacağıdır. Öte yandan kapalı döngü tıkanıklık kontrol yöntemleri de ek trafik oluşturması, istemci taleplerinin genellikle periyodik olmasından dolayı tercih edilmemiştir. Basitlik ve etkinliğinden dolayı önerilen sistemde açık döngü (open loop) tıkanıklık kontrol çözümlerden olan pencere politikası (window policy) benzeri bir teknik kullanılmıştır. Bu teknikte kuyruğa eklenecek olan VTask 'in oluşturulma zamanı ve VTask ID 'si flw_zVTask sınıfı ile ayrı bir listede (takip listesi) tutulmaktadır. Bu liste ayrı bir thread tarafından belirli zaman aralıkları ile kontrol edilmektedir. Liste verilerine göre ön tanımlı bir bekleme zaman çerçevesine (window size), belirli bir tolerans değeri (σ_w) kadar yaklaşan VTask 'ler ana kuyruktan çıkarılırlar. Kullanılan kuyruk yapısı ve çalışma akış şeması Şekil 3.16 'de verilmiştir. TaskQueue aynı zamanda zamanlanmış görevler listesine de sahip bir modüldür. İstemciler tarafından çalışma zamanları ayarlanmış VNd 'lere ait bilgiler yine bu modül içerisinde tutulmaktadır. Bu bilgiler Scheduler modülü tarafından sağlanmaktadır. Zamanı gelmiş olan VNd ler için VTask 'ler oluşturularak kuyruğa eklenmektedir. TaskQueue'a it çalışma akış diyagramı ve tanımlı olan diğer yordamlara ait bilgiler Tablo 3.5 'de sunulmuştur.

Tablo 3.5. TaskQueue sınıfı fonksiyonları

Fonksiyon	Parametre	Dönüş Tipi	Açıklama
addTask()	VTask	boolean	Kuyruğa yeni VTask ekler
removeFrom()	VTask	boolean	Belirtilen VTask ve sonraki tüm görevleri siler
size()	-	integer	Kuyruk boyutu
getzVTask()	pre_zVTask	boolean	Kuyruğun boş olup olmadığını kontrol
getTask()	-	VTask	Sıradaki VTask



a) TaskQueue Yapısı



b) Çalışma akış şeması

Şekil 3.16. TaskQueue yapısı ve tıkanıklık kontrol algoritması

Sistem genel davranışı Bölüm 4 ‘de açıklandığı üzere M/M/1/N olduğundan TaskQueue ‘daki zaman çerçevesinin boyutu “N” ‘e bağlıdır. Kuyruk kontrolünde Bölüm 4.4 Denklem 4.22-4.24 ve Denklem 4.38 ve 4.39 doğrudan etkilidir. Sisteme gelen ortala istemci talep süresinin birim zamandaki sistem hizmet süresine yaklaşması veya geçmesi durumunda sistem cevabındaki gecikme yüksek olacaktır. Bu nedenle istemci taleplerinin sistemde geçirdikleri sürelerin takip edilmesi gerekmektedir. Bununla alakalı detaylı Bölüm 4 ‘de verilecektir.

3.1.2.2.5.18. Scheduler (Zamanlanmış VNd ‘ler)

İstemci VNd ‘leri her zaman uzaktan gönderilen taleplerle çalıştırılmak zorunda değildir. Önerilen sistemin bir diğer özelliği, istemcilere kendi VNd ‘lerinin çalışma zamanlarını belirleyebilme imkanını sunmasıdır. Bu şekilde bir VNd farklı zamanlarda periyodik bir talep olmadan çalıştırılabilir. İstemci zamanlanmış VNd için web üzerinden gerekli ayarlamaları gerçekleştirmek ve çalışma programını belirlemek zorundadır. Scheduler modülü veri tabanından almış olduğu veriye dayalı olarak ilgi VNd ‘ye ait bilgileri içeren bir ara instance (pre_zVTask) oluşturur ve bunu TaskQueue ‘ya iletir. Zamanı gelen görev TaskQueue içerisinde VTask ‘e dönüştürülür ve kuyruğa bırakılır. Zamanlanmış VTask kuyruğa gönderildikten sonra gerekli bilgiler aynı zamanda AddAndFollow modülüne de gönderilmekte ve takibi buradan yapılmaktadır. Zamanlanmış VTask ‘in sonucu istemciye pub/sub ara katman yazılımı vasıtasıyla ulaştırılmaktadır. Bununla beraber zamanlanmış VTask ‘lerin doğrudan yürütüme gönderilmemesi ve kuyruğa atılması durumunda bir problem ortaya çıkmaktadır. Bir zamanlanmış VTask kuyruğa bırakıldığında üretilme zamanı kendinden önceki diğer VTask ‘lerden daha yeni olabilir. Bu durumda kendisinden önceki VTask ‘lerin zaman çerçevesi dışında kalması durumunda kendisi de kuyruktan çıkarılacaktır. Öte yandan doğrudan OperationRunner ‘a gönderilmesi durumunda ise kuyrukta bekleyen diğer VNd ‘lerin zaman çerçevesi dışında kalmasına yol açabilecektir. Bu, zamanlanmış VNd sayısının yüksek olması durumunda daha kötü bir performansa sebep olacaktır. Bu nedenle VTask ‘in kuyruğa gönderilmesi performans açısından daha iyi olacaktır. Zaman çerçevesi dışında kalan VTask ‘ler ikinci kez kuyruğa gönderilirler. Bir kez daha kuyruk dışında kalmaları durumunda ilgili hata koduyla istemciye bilgi gönderilmektedir. Bu modüle ait çalışma akış diyagramı Şekil 3.17 ‘de ve sınıf fonksiyonları ise Tablo 3.6 ‘de verilmiştir.

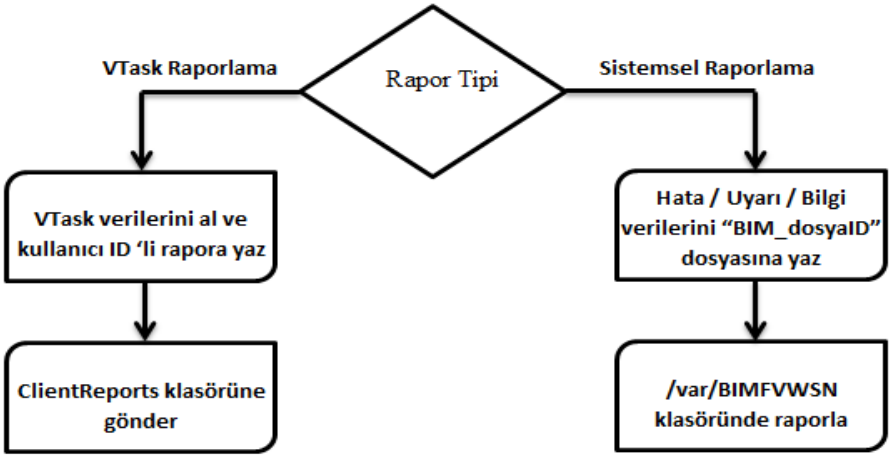
Şekil 3.17. Scheduler çalışma akış diyagramı

Tablo 3.6. Scheduler sınıf yapısı

Fonksiyon	Parametre	Dönüş Tipi	Açıklama
addZVTask()	pre_zVTask	void	Talep kaydının alınmasını kotarır
deleteZVTask()	Pre_zVTask	boolean	Bir zamanlanmış talebin silinmesini sağlar
sendToTaskQueue()	Pre_zVTask	boolean	Talep kaydını zamanı geldiğinde askQueue 'ya gönderir
size()	-	integer	Kaç adet zamanlanmış görev olduğunu verir

3.1.2.2.5.19. LogAndReport

Sistemin raporlama işlemlerini yürüten ana sınıftır. İstemcilerden gelen talepler doğrultusunda oluşturulan VNd 'lera ait tüm bilgiler sanal görev sınıf nesnelerinde tutulmaktadır. Sürecini olumlu veya olumsuz bir şekilde tamamlayan sanal görev nesnelerinin geçirdikleri süreçler bu sınıf nesnesi tarafından kaydedilir. Bununla birlikte sistem yürütümü esnasında olan olaylara ilişkin raporlamalar yine bu sınıf nesnesi tarafından takip edilmektedir. Raporlama sonuçları iki tipte kaydedilmektedir. Birinci tip, istemcilerin ulaşabildiği ve VNd operasyonlarındaki süreçleri raporlamaktadır. İkinci tip ise sistem yürütümü hakkında bilgileri içeren raporlardır. İkinci tip raporlar sadece sistem içi okunabilirliğe sahiptirler. Bu sınıfa ait fonksiyon yapısı Tablo 3.7 'de ve akış diyagramı Şekil 3.18 'da verilmiştir.



Şekil 3.18. LogAndReport çalışma akış diyagramı

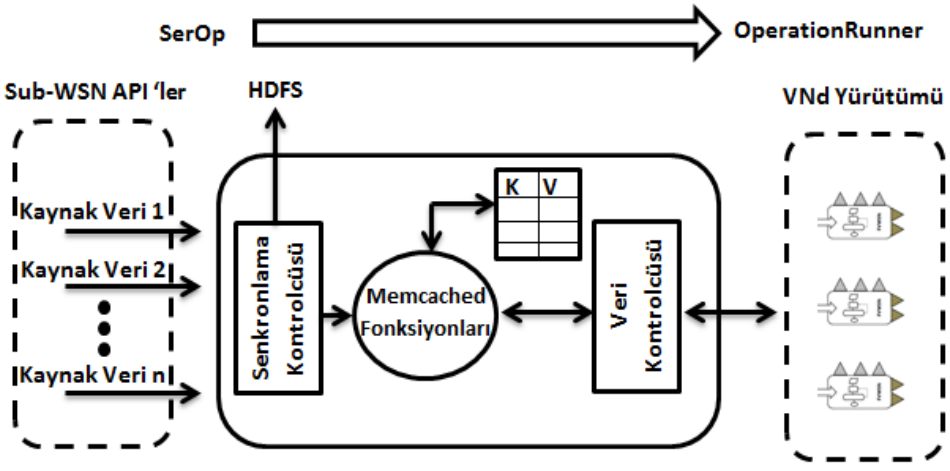
Tablo 3.7. LogAndReport sınıf yapısı

Fonksiyon	Parametre	Dönüş Tipi	Açıklama
addtoFile()	VTask	void	Sanal görev dosyasının verilerini rapor dosyasına döker
addtoError()	String	void	Hata koduna göre rapor dosyası hazırlar
getReport()	String	Path	Rapor ID 'sine göre dosya yolunu verir

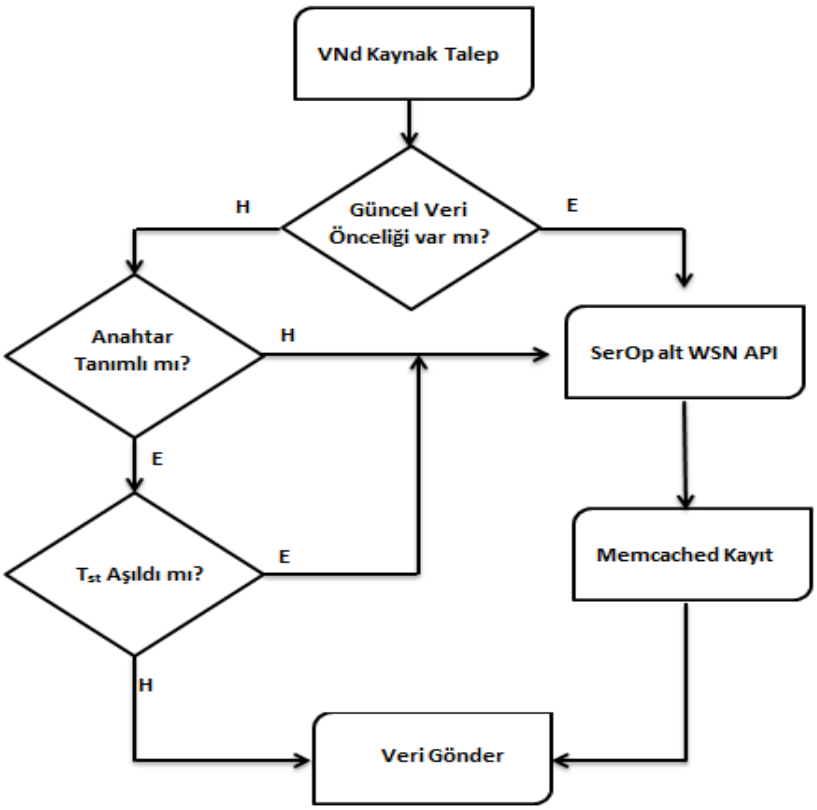
3.1.2.2.5.20. SynchronizingList

VNd 'ler seçilen kaynak verilerine göre işlem yapmaktadırlar. Bununla beraber, aynı kaynağı kullanan her bir VNd için kaynakların tekrardan fiziksel ağ üzerinden alınması hem performans hem de kaynağın sahibi düğümün enerji tüketimi açısından olumsuz olacaktır. Bu nedenle belli bir zaman içerisinde (T_{st}) aynı kaynağı kullanan VNd 'lerin önbelleklenmiş kaynak verilerini kullanması gerekmektedir. Bu nedenle T_{st} içerisinde aynı kaynak verilerine ihtiyaç duyan VNd 'ler senkron edilirler. Ancak bununla birlikte bazı VNd 'ler yüksek öncelik değerlerine sahip olabilir veya sürekli olarak güncel kaynak verileri ile çalışmak isteyebilir (örneğin güvenlik uygulamaları VNd 'leri). Bu durumda senkronlama işlemi bu tip VNd 'ler için uygulanmayacaktır. Kaynak verilerini senkronlama yönetimleri SynchronizingList sınıf nesnesi tarafından gerçekleştirilir. Sistemin senkron modda çalışıp çalışmaması hem sağlayıcı sistem hem de alt sistemler açısından oldukça önemlidir. Bunun yanında T_{st} 'nin optimum değerde tutulması bir başka kriterdir. Şöyle ki, her bir kaynak için

T_{st} değeri farklı olabilmektedir. Örneğin sıcaklık gibi fiziksel ölçümlerde ani değişimler çok kısa sürede olmayacağından bu kaynaklar için kullanılan T_{st} değeri daha uzun olabilir. Öte yandan bir aktuatör kaynağın birkaç VNd arasında paylaşılması durumunda aktuatörün güncel durumunun ani değişebileceği dikkate alınmalıdır. Bu durumda T_{st} değeri daha kısa seçilebilir. Benzer şekilde bir VNd farklı T_{st} değerlerine sahip kaynaklarla çalışabilir. Bu durumda bir VNd 'nin optimum yürütümünün hangi şartlarda sağlanacağı problemi ortaya çıkmaktadır. Bu konuyla ilgili detay hesaplamalar Bölüm 4 'de sistemin analitik analizinde verilecektir. SynchronizingList sınıf nesnesi, açık kaynak kod bir teknoloji olan "Memcached [117] temellidir. Memcached yüksek performanslı, anahtar-değer yapısında çalışan, dağıtık hafıza nesneleri önbellekleme sistemidir. Veri tabanı yükünü büyük oranda azaltmasından dolayı bugün Youtube, Wikipedia, Twitter gibi pek çok internet teknolojilerinde kullanılmaktadır. Ancak bu tez kapsamında önerilen sistemde Memchaced veri tabanı ve istemci arasına değil sadece VNd ve alt WSN API 'leri arasında bulunmaktadır. Sisteme ait senkronlama modülünü gösteren blok diyagramı ve akış diyagramı Şekil 3.19. a ve b 'de verilmiştir. SynchronizingList sınıfına ait fonksiyon yapısı ise Tablo 3.8 'de verilmiştir. Bu sınıfta kullanılan fonksiyonların bir kısmı Memcached fonksiyonlarının dolaylı olarak yürütümleri ile bağlantılıdır.



a) Senkronlama birimi blok yapısı



b) Akış diyagramı

Şekil 3.19. Senkronlama birimi blok yapısı ve akış diyagramı

Tablo 3.8. SynchronizingList sınıf yapısı

Fonksiyon	Parametre	Dönüş Tipi	Açıklama
putValue()	String, String	void	Önbelleğe değer at
getVelue()	String	Object	Önbellek değeri ver
isKeyValidate()	String	boolean	Anahtar tanımlı mı?
isTstExpired()	String	boolean	Tst aşıldı mı?
callSerop	String	String	Yeni değer al
deleteValue()	String	boolean	Önbellekten çıkar
clear()	void	boolean	Önbelleği temizle

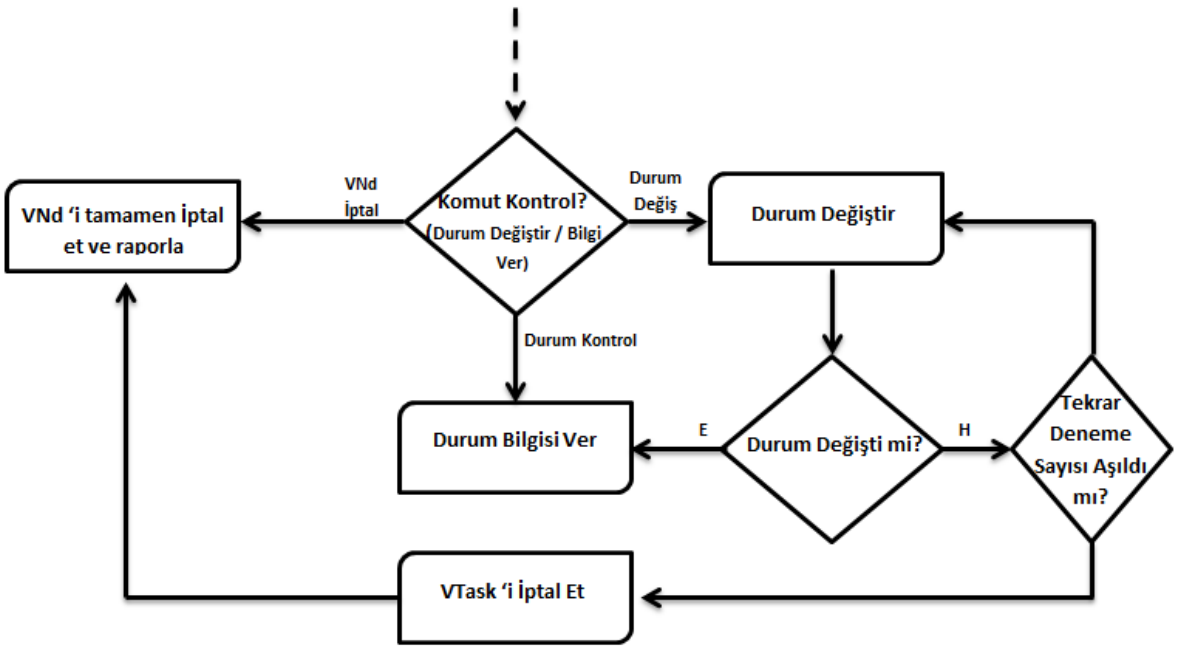
3.1.2.2.5.21. VNMonitor

Önerilen sisteme yüklenen VNd 'ler, aktif, çalışmayan veya uykuda gibi farklı durumlarda bulunabilirler. VNd taleplerinin alınması durumunda ilgili VNd 'lerin aktif duruma getirilmeleri, oluşacak aksi durumda sistemin bundan haberdar edilmesi ve raporlanması

sistem yürütümü açısından oldukça önemlidir. VNMonitor, VNd kayıtlarını tutan, onların durumu hakkında bilgilendirme yapan ve yine durumları arasında geçiş yaptırabilen sınıf nesnesidir. OperationRunner, VEngine gibi modüller tarafından sıklıkla kullanılan bir alt modüldür. Bir VNd 'nin yürütümü herhangi bir olumsuzlukla (güvenlik gibi) karşılaşırsa sistem bu modül aracılığı ile onu tamamen durdurabilir ve VNd kayıtlarını ilgili veri tabanından silebilir. Bu modül sınıfına ait çalışma akış diyagramı ve sınıf yapısı Şekil 3.20 ve Tablo 3.9 'da verilmiştir.

Tablo 3.9. VNMOnitor sınıf yapısı

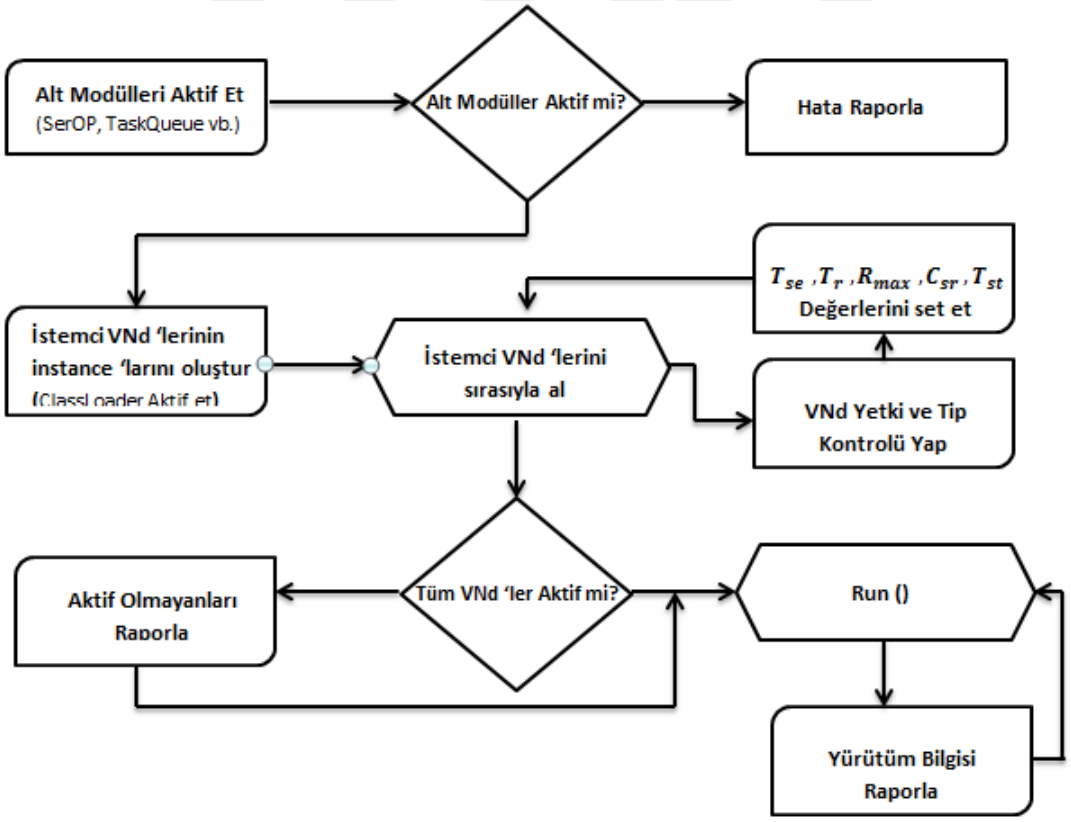
Fonksiyon	Parametre	Dönüş Tipi	Açıklama
getState()	String	void	VNd durumunu ver
changeState()	String	boolean	VNd durum değiştir
cancelVNd()	String	boolean	VND 'yi tamamen iptal et



Şekil 3.20. VNMonitor çalışma akış diyagramı

3.1.2.2.5.22. VEngine

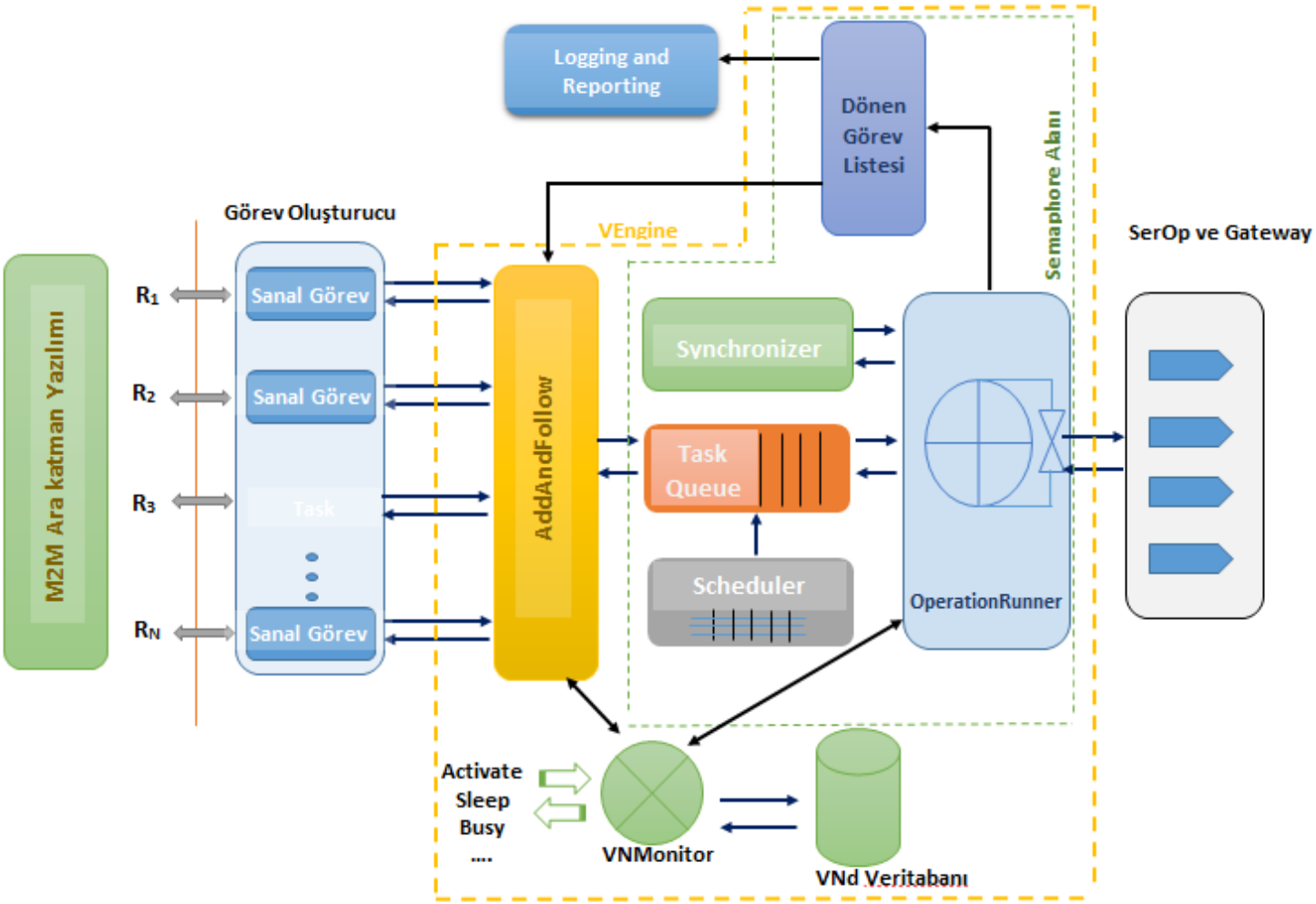
Sisteme ait pek çok alt modüllerin yürütümünü kontrol eden modüldür. Bu modüller; TaskFollower, TaskQueue, Scheduler, Syncronizer (SynchronizingList), VNMonitor ReturnList ve OperationRunner ‘dır. Bu modüllere ait instance ‘ları oluşturulması, sistem başlangıcında VNd ‘lerin aktif hale getirilmesi ve bunlara ait süreçlerin takibi bu modülde gerçekleştirilmektedir. VEngine aynı zamanda sanal görev oluşturucusu TaskCreator ile alt WSN kaynaklarına erişimi sağlayan SerOp modülü ile de bağlantı halindedir. Bu modül aynı zamanda FVWSN içerisindeki Auxiliary Class fonksiyonları kullanarak bazı önemli işlevleri de yerine getirmektedir. Bu işlevlerden ilki, yeni eklenen VNd sınıf instance ‘larının oluşturulup sistemi kesintiye uğratmaksızın yüklenmesini sağlamaktır. Bu işlev esnasında ClassLoader sınıf instance ‘ları kullanılmaktadır. Diğer yardımcı servisler ise VTask sonuç bildirimlerinin raporlanmasını sağlamak ve sistem durumuna göre optimum parametreleri set etmektir. Şekil 3.21 ‘de VEngine ‘nin çalışma akış diyagramı sunulmuştur.



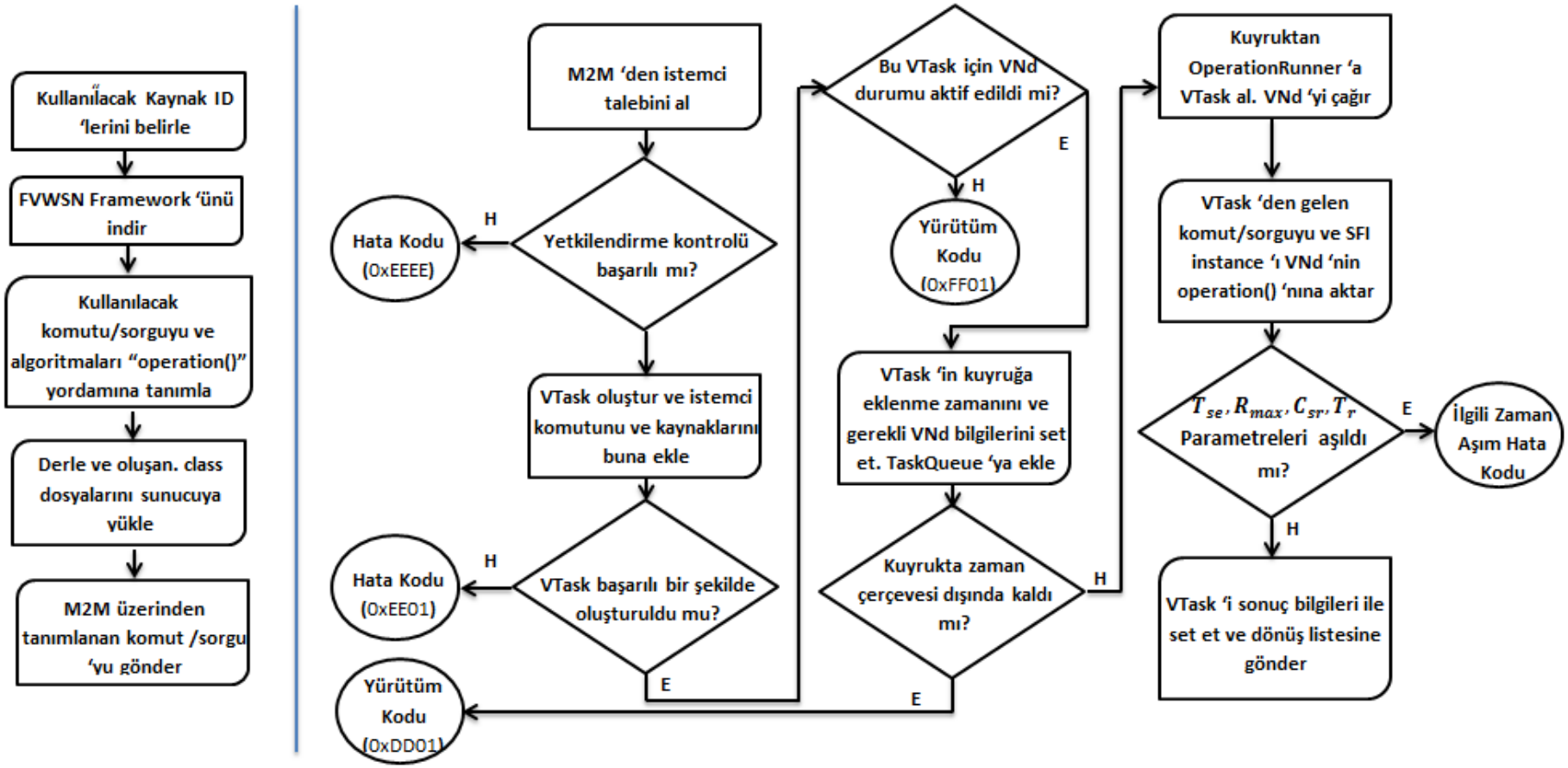
Şekil 3.21. VEngine çalışma akış diyagramı

Sistemin iç işleyişini gösteren genel blok şeması Şekil 3.22 ‘ de verilmiştir. Bu blok yapısı üzerinden bir istemci talebinin nasıl işlendiği hangi adımlardan geçtiğini gösteren akış şeması ise Şekil 3.23 ‘de gösterilmektedir. Sistem geliştirilebilir olması bakımından modüler bir yapıda tasarlanmıştır. Sistem işleyişinde öncelikle bir istemcinin kendi ortamında FVWSN framework ‘ü ile hazırlamış ve sunucu sisteme yüklemiş olduğu VNd class ‘lar gerekli güvenlik adımlarını geçmektedir. Sistem tarafından başarı ile yüklenen VNd class ‘ları VEngine tarafından ana sistemi kesintiye uğratmaksızın sisteme eklenir ve instance ‘ları oluşturulur. Daha sonra sağlayıcı sisteme M2M ara katman üzerinden gelen istemci talepleri için VTask ‘ler oluşturulur. Daha sonra bu VTask ‘ler gerekli bilgilerin set edilmesi ile AddAndFollow modülü tarafından kuyruğa gönderilir. Kuyrukta iki tip VTask bulunmaktadır. Bunlardan ilki M2M ile gelen talepler üzerine oluşturulmuş VTask ‘ler, diğeri ise Scheduler modülü tarafından oluşturulmuş VTask ‘ler. Bu TaskQueue modülü kendi içerisinde bir zaman çerçevesi kontrolü gerçekleştirmektedir. Bu çerçevenin dışında kalan tüm VTask ‘ler kuyruktan çıkarılır. Bu çıkarım işleminde çıkarılan tüm VTask ‘ler ilgili durum koduyla etiketlenerek raporlanmalarının yapılması sağlanmaktadır. Kuyruktan OperationRunner modülüne alınan VTask için ilgili yürütümü başlatılır. VTask ‘den alınan istemci komutu ve SFI instance ‘ı iligili VNd ‘nin “operation()” yordamına aktarılır. Bu VNd ‘nin ihtiyaç duyduğu kaynaklara ait okuma iki şekilde gerçekleştirilir. Mode-1 çalışmada yürütüm esnasında istenmesi durumunda, Mode-2 ‘de ise yürütüm başlatılmadan hemen önce kaynak verileri elde edilir. Bu verilerin elde edilmesi ise yine kendi içinde iki şekilde yapılır. VTask ‘in öncelik derecesine göre ya Syncronizer modülünden ya da doğrudan SerOP alt yüklenici API ‘lerinden bu verileri elde eder. Yürütüm işleminde dikkat edilen üç temel husus vardır. Bunlardan ilki her bir VNd yürütümü belli bir sürede tamamlanmalıdır. İkincisi her bir VNd belli sayıda kaynak okuyabilir/işleyebilir. Üçüncüsü ise bir kaynak yine belli sayıda yürütüm gerçekleştirebilir. Bu kriterlere takılan her VNd başarısız yürütüm koduyla etiketlenmektedir. Yürütümü başarılı veya başarısız olarak tamamlanan tüm VTask ‘lere ait sonuçlar istemcilere M2M veya Pub/Sub ara katman aracılığı ile döndürülür.

Bu bölümde anlatılan FVWSN yazılım çerçevesi ile bulut sistem üzerinden nasıl WSN sanallşatırma ve kaynak paylaşımı yapıldığı EK-2 ‘de bir örnek ile açıklanmıştır.



Şekil 3.22. AYÇ sistem yürütüm modülleri genel yapısı



a) istemci tarafı işlemler

b) sunucu tarafı işlemler

Şekil 3.23. Bir VNd nin genel çalışma akış şeması

3.1.3. Servis-Kaynak Keşfi Modülü

Önerilen sistemde servis ve kaynak keşfi iki şekilde yapılmaktadır. Bunlardan ilki istemcilere sunulan web arayüzü üzerinden yapılan keşif işlemleridir. Bu şekilde yapılan keşif işlemleri genelde sisteme kayıt öncesi veya yeni VNd oluşturma öncesi gerçekleştirilen adımlar arasında yer almaktadır. İkinci servis- kaynak keşfi ise FVWSN içerisinde sunulan ve daha çok servis-kaynak kontrolü için kullanılan yordamlara dayalıdır. Her servis ve kaynak eşsiz bir ID 'ye sahip olduğundan dolayı VNd yürütümü esnasında sorgulanabilirdir.

3.1.4. Kayıt Yönetimi, Kaynak Atama ve Kaynak Enerji İzleme Modülü

Önerilen sistemi kullanmak isteyen istemciler ve alt WSN sahipleri öncelikle gerekli kayıt sürecini geçmek zorundadır. Sisteme ait güvenlik gereksinimleri gereği kullanıcılar belirli yetkilerle sahip olabilmektedir. Kullanıcılara verilen yetkilerin tanımı ve kimliklendirme kayıt süresince belirlenmektedir. Bu kimliklendirme daha sonra oluşturulacak VNd 'tanımlamaları ve sisteme erişim için kullanılacaktır. Benzer şekilde bir alt WSN sisteme dahil olmadan önce önerilen sistem tarafından istenilen gerekli bilgileri sistem kaydetmek zorundadır. Bu özellikle kaynak izleme ve atama süreçlerinde hayati rol oynamaktadır. Bir alt WSN sahibi hangi kaynakları paylaşabileceğini ve ilgili API 'leri bu safhada belirleyecektir. Önerilen sistem klasik kaynak atama modellerinden farklı bir yaklaşım kullanmaktadır. Klasik kaynak atam yöntemlerinde genellikle istemci taraflı parametreler dikkate alınmaktadır. Matematiksel detayları Bölüm 4.4 'de Denklem 4.18-4.21 'de matematiksel modeli verilen alternatif bir yaklaşım kullanan bu sistem modelinde ise hem istemci hem de alt WSN tarafı parametreler dikkate alınmaktadır. Alt WSN 'lerde bulunan düğüm kapasiteleri, kaynak durumları ve yüklenebilecekleri yük parametreleri dikkate alınarak kaynak atama süreci gerçekleştirilmektedir. Bu parametrelerin kontrolü Kaynak Enerji İzleme modülü tarafından gerçekleştirilmektedir. Bu modül kendisine alt WSN 'lerden gerekli bilgilerin gelmemesi durumunda tahmini bir enerji izleme yöntemi kullanmaktadır. Alt WSN 'lerin düğümlerinin lokasyonlarının, çalışma düzenlerinin ve kullandıkları protokol yapılarının bilinmesi durumunda düğümlerin enerji seviyelerini uzaktan tahmin edebilen bir mekanizma bu bölümde bulunmaktadır. Bununla beraber bir

alt WSN düğümünün işlem kapasitesi, ilgili alt WSN sistemin sağlayıcı sisteme kaydı esnasında bilindiğinden, kaynaklar üzerine yapılabilecek dengeleme işlemini de kolay bir şekilde başarılabilecektir. Bu modellerle ilgili daha detaylı bilgi Bölüm 4'de matematiksel olarak sunulmuştur.

3.1.5. Hata Tolerans Modülü

Önerilen sistemde meydana gelebilecek hatalar iki sınıfta değerlendirilmiştir. Bunlardan ilki VNd yürütümleri esnasında oluşan hatalar ikincisi ise büyük veri yönetiminde oluşan hatalar. Büyük veri bölümünde oluşabilecek hatalara ait operasyonlar Hadoop teknolojisinin sağlamış olduğu gelişmiş hata tolerans fonksiyonlarınca yürütülmektedir. Bu nedenle büyük veri işleme ile ilgili ikinci bir hata tolerans yöntemi kullanılmamıştır. Diğer hataya müsait olan bölüm VNd ve VEngine ile alakalı modüllerdir. Bölüm 4.6 'da ifade edildiği gibi bu yürütüm hataları üç şekilde meydana gelmektedir. Bunlardan ilki bireysel VNd yürütüm hataları, ikincisi bütünsel VEngine işlevsizliği ve üçüncüsü ise alt WSN 'lerden kaynaklanan hatalardır. Bireysel VNd hatalarında VEngine tarafından durdurma, beklemeye alma/yeniden başlatma uygulanır. Gerekli raporlama ve bildirim ilgili hata koduyla bildirilmektedir. Genel VEngine hataları için ise "Replikasyon" teknikleri kullanılmıştır. Anlık sistem kesintilerinde ve sistem çökmelerinde sistemin kesintiye uğramaması için anlık veri replikasyonu başka bir sanal makine üzerinde tutulmaktadır. Bu durumda anlık kuyruk verileri olduğu gibi VEngine 'e tekrardan aktarılabilmektedir. Hata yönetimleri ile ilgili modeller ve tolerans yöntemleri Bölüm 4.6, Denklem 4.53-4.54 'de verilmiştir.

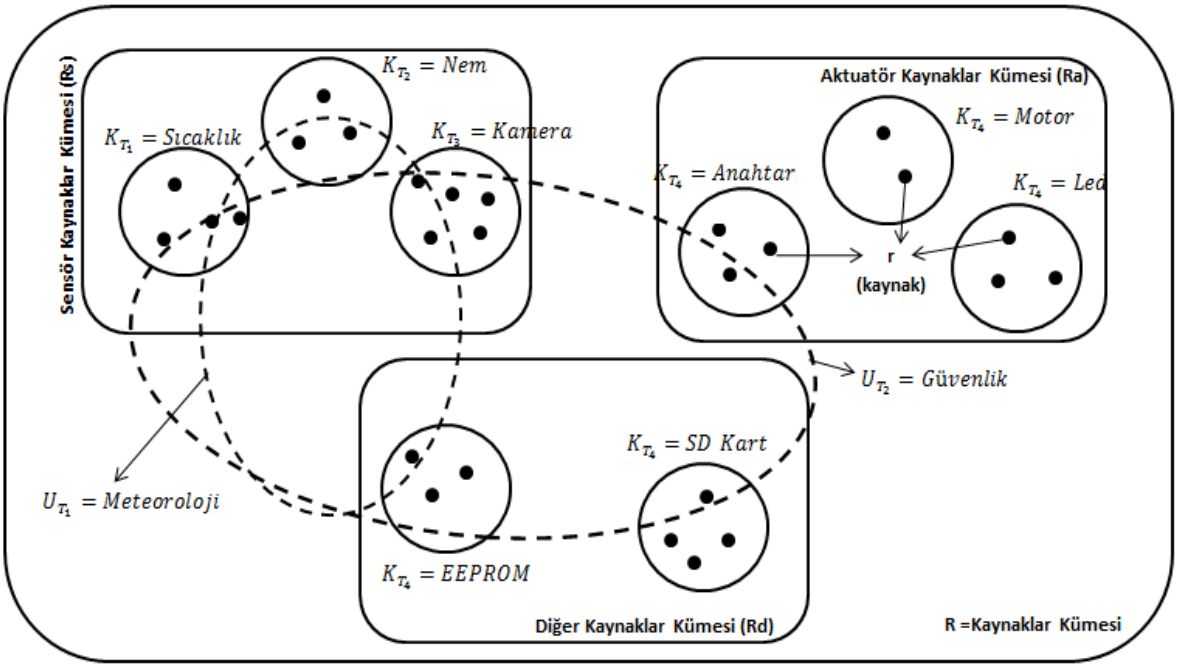
3.1.6. Güvenlik

Önerilen sistemde istemci kodlarının sağlayıcı tarafta çalıştırılması doğal olarak güvenlik sorunlarını ortaya çıkarmaktadır. Her ne kadar sistem güvenlik konusunda iyileştirmelere ihtiyaç duysa da temel olarak iki aşamalı bir güvenlik mekanizması içermektedir. Bunlardan ilki istemci sınıflandırmalarıdır. İstemciler üç grupta tanımlanmıştır. Bunlar Tip A, Tip B ve Tip C 'dir. Tip A istemciler, gerekli yasal zorunlulukları onaylamış olan ve VNd oluşturarak sisteme yükleme ve bunlara kimlik denetimli M2M arakatman yazılımı ile ulaşabilen gruptur. Tip B istemciler sadece kendi

ağlarında kullandıkları komut/sorgular ile kaynaklara ulaşabilen ve özel algoritmalar çalıştıran VNd ‘leri oluşturma yetkisine sahip olmayan istemcilerdir. Bu istemci sınıfı kendi VNd ‘lerini web ara yüzü aracılığı ile oluşturmaktadırlar. Bu istemcilerde gerekli yasal gereksinimleri sağladıkları varsayılır. Tip C ise sistemi sadece Pub/Sub ara katman yazılımı üzerinden kullanabilen ve her hangi bir VNd operasyonu yürütemeyen istemcilerden oluşmaktadır. Tip A ve B aynı zamanda sistemi ful kullanma yetkisine sahip olduklarından Pub/Sub modülünü de kullanabilmektedir. Bu güvenlik önlemlerine ek olarak ikinci seviye güvenlikte, VNd operasyonlarına Java Security ile okuma/yazma sınırlamaları getirilmiştir. Böylece sistem bazlı okuma, yazma ve erişimler güvenlik politikalarına uygun olarak kısıtlanmaktadır. Ancak daha önce de söylendiği üzere önerilen sistem üzerine devam edilecek olan sonraki çalışmalarında güvenlik iyileştirmeleri ilk sırayı alacaktır.

4. ÖNERİLEN SİSTEMİN ANALİTİK ANALİZİ

Bu bölümde, önerilen sistemin OPNET simülatöründe simülasyonu için gerçekleştirilen teorik modellemesi açıklanacaktır. Bunun için ilk olarak kullanılacak genel tanımlar sunulacak ve daha sonra kaynak seçim ve atama sürecine ait teorik model açıklanacaktır. Daha sonra sistemdeki genel yürütüm prensibine ait çalışma modeli verilecektir. Alt WSN sistemlerde gerçekleşen süreçler, enerji sarfiyatı, düğüm hafıza ve işlem yükü incelenecek ve bu alt WSN 'lere ait çalışma modeli teorik olarak gösterilecektir. Daha sonra sistemin hata tolerans alt yapısı ve diğer yardımcı modüllere ait teorik çalışma modelleri gösterilecektir. Bu bölümde kullanılan teorik yaklaşımlarda [63,64,118] 'de sunulan temel prensipler baz alınmıştır. Ancak burada sunulan prensipler ya genel geçer ya da amaca yönelik senaryoları baz aldığından doğrudan bu tezde önerilen yapıları ifade edememektedirler. Bu nedenle bu yaklaşımalar sistem modeline göre revize edilmiştir.



Şekil 4.1. Örnek bir sistem kaynaklar kümesi ve tanımlı diğer alt kümeleri

Şekil 4.1, aşağıda yapılacak olan bazı tanımların daha iyi anlaşılması için verilmiş bir örnek şemadır. Bu şema sistemin kaynaklar kümesini ve içinde tanımlı diğer alt kümeleri göstermektedir. Şekilde de görüleceği üzere kaynak tipleri uygun senaryolarla eşleştirilebilmektedir. Bu özellikle kaynak atama safhasında önemli bir kullanım kolaylığı sağlamak içindir.

4.1. Genel Tanımlar

Tanım 1: Önerilen sağlayıcı sistemde bulunan alt WSN sistemler "$\boldsymbol{W_k}$" ile gösterilmektedir ve bu durumda sağlayıcı sistem $\boldsymbol{S} = \{\boldsymbol{W_1}, \boldsymbol{W_2}, ..\}$ şeklinde ifade edilir. Belirli bir anda sistemdeki alt WSN sayısının değişken olabileceği ancak sıfır olamayacağı varsayılmaktadır. Bu durumda sistem her zaman $\boldsymbol{0 < |S| < \infty}$ şeklinde ifade edilecektir.

Tanım 2: Tüm alt WSN sistemler ticari amaçlı veya gönüllü bir kuruma/kişiye ait olabilirler. Bu durumda gerek kaynak paylaştırmada gerekse ekonomik işlemlerde kullanılmak üzere alt WSN sahipleri $\boldsymbol{O} = \{\boldsymbol{O_1}, \boldsymbol{O_2}, ..\}$ ile ifade edilecektir. Sistem farklı şahıs/kurumlara ait alt WSN 'lere sahip olmak zorunda değildir. Bu alt WSN 'ler farklı sistem çatısı altında oluşturulmuş farklı otonom yapılar olabilmektedir. Başka bir deyişle $\boldsymbol{0 \leq |O|}$ olarak ifade edilebilir.

Tanım 3: Sistem altındaki tüm kaynaklar **"R"** kaynaklar kümesinin elemanıdır. Bununla bütün kaynaklar sensör kaynaklar **($\boldsymbol{R_s}$),** aktuatör kaynaklar **($\boldsymbol{R_a}$)** veya diğer kaynaklar **($\boldsymbol{R_d}$)** alt kümelerinden birinin elamanıdır. Bu durumda $\boldsymbol{R_s, R_a, R_d \subseteq R}$ 'dir.

Tanım 4: Her bir kaynak belirli bir fiziksel olay tipi ile etiketlenmektedir. Bu nedenle sistem, ön tanımlı bir kaynak tipi veri setine (kümesine) sahiptir. Bu $\boldsymbol{K_T} = \{\boldsymbol{K_{T_1}}, \boldsymbol{K_{T_2}}, ..\}$, $\boldsymbol{K_{T_l} \in K_T}$ ile temsil edilecektir. Örneğin K_{T_1}="Sıcaklık" 'lığı ve K_{T_2}="Nem" 'i gösterebilir. Burada $\boldsymbol{0 < |K_T| < \infty}$ olduğu kabul edilmektedir ve $\boldsymbol{K_{T_l} \subseteq K_T \subseteq R}$ 'dir.

Tanım 5: Önerilen sistem kullanım kolaylığı ve daha hızlı kaynak paylaşımı düzenlemesi yapılabilmesi için ön tanımlı bir uygulama tipi kümesi de içermektedir. Bu küme $\boldsymbol{U_T} = \{\boldsymbol{U_{T_1}}, \boldsymbol{U_{T_2}}, ..\}$, $\boldsymbol{U_{T_t} \in U_T}$ ile gösterilmektedir. Örneğin burada U_{T_1} "meteoroloji" yi U_{T_2} ise "güvenlik" i temsil edebilir. Sistemde tanımlı her bir kaynak en az bir uygulama tipine sahiptir. Her kaynak aynı zamanda "Genel Amaçlı" tipine sahip olup istemcinin bu tipi seçmesi ile tüm kaynaklar görüntülenmektedir. Bu demek oluyor ki U_{T_t} kümesi R 'de tanımlı alt kümeleri temsil etmektedir. Yani $\boldsymbol{U_{T_t} \subseteq R}$ olarak ifade edilir.

Bir alt kaynak tipi sisteme kaydedilirken "$\boldsymbol{g_{tip}}(.)$" fonksiyonu ile ilgili uygulama tipleri ile eşleştirilmektedir. Bu fonksiyon tanımı $\boldsymbol{g_{tip}}(.): \boldsymbol{R} \longrightarrow \boldsymbol{U_T}, \boldsymbol{U_{T_t}} \in \boldsymbol{2^{|R|}}$ şeklindedir.

Tanım 6: k. ninci WSN ağdaki i .ninci düğümün özellik notasyonu $\boldsymbol{N_i^{W_k}}$ ile gösterilecek ve bu 3-tuple olarak şu şekilde ifade edilecektir;

$$N_i^{W_k} =< M_{d_i}, E_{d_i}, L_{d_i} > \tag{4.1}$$

Burada sırası ile $\boldsymbol{M_{d_i}}$, işlem oranı (processing rate-MIPS), $\boldsymbol{E_{d_i}}$ mevcut enerji miktarını (J) ve $\boldsymbol{L_{d_i}}$ sistem tarafından belirlenmiş kullanım maliyeti katsayısını ifade eder. Örnek olarak bu sistemin gerçekleştirildiği WSN laboratuvarında kullanılan TelosB düğümler için bu değerler $M_{d_i} = 8\ MIPS, E_{d_i} = 32400\ J, L_{d_i} = 1$ 'dir.

Tanım 7: Bu tip bir sistemde bulunan bir alt WSN düğüm üzerindeki tüm kaynaklar aynı talebi görmeyebilir. Bu nedenle kaynak bazlı bilgilerin sistemde tutulması gereklidir. Daha sonra da görüleceği üzere bu bilgiler tahmini enerji seviyeleri ve fiyatlandırma gibi alt işlevlerde kullanılabilecektir. Bu nedenle her bir kaynağın sisteme ve üzerinde bulunduğu düğüme etkisini gösteren 3-tuple notasyonu bulunmaktadır;

$$\hat{E} =< g, m, \Theta > \tag{4.2}$$

burada **"g"** kaynağın üretebileceği veri boyutu (kb/s), **"m"** kaynağın işlem yükü (MIPS), **"Θ"** ise kaynak kullanım maliyetini ifade etmektedir. Örneğin 127 byte 'lık sabit boyutlu paketlerin kullanıldığı varsayılan bir 802.15.4 ağında 0.25Hz 'lik örnekleme hızında çalıştırılan bir nem sensörünün üreteceği veri boyutu yaklaşık g =0.25 kb/s olacaktır. Bunun yanında sıcaklık, nem, ışık gibi temel fiziksel büyüklük sensörlerinin işlem yükü ihmal edilebilecek düzeyde iken kamera gibi kaynakların kullanıldığı durumlarda 10 ile 100 MIPS 'lik bir işlem yükü söz konusu olabilmektedir [1,64].

Tanım 8: Sistem içerisindeki bir kaynak her an için aktif olmayabilir. Bu nedenle bir kaynağın aktif veya pasif olması durumu ikili (binary) tip "$\boldsymbol{d}$" değişkeni ile gösterilecektir. Burada d=1 olması kaynağın aktif, d=0 olması ise pasif olması anlamına gelmektedir.

Tanım 9: Sistem içerisinde bulunan aktutörlerin birden fazla uygulama tarafından kullanılması durumunda gerekli izinler "$\boldsymbol{a},\ \boldsymbol{a} \in \boldsymbol{Z}$" değişkeni ile kontrol edilecektir. Bir aktuatöre ait bu değişkenin değerinin sıfır olması bu kaynağın hiçbir istemci tarafından

kullanılamayacağı anlamına gelmektedir. Bu değerin "-1" olması ise bu kaynağın hiçbir şekilde aktuatör olarak değerlendiremeyeceğini ifade edecektir. Her bir aktuatör kaynağa ait bu değer sistem tarafından veya ilgili $\boldsymbol{O_i}$ tarafından belirlenmekte ve istemcilere ilan edilmektedir. Bu değerin değiştirilmesi istemci VNd 'sinin sahip olduğu öncelik değeri doğrudan ilişkilidir. Öncelik değeri yüksek olan VNd bu değeri değiştirme hakkına sahiptir. Bir aktuatör kaynağı hangi VNd 'lerin kullanabileceği sistem tarafından belirlenebilmekte ve özel izin gerektirebilmektedir.

Tanım 10: Sistemde tanımlı olan bir kaynağa ait genel notasyon ise aşağıdaki gibi 8-tuple bir notasyona sahip olacaktır;

$$r =< id, o, \ddot{\mathrm{t}}, d, a, N_i^{W_k}, \hat{\mathrm{E}}, \{P_r\},\ R_{pos} >, o \in O, \ddot{\mathrm{t}} \in U_T \tag{4.3}$$

"**id**", kaynağa sistem tarafından "$\tilde{\mathrm{I}}_{\boldsymbol{id}}$ kaynak kimlik kümesi "nden verilen benzersiz bir kimlik numarasıdır. "**d**" kaynağın güncel durumunu, "a" ise kaynağın aktuatör olması durumunda ortak kullanım değerini vermektedir. Bu değer bu aktuatörü kullanmaya yetkili VNd sayısına bağlı olup sistem tarafından atanmaktadır. Kaynak eğer bir sensör ise otomatik olarak bu değer "-1" dir. $\{P_r\}$ ilgili kaynağın çalışma aralıklarını göstermektedir. Öte yandan "$\boldsymbol{R_{pos}}$" kaynağın bulunduğu koordinatı enlem ve boylam olarak veren değişkendir. Bu değişkenlerden bazıları istemcilerle paylaşılabilirken bir kısmı sadece sistem tarafından kullanılmaktadır.

Tanım 11: Sistemi kullanan istemciler $\boldsymbol{I} = \{\boldsymbol{I_1}, \boldsymbol{I_2}, ..\}$, $\boldsymbol{I_p} \in \boldsymbol{I}$ kümesinde tanımlıdır. Bir istemci ikili tip çok değişkenli (2-tuple) ile şu şekilde gösterilmektedir;

$$I_p =< \Phi, \{V^{I_p}\} > \tag{4.4}$$

Burada "$\boldsymbol{\Phi}$" istemciye ait kullanıcı tipini göstermektedir ve istemci tip kümesi " **Ś**" de tanımlı bir değer almaktadır ($\boldsymbol{I_p}.\boldsymbol{\Phi} \in$ Ś). Daha sonra görüleceği üzere bu kullanıcı tipi sistem yürütümünde ve güvenlikte kullanılacaktır. Bununla beraber bir istemciye ait olan VNd 'ler $\{\boldsymbol{V^{I_p}}\}$ 'de tanımlıdır ve $\boldsymbol{V^{I_p}} = \left\{\boldsymbol{V_1^{I_p}}, \boldsymbol{V_2^{I_p}}, ..\right\}$ ile gösterilmektedir. Her bir istemci uygulama tipine göre farklı sayıda VNd 'ye sahip olabilir. Bu sayı aynı zamanda bir VNd

‘nin yönetebileceği maksimum kaynak sayısı ($\boldsymbol{R_{max}}$) ile de ilişkilidir. Bu nedenle aynı ölçüm alanı için bir $\boldsymbol{I_p}'$nin birden fazla VNd ‘si bulunabilir.

Tanım 12: Sistem geneli için her hangi bir komut/sorgu q_{xx}^{X} ile gösterilecek ve Q^X kümesinde gösterilecektir. Üst indis X ‘in “W_k” olması i. ninci alt ağı, “S” olması sistem komut setini, “I_p” olması da istemci tarafındaki tanımlı komut setlerini ifade edecektir. Alt indis “xx” ise ilgili komut indisini verecektir. Bu durumda alt WSN sistemlerde kullanılan komut ve sorgular $\boldsymbol{Q^{W_k}} = \{\boldsymbol{q_1^{W_k}}, \boldsymbol{q_2^{W_k}}, ..\}$ kümesinde tanımlıdır. Bu küme içerisinde tanımlı her bir komut/sorgu belli bir kaynağa ait veri edinme veya set etme işlemlerini yerine getirmektedir. Benzer şekilde sunucu sistemde tanımlı olan komut kümesi $\boldsymbol{Q^S} = \{\boldsymbol{q_1^S}, \boldsymbol{q_2^S}, ..\}$ ve bir istemci de tanımlı komut/sorgu seti ise $\boldsymbol{Q^{I_p}} = \{\boldsymbol{q_1^{I_p}}, \boldsymbol{q_2^{I_p}}, .., \}$ ile ifade edilecektir.

Tanım 13: Her bir istemci VNd uygulaması ise aşağıdaki gibi 4-tuple olarak ifade edilecektir.

$$U = < U_{id}, U_{tip}, U_{alan}, \{P_u\} > \tag{4.5}$$

Kullanıcı uygulaması bir “id” ile ayırt edici özelliğe sahiptir. U_{alan} ise sanal düğümün görüntüsünü oluşturacağı ilgili alanın karesel koordinatlarını temsil etmektedir, U_{alan} =<$Koor_1, Koor_2, Koor_3, Koor_4$>. $\{P_u\}$ ilgili uygulama için istenilen çalışma periyotlarını içeren kümedir. $CI_i =< b_i, e_i >$ bir uygulamaya ait çalışma zamanını gösteren notasyon olmak üzere $\boldsymbol{P_u} \in \boldsymbol{CI_i} \in \boldsymbol{P_r}$ ‘dir. Burada “b_i” başlangıç zamanını “e_i” bitiş zamanını göstermektedir. Bunun yanında U_{tip} istemcinin sunucu tarafta tanımlı uygulama tipilerinden yaptığı seçimi temsil eder.

Tez kapsamında kullanılacak olan gerekli tanımlar yapıldıktan sonra WSN ağlara ait bazı temel bilgilerin sunulması faydalı olacaktır. Bunlardan ilki bir WSN ağdaki (W_k) herhangi bir düğümde (“i” veya “h”) gerçekleşen güç tüketimleridir. Bir düğümdeki güç tüketimleri genel olarak 3 grupta incelenir. Bunlar veri iletiminde ki güç tüketimi ($\boldsymbol{P_i^{Tr}}$), veri alımındaki güç tüketimi ($\boldsymbol{P_i^{Rc}}$) ve kaynak okuma/veri işleme sürecindeki güç tüketimidir ($\boldsymbol{P_i^{OYI}} = \boldsymbol{f_{OYI}(r.id)}$) [119]. Bu ifadelere ait eşitlikler şu şekildedir.

$$P_i^{Tr} = \sum_{h \in W_k, i \neq h} \left(\beta_1 + \beta_2 d_{ih}^{\gamma}\right) Pl_{ih} \qquad \forall i,h \in W_k \tag{4.6}$$

$$P_i^{Rc} = \rho \sum_{h \in W_k, i \neq h} Pl_{ih} \qquad \forall i,h \in W_k \tag{4.7}$$

$$P_i^{tot} = P_i^{Tx} + P_i^{Rx} + f_{OYI}(r.id) \; \forall \; i,h \in W_k \tag{4.8}$$

Burada $\boldsymbol{P_i^{OYİ}}$**,** düğüm üzerindeki kaynak tipine, bu kaynaklardan alınan verilerin nasıl işlendiğine yani işlem yüküne bağlıdır. Basit kaynakların (sıcaklık, nem sensörü vb.) veri edinimleri basit olduğundan ve kompleks veri işlemelere gerek duymadığından bu tip bir güç tüketimi, radyo işlemlerinde gerçekleşen güç tüketimleri yanında ihmal edilebilir. Bunun yanında multimedia uygulamaları gibi daha karmaşık senaryoları çalıştıran WSN düğümlerde $\boldsymbol{P_i^{OYİ}}$ ihmal edilemeyecek düzeydedir. $\boldsymbol{P_i^{OYİ}}$ değeri $f_{OYI}(r.id)$ fonksiyonu ile elde edilmektedir. Bu sistem tarafından belirleneceği gibi alt WSN tarafından da hesaplanabilen bir fonksiyon olabilir. Pl_{ih} , W_k ağındaki "i" ve "h" düğümleri arasındaki veri akışını (bps) temsil etmektedir. Literatürde β_1=50 nj/bit , β_2 = 0.0013 pJ/bit/m4, ρ = 50nj/bit ve γ = 4 değerlerine sahip olan sabitlerdir [120]. d_{ih} ise "i" ve "h" düğümleri arasındaki mesafeyi temsil etmektedir.

Bunlara ek olarak bir WSN düğümün iletim gücüne bağlı olarak kapsama yarıçapı da Denklem 4.9 'e göre hesaplanabilmektedir.

$$R_i^T(p_{tx}) = (p_{tx}.\frac{g_0}{P_{trs}})^{1/\gamma} \tag{4.9}$$

Bu denklenmede, "$\boldsymbol{P_{trs}}$**"** belirli paket alım eşik RSSI değerini (örneğin -95 dBm), $\boldsymbol{g_0}$ ise anten bağlı sabiteleri göstermektedir (örneğin 0.0081) [121].

Bir WSN düğümü veri toplama ve veri yönlendirme yeteneklerine sahiptir. Klasik bilgisayar ağlarında olduğu gibi yönlendirme için özel cihazlara gerek yoktur. Bu durumda her bir düğüm için "Akış Korunumu (Flow Conservation)" kuralı geçerlidir. Bu kurala göre bir düğüme gelen veri akış oranı ile o düğümde üretilen veri oranının toplamı, düğümden çıkan veri akış oranına eşit olmalıdır. Bu kurala ait ifade Denklem 4.10 'da verilmiştir.

$$\sum_{\substack{h \in W \\ i \neq h}} I_{f_{hi}} + Gx \sum(\hat{E}.g) = \sum_{\substack{h \in W \\ i \neq h}} O_{f_{ih}} \tag{4.10}$$

Bu denklenmede $\boldsymbol{I_{f_{hi}}}$, düğüm "i" ye diğer düğümlerden gelen veri akışını, $\boldsymbol{O_{f_{ih}}}$, ilgili düğümden yönlendirme yapacağı düğüme yaptığı veri akışını göstermektedir. "Ê. $\boldsymbol{g}$" değeri açıklandığı üzere düğüm üzerinde aktif kaynakların üreteceği veri oranını vermektedir. "**G**" katsayısı sistem ve ilgili "$\boldsymbol{O_i}$" arasında belirlenmiş bir sabite olup bununla ilgili detay aşağıda kaynak seçme ve atama bölümünde verilecektir.

4.2. Problemin Tanımı:

WSN sistemlerin büyük bir bölümü bir uygulama katmanına sahiptir. Bu katmanda genel olarak iki tip çalışma yürütülmektedir. Bunlardan ilki tanımlı komut/sorguları alıp gerekli yürütümleri yapmak veya bu komut/sorgularla ağ kaynaklarıyla etkileşim kurmak, ikincisi ise varsa gerekli uygulama katmanı servislerini/yazılımlarını yürütmektedir. Farklı uygulama katmanlarına sahip WSN 'leri içeren, yani "katman bazlı heterojenlik" e sahip bir IoT-WSN yapısı ele alındığında, bir "$\boldsymbol{W_k}$ ($\boldsymbol{W_k} \in \boldsymbol{S}$)" alt WSN ağındaki komutlar /sorgular $\boldsymbol{Q^{W_k}}$ 'da tanımlı olacaktır. Aynı şekilde "$\boldsymbol{I_p}$ ($\boldsymbol{I_p} \in \boldsymbol{I}$)" istemcisinde kullanılan komut/sorgular $\boldsymbol{Q^{I_p}}$ 'de tanımlıdır. Pek çok IoT WSN sistemde olduğu gibi bu tez kapsamında önerilen sistem de alt yapı karmaşıklığını istemcilerden gizlemektedir. Alt WSN 'lerin sisteme dinamik olarak dâhil olup ayrılabilmeleri, oluşacak alt WSN problemlerinden istemcilerin etkilenmemesi gibi durumlardan dolayı şeffaflık hayati bir özelliktir. Bu nedenledir ki sistem alt kaynaklara, kendi komut/sorgu seti olan $\boldsymbol{Q^S}$ üzerinden ulaşmaktadır. Bu ise bir $\boldsymbol{g_{SW}(.)}$ fonksiyonu ile gerçekleştirilmektedir. Bu durumda "$\boldsymbol{g_{SW}}: \boldsymbol{Q^S} \rightarrow \boldsymbol{Q^{W_k}}$" şeklinde tanımlı olacaktır. Klasik yöntemlerde ise istemci $\boldsymbol{I_p}$ ulaşmak istediği alt WSN kaynağına ise $\boldsymbol{f_{IS}}$**(r.id)** üzerinden dolaylı veya direk olarak ulaşabilmektedir (RPC, web servis vb.). Burada "$\boldsymbol{f_{IS}}: \tilde{\mathrm{I}}_{\boldsymbol{id}} \rightarrow \boldsymbol{Q^S}$" olarak tanımlıdır. Burada $\boldsymbol{I_p}$ 'nin sistem üzerinden kaynağa ulaşımı bu iki fonksiyonun ardışık yürütümünü sağlayan $\boldsymbol{f_{IW}}(.)$ fonksiyonudur ve şu şekilde gösterilir;

$$
\begin{aligned}
f_{IW}(r) &= g_{SW}(f_{IS}(\mathrm{r.id})\ ) \\
f_{IW}(r) &= g_{SW}\left(q_i^S, |\ q_i^S \in Q^S\right) \\
f_{IW}(r) &= q_j^{I_p}\ \ (q_j^{I_p} \in Q^{I_p})
\end{aligned}
\tag{4.11}
$$

Geleneksel sistemlerde $\boldsymbol{f_{IW}}(.)$ Bir gateway veya RPC teknolojileri ile yürütülmektedir. Burada $\boldsymbol{f_{IW}}(.)$, genellikle bir kaynak değerinin okunması veya set edilmesi gibi temel operasyonları yürütmektedir. Bunun dışında sunucu-tanımlı bazı özel servisleri de yürütebilmektedir. Bilindiği üzere WSN sistemlerden elde edilen ham kaynak verileri belli analiz ve algoritmalardan geçirilerek değerlendirilebilmektedir. Bunun en bilinen örnekleri "anlamlı veri toplama (data aggregation-DA)" ve "veri birleştirme (data fusion-DF)" dir. Bu tip işlemler genellikle aynı ağ içerisindeki ya biraz daha gelişmiş düğümler üzerinde veya merkezi bir bilgisayar üzerinde yürütülmektedir. Ancak IoT WSN sistemlerden faydalanan istemciler farklı WSN 'lerdeki farklı kaynaklardan elde edecekleri verileri, $\boldsymbol{f_{IW}}(.)$ aracılığı ile alarak gerekli DA/DF operasyonlarını kendi taraflarında yürütebilmektedirler. Bunun dışında sağlayıcı tarafta sadece basit veri filtreleme işlemleri gerçekleştirebilmektedirler (örneğin eşik değer, belirli zaman aralıkları için veri değerleri vb.). Bu tip bir IoT-WSN sistemin bir alt WSN kaynağından veri edinim süresi "$\text{Ŧ}_{\boldsymbol{WS}}^{\boldsymbol{i}}$", bu verinin ilgili istemciye iletilmesi "$\text{Ŧ}_{\boldsymbol{SI}}^{\boldsymbol{i}}$" ve ilgili verinin işlenmesi "$\boldsymbol{f_{yrt}}(.) = \text{Ŧ}_{\boldsymbol{DADF}}$" ile gösterilsin. Bu durumda farklı alt WSN 'lerdeki n adet farklı kaynaklardan ayrı ayrı veri talebinde bulunan istemci operasyonu için toplam yürütüm süresi $\text{Ŧ}_{\boldsymbol{y}}$;

$$\text{Ŧ}_{\boldsymbol{y}} = \text{Ŧ}_{\boldsymbol{DADF}} + \sum_{i=1}^{n} \text{Ŧ}_{\boldsymbol{WS}}^{\boldsymbol{i}} + \sum_{i=1}^{n} \text{Ŧ}_{\boldsymbol{SI}}^{\boldsymbol{i}}, \tag{4.12}$$

olacaktır. Dahası bu değerlendirme sonucunda IoT-WSN tarafında tanımlı "m" adet alt WSN aktuatörün set edilmesi gibi dönüş verili senaryolarda ise;

$$\text{Ŧ}_{\boldsymbol{y}} = \text{Ŧ}_{\boldsymbol{DADF}} + \sum_{i=1}^{n} \text{Ŧ}_{\boldsymbol{WS}}^{\boldsymbol{i}} + \sum_{i=1}^{n} \text{Ŧ}_{\boldsymbol{SI}}^{\boldsymbol{i}} + \sum_{j=1}^{m} \text{Ŧ}_{\boldsymbol{IW}}^{\boldsymbol{j}}, \tag{4.13}$$

olacaktır. Burada $\text{Ŧ}_{\boldsymbol{IW}}^{\boldsymbol{j}}$, ilgili aktuatör kaynaklarının toplam yürütüm sürelerini göstermektedir. Tüm kaynakların tek bir alt-WSN sistem üzerinde bulunması durumunda en küçük "$\text{Ŧ}_{\boldsymbol{y}}$" süresi "Düğüm Bazlı Sanallaştırma-DBS" 'de elde edilecektir. Bu durumda $\sum_{i=1}^{n} \text{Ŧ}_{\boldsymbol{SI}}^{\boldsymbol{i}} = 0$ dır ve $\sum_{i=1}^{n} \text{Ŧ}_{\boldsymbol{WS}}^{\boldsymbol{i}}$ ve $\sum_{j=1}^{m} \text{Ŧ}_{\boldsymbol{IW}}^{\boldsymbol{j}}$ değerleri ise minimum değerdedirler. Burada düşük işlem kapasitesinden dolayı büyüyebilecek tek parametre $\text{Ŧ}_{\boldsymbol{DADF}}$ (verinin işlenmesi)'dir fakat bu değerdeki artış çok yüksek olmayacaktır. Öte yandan IoT sistemlerin doğası gereği çok sayıda istemci ve farklı heterojen ağların bir arada bulunması dikkate alındığında DBS (Düğüm bazlı sanallaştırma) verimsizdir. Bunun

yanında WSN düğümler arası etkileşime olanak sağlayan ABS (ağ bazlı sanallaştırma) 'de ise $[\sum_{i=1}^{n} \text{₮}_{\boldsymbol{WS}}^{\boldsymbol{i}} + \sum_{i=1}^{n} \text{₮}_{\boldsymbol{SI}}^{\boldsymbol{i}} + \sum_{j=1}^{m} \text{₮}_{\boldsymbol{IW}}^{\boldsymbol{j}}]_{DBS} < [\sum_{i=1}^{n} \text{₮}_{\boldsymbol{WS}}^{\boldsymbol{i}} + \sum_{i=1}^{n} \text{₮}_{\boldsymbol{SI}}^{\boldsymbol{i}} + \sum_{j=1}^{m} \text{₮}_{\boldsymbol{IW}}^{\boldsymbol{j}}]_{DBS} < [\sum_{i=1}^{n} \text{₮}_{\boldsymbol{WS}}^{\boldsymbol{i}} + \sum_{i=1}^{n} \text{₮}_{\boldsymbol{SI}}^{\boldsymbol{i}} + \sum_{j=1}^{m} \text{₮}_{\boldsymbol{IW}}^{\boldsymbol{j}}]_{ATBS\_API}$ olacaktır. Başka bir değişle bu değerler DBS 'ye göre yüksek, API kullanan bulut sistemlere göre daha düşük olacaktır. Bölüm 2 'de açıklanan dezavantajlarından dolayı ABS 'lar IoT-WSN sistemler için her zaman kullanılamazlar. Bu durumda en etkin IoT-WSN yapısı olan ATBS (Arakatman yazılım temelli bulut sanallaştırma) sistemlerde bu değerlerin düşürülmesidir. ATBS 'de $\text{₮}_{\boldsymbol{WS}}^{\boldsymbol{i}}$ her halükarda geçerli olacağından burada önemli olan $\sum_{i=1}^{n} \text{₮}_{\boldsymbol{SI}}^{\boldsymbol{i}} + \sum_{j=1}^{m} \text{₮}_{\boldsymbol{IW}}^{\boldsymbol{j}}$ 'nin değerinin minimum yapılması olacaktır. Bu ise istemci komut ve operasyonlarının alt WSN sistemlere en yakın noktada yapılmasıyla mümkün olacaktır. Bu tezde önerilen FVWSN framework 'ü ile istemci değerlendirme ve yürütüm işlemi sağlayıcı tarafta gerçekleştirdiğinden, toplam yürütüm süresi $[\sum_{j=1}^{m} \text{₮}_{\boldsymbol{IW}}^{\boldsymbol{j}}]_{FVWSN} < [\sum_{i=1}^{n} \text{₮}_{\boldsymbol{SI}}^{\boldsymbol{i}} + \sum_{j=1}^{m} \text{₮}_{\boldsymbol{IW}}^{\boldsymbol{j}}]_{ATBS\_API}$ olacaktır.

4.3. Alt WSN 'lere ait Çalışma Modeli

WSN 'ler genel olarak iki tip çalışma modunda çalışmaktadırlar [118]. Bunlardan ilki "ayrık veri noktaları çalışma modu (discrete data point)" dur. Bu çalışma modunda ağ kaynakları çok kısa bir süre içerisinde okunur/işlenir ve sistem enerji tasarruf moduna geçer. Enerji verimliliği yüksek olan bir moddur ancak çoklu uygulama çalıştıran sistemler için uygun değildir. Diğer çalışma modu ise ağ kaynaklarının belirli bir süre için erişilebilir olduğu "sürekli çalışma aralığı modu (continuous interval mode)" dur. Bu mod genellikle sorgu/cevap tipi ağlarda kullanılır ve çoklu uygulama çalıştıran sistemler için uygundur. Bu tez doğal olarak bu çalışma modunu içeren WSN sistemleri esas almaktadır. Burada asıl amaç aynı kaynakları aynı zaman aralıklarında kullanan VNd 'lerin sistem tarafından belirlenen süre aşımlarına takılmadan başarılı bir şekilde yürütebilmektedir. Bu nedenle her halükarda $\boldsymbol{U}.\{\boldsymbol{P_u}\} \subseteq \boldsymbol{r_1}.\{\boldsymbol{P_r}\} \cap \boldsymbol{r_2}.\{\boldsymbol{P_r}\} \cap .. \boldsymbol{r_n}.\{\boldsymbol{P_r}\}$ olmak zorundadır. Burada "n" ilgili VNd için kullanılacak olan kaynak sayısını göstermektedir. Bu şartı sağlamayan farklı kaynaklar için ayrı VNd 'lerin oluşturulması gerekmektedir. Oluşturulan VNd için gerekli yürütüm şartı Denklem 4.14 'deki şekilde olacaktır;

$$f_{yrt}(V^{I_p}) + \sum_{i=1}^{n} \text{₮}_{WS}^{i} + \sum_{j=1}^{m} \text{₮}_{IW}^{j} \leq T_{se} \,, \tag{4.14}$$

Burada $\boldsymbol{f_{yrt}(V^{I_p})}$ ilgili VNd 'nin yürütüm süresini vermektedir.

4.4. Kaynak Atama ve Kaynak Enerji İzleme Modülleri

Geleneksel sensör bulut sunucularda kaynak atamada temel kriter istemci parametreleridir. Bu tez kapsamında önerilen sistem iki tip kaynak atama yöntemi ile çalışabilmektedir. Bunlardan ilki geleneksel istemci talebine göre yapılan kaynak atama ikincisi ise hem istemci hem de alt WSN parametrelerine göre yapılan kaynak atama modelidir [123].

İstemci parametre temelli modelde kaynak atama fonksiyonu f_s (.) üç iç içe fonksiyondan oluşmaktadır. Bunlar, $\boldsymbol{f_1}$ **(.),** $\boldsymbol{f_2}$ **(.)** ve $\boldsymbol{f_3}$ **(.)** 'dür. Bunlardan $\boldsymbol{f_1}$ **(.)** uygulama tipi kriterini kontrol etmektedir ve $\boldsymbol{f_1}$ (.) **:** $\mathbf{U_T \rightarrow U_T'}$, $U_T'' \subset U_T \subset R$ 'dir.

$$f_1(U.U_{tip}) = \{r \mid \boldsymbol{r}.\ddot{\mathrm{t}} = U_{tip}\ , \quad r \in U_T\} = \mathrm{U_T}' \tag{4.15}$$

Elde edilen sonuç seti $\boldsymbol{f_2}(.)$ ile çalışma aralığı uyumluluğu bakımından incelenmektedir. Bu alt kümede bulunan kaynakların çalışma aralıkları istemci VNd uygulaması için örtüşmelidir. Tanımı ise $\boldsymbol{f_2}$ (.) **:** $\mathbf{U_T'}$**,** $\rightarrow \in \mathbf{2}^{|\boldsymbol{UT'}|}$ şeklindedir.

$$f_2\big(f_1(U.U_{type})\big) = \{r \mid r \in U_{\mathrm{T}}',\ \mathrm{U_T'} \subset U_T\ ,\ U.\{P_u\} \subseteq r.\{P_r\} = \mathrm{U_T}''\} \tag{4.16}$$

Son adımda ise kaynak durum ve lokasyon bilgilerinin kontrol edildiği $\boldsymbol{f_3}$ **(.)** vardır. Burada sadece aktif olan ve ilgili alan içerisine düşen kaynaklar filtrelenecektir.

$$\begin{aligned} f_S\ (U) &= f_3\Big(\ f_2\big(f_1(U.U_{tip})\big)\Big), \\ &= f_3\big(\ f_2(\mathrm{U_T'})\big), \\ &= f_3(\mathrm{U_T''}), \\ &= \{r \mid \boldsymbol{r} \in \mathrm{U_T''}, \mathrm{U_T''} \subset U_T' \subset R\ , \\ &\qquad r.\ R_{pos} \subset U.U_{alan}\ r.d = 1\} = \mathrm{U_T'''} \end{aligned} \tag{4.17}$$

Bu modelde temel parametre istemcidir. Öte yandan alt WSN 'lerdeki kaynakların ve ağın durumu kaynak atamada hesaba katılmalıdır. Bu özellikle sistem sürdürümü ve alt WSN ağların yaşam ömrü açısından oldukça önemlidir. Şöyle ki; belli sayıda uygulamaya hizmet eden bir kaynağın, çalışma periyodunun çalışma şartlarına göre ayarlanması ilgili kaynağı kullanan düğümün enerji tüketimini optimize edebilir. Baş bir örnek olarak bir WSN ağ kendi iç uygulamasına sahip olabilir ve bunun yanında bir IoT/CPS WSN sisteme entegre olabilir. Bu durumda ağ kaynaklarına gelecek harici taleplerin, ağ trafiği üzerindeki etkisi özellikle ağ içi işleyişin kalitesine doğrudan etkisi olacaktır.

Bu amaçla bu tezde önerilen kaynak atama süreçlerinde ilgili kaynağa ait Ê ve N^{W_k} özellikleri dikkate alınmaktadır. Her ne kadar dikkate alınacak parametre sayısı farklılık gösterebilse de temelde dikkate alınması gereken iki husus enerji tüketimi, işlem yükü ve ağ üzerindeki etkisi olacaktır. Bu nedenle Denklem 4.10'da ki **"G"** değeri önem arz etmektedir. Alt WSN sağlayıcı **"O_i"** ve sağlayıcı sistem arasında belirlenecek olan bir "G_{max}"" değerinin aşılmaması ($G < G_{max}$"") hem hesap yükü hem de ağ trafiği açısından etkin rol oynayacaktır. Bu değer alt WSN kaynak tipi ve kaynak sayısı ile orantılı olabilir veya dinamik olarak ayarlanabilir. Bu ise hem istemci hem de alt WSN arasındaki FVWSN ile mümkün olabilecektir.

İlk kaynak atama modeli gibi bu modelde iç içe fonksiyonlardan oluşmaktadır. $\boldsymbol{f_1}$ **(.)** aynı şekilde uygulama tipine göre filtreleme işlemi yapmaktadır ve $\boldsymbol{f_1}(.)$**:** $\mathbf{U_T \rightarrow U_T'}$**,** $\mathbf{U_T' \subset U_T \subset R}$ 'dir.

$$f_1(U.U_{tip}) = \{r \mid \boldsymbol{r}.\ddot{\mathrm{t}} = U_{tip}\ ,\ \ r \in\ U_T\ \} = \mathrm{U_T}' \tag{4.18}$$

Daha sonra $\boldsymbol{f_2}(.)$ enerji ve veri üretim parametreleri kontrol edilmektedir ve $\boldsymbol{f_2}(.)$**:** $\mathbf{U_T'}$**,** $\rightarrow \in \mathbf{2}^{|\boldsymbol{UT'}|}$ şeklinde tanımlıdır.

$$f_2\left(f_1(U.U_{type})\right) = \{r \mid r\ \in \mathrm{R_T'}, \mathrm{R_T'} \subset\ R_T\ ,\ \ g(E.g_i) \leq G_{max},$$
$$M\left(N_i^{W_k}.E_{d_i}\right) \leq\ \Gamma\ \} = \mathrm{R_T}'' \tag{4.19}$$

Burada " **Γ** " ön tanımlı enerji eşik seviyesini göstermektedir. $g(E.g_i)$ ilgili kaynağın veri üretim oranını verirken, $M\left(N_i^{W_k}.E_{d_i}\right)$ ise ilgili kaynağı barındıran düğümün enerji seviyesini vermektedir. Burada temel nokta ilgili düğümün enerji seviyesinin bilinmesidir.

Eğer bu WSN düğümü içeren alt WSN tarafından enerji seviyesi ile alakalı bir API bulunuyorsa bu değer kolaylıkla elde edilebilir. Bu tip bir API 'nin olmaması durumunda ilgili düğümün enerji seviyesi en kötü şartlara göre tahmin edilebilir. Eğer IoT/CPS WSN sistem alt WSN 'lerin Denklem 4.6-4.8 'deki temel değerleri biliniyorsa (paket alım hassasiyeti, kullanılan paket boyutu, yol kaybı katsayısı, çalışma periyodu vs.) bu tahmini Denklem 4.6 'daki "d_{ih}" değerini hesaplayarak yapabilir. Bu değer ilgili düğümün yönlendirme tablosu sistem tarafından biliniyorsa önceden bellidir. Eğer bilinmiyorsa, bilinen farklı Tx iletim güçlerine, göreDenklem 4.20 'e göre hesaplanabilir.

$$d_{ih}^{wrst} = \arg max_{h \in W, i \neq h} \left(\left\| (R_{pos_i} - R_{pos_h}) \right\| \right) \tag{4.20}$$

Bütün düğüm koordinatları sistem tarafından bilindiğinden Denklem 4.9 'dan hesaplanabilen iletim yarı çapı içine düşen komşu düğümlerden en uzak mesafeli düğüm hesaplanabilir. Bu öklit mesafesidir ve düğümlerin statik olduğu durumlarda geçerlidir.

Son olarak $\boldsymbol{f_3(.)}$ ile birinci modelde olduğu gibi kaynak geçerliliği, konumsal uygunluk ve çalışma zaman aralığı değerlendirilmektedir;

$$\begin{aligned} f_S\,(U) &= f_3\left(f_2\big(f_1(U.U_{tip})\big)\right), \\ &= f_3\big(f_2(\mathrm{U}_\mathrm{T}')\big), \\ &= f_3(\mathrm{U}_\mathrm{T}''), \\ &= \{r \,|\boldsymbol{r} \in \mathrm{U}_\mathrm{T}'', \mathrm{U}_\mathrm{T}'' \subset U_T' \subset R\ , \\ &\quad r.\,R_{pos} \subset U.U_{alan} \wedge U.\{P_u\} \subseteq r.\{P_r\} \wedge r.d = 1\} = \mathrm{U}_\mathrm{T}''' \end{aligned} \tag{4.21}$$

şeklinde ifade edilecektir. Filtrelenmiş kaynaklardan kaç VNd 'nin üretileceği maksimum kaynak değerine yani R_{max}'a bağlıdır.

4.5. Operasyonel Çalışma

Sistemin genel çalışma prensibi istemciler tarafından implement edilen "$\boldsymbol{f_{op_i}(q^{I_p}), (q^{I_p} \in Q^{I_p})}$" fonksiyonunun sistem tarafından yürütülmesidir. Bu ise OperationRunner modülü tarafından sağlanan işlev ($\boldsymbol{V_{op}(.)}$) ile gerçekleştirilmektedir. Bu durumda ilgili VNd yürütümü $\boldsymbol{V_{op}(f_{op_i}(q^{I_p}))}$ olarak ifade edilecektir. Sistem

yürütümünde belli başlı sınırlamalar söz konusudur. Bunlardan ilki bir VNd 'nin yürütüm süresinin ön tanımlı "$\boldsymbol{T_{se}}$" süresini aşmamasıdır. $\boldsymbol{f_{Tse}}(.)$ ilgili VNd 'nin güncel yürütüm süresini vermek üzere bu sınırlama;

$$\boldsymbol{f_{Tse}}(\boldsymbol{V_{op}}(\boldsymbol{f_{op_i}}(\boldsymbol{q^{I_p}}))) \;)<\boldsymbol{T_{se}} \tag{4.22}$$

şeklinde olacaktır. Bunun yanında kullanılan kaynakların veri edinim süreleri de önemlidir. Örneğin aynı anda farklı WSN 'lerde bulunan kaynakları kullanan bir VNd nin veri edinim süresi bu alt ağlardaki gecikmeye bağlıdır. O anda trafik yoğunluğu veya başka bir nedenle bir kaynaktan veri edinim süresinin uzaması $\boldsymbol{T_{se}}$ süresinin aşılmasına yol açacaktır. Bu nedenle hem kaynak okuma/ulaşma sürelerinin istatistiksel durumunun izlenmesi hem de bir kaynak için tekrardan okuma yapılıp yapılmayacağının belirlenmesi için $\boldsymbol{T_r}$ kaynak erişim eşik süresi kullanılır. Bu değer bütün kaynaklar için aynı olmak zorunda değildir ancak belirli bir değerin (ε) altına da inemez. İlgili ağdaki talep oranına göre bu değer her kaynak için optimize olarak ayarlanabilir ve bu $\boldsymbol{T_{ro}}$ olarak ifade edilir. Optimize değer $\boldsymbol{T_{ro}}$ hiçbir zaman $\boldsymbol{T_r}$ den küçük olamaz. Başka bir değişle $\boldsymbol{T_{ro}} \geq \boldsymbol{T_r} \geq \varepsilon$ şartı her zaman geçerlidir. Eğer bir optimize değer atanmamış ise bu durumda $\boldsymbol{T_{ro}} = \boldsymbol{T_r}$ olacaktır. Buna bağlı olarak bir kaynağın bir VNd 'nin yürütümü esnasında kaç kez okunabileceği de hesaplanabilir. Bu durumda $\boldsymbol{T_{ro}}$ ve $\boldsymbol{T_r}$ değerleri önemlidir. Bir kaynağın okuma sayısı $\boldsymbol{C'_{sr}}$,

$$\boldsymbol{C'_{sr}} = \frac{\boldsymbol{T_{ro}}}{\boldsymbol{T_r}} \tag{4.23}$$

olacaktır. Öte yandan bu değerin çok fazla olması durumunda ise hem alt WSN işlem trafiği artacak hem de sistem performansını olumsuz etkileyecektir. Bununla birlikte bir VNd 'nin çok fazla sayıda kaynağa sahip olması da sistem performansını olumsuz etkileyebilir. Bu sebeple iki önemli kısıt sistemde tanımlıdır. Bunlardan ilki bir kaynağın tekrardan okuma değerinin sınırlandırılması ikincisi ise bir VNd 'nin kullanabileceği maksimum kaynak sayısıdır. $\mathrm{M_{Rm}}(.)$ ilgili VNd 'nin okuduğu kaynak sayısını ve $\mathrm{C_R}(.)$ bir kaynağın okuma sayısını veren modül fonksiyonları olmak üzere, söz konusu bu şartlar;

$$\mathrm{C_R}\left(\mathrm{V_{op}}\left(\mathrm{f_{op_i}}\left(\mathrm{q^{I_p}}\right)\right)\right) \leq \mathrm{C'_{sr}} \leq \mathrm{C_{sr_{max}}} \tag{4.24}$$

$$\mathrm{M_R}\ (\mathrm{V_{op}}\left(\mathrm{f_{op_i}}(\mathrm{q^{I_p}})\right)) \leq R_{max} \tag{4.25}$$

Öte yandan sisteme bağlanacak olan bireysel abone veya abone WSN 'lerdeki düğüm sayılarının dinamik olarak değişebileceği, alt WSN 'lerden gelecek cevap sürelerinin ise farklı talep ortamlarında değişken olacağı dikkate alındığında, görev kuyruğu ve doğal olarak sistem, FIFO temelli çalışan sonlu geliş kaynağına sahip M/M/1/N (poisson giriş/exponential servis/tekli kuyruk yapısı/N kapasiteli) tipi davranış gösterecektir [123]. Bu davranış istemci talep oranı ve verilen hizmet süresi oranlarını daha bir önemli kılmaktadır. Bilindiği üzere M/M/1/N kuyruk içeren sistemlerde ortalama talep gelme hızı "**λ**" 'nın, birim zamandaki ortalama servis hızı "**μ**" 'e oranı bir ve üzeri olması durumunda sistemdeki gecikme artacaktır. Sistemdeki güvenlik ve hafıza gibi diğer kısıtlarda dikkate alındığında sistem tarafından atanan maksimum girdi sayısı "$\mathrm{N_{max}}$" sistem TaskQueue modülündeki zaman çerçevesi değerini belirleyecektir. Sistemdeki sanal görevlerin akış modeli bu nedenle ayrı bir önem kazanmaktadır.

Kuyruk teorisinden bilindiği üzere bir sanal görev talebinin kuyrukta olma ihtimali,

$$\mathrm{Sg_i} = \left[\frac{\lambda}{\mu}\right] Sg_0 \tag{4.26}$$

ile hesaplanır. Benzer şekilde $\mathrm{N_{max}}$ sanal görev talebinin önerilen sistemde olması olasılığı ise,

$$\mathrm{Sg_{N_{max}}} = \left[\frac{\lambda}{\mu}\right]^{\mathrm{N_{max}}} Sg_0 \tag{4.27}$$

şeklinde olacaktır. Sistemde herhangi bir anda bulunan "N" adet sanal görev taleplerinin olasılık toplamının 1 olması gerektiğinden,

$$\sum_{n=0}^{N} \mathrm{Sg_N} = 1 \tag{4.28}$$

$$\sum_{n=0}^{N} \left[\frac{\lambda}{\mu}\right]^{\mathrm{n}} Sg_0 = 1 \tag{4.29}$$

şeklinde ifade edilebilir. Bu durumda sistemde herhangi bir sanal görev talebinin bulunmama olasılığı,

$$Sg_0 = \frac{1}{\sum_{n=0}^{N}\left[\frac{\lambda}{\mu}\right]^{\mathrm{n}}} \tag{4.30}$$

olarak bulunabilir. Burada $\sum_{n=0}^{N}\left[\frac{\lambda}{\mu}\right]^{\mathrm{n}}$ bir geometrik seri olduğundan, sistemde herhangi bir sanal görev bulunmaması,

$$Sg_0 = \frac{1}{\frac{1-\left[\frac{\lambda}{\mu}\right]^{N+1}}{1-\left[\frac{\lambda}{\mu}\right]}} \tag{4.31}$$

$$Sg_0 = \frac{1-\left[\frac{\lambda}{\mu}\right]}{1-\left[\frac{\lambda}{\mu}\right]^{N+1}} \tag{4.32}$$

olarak yazılabilir. Bu değerlerin bilinmesi ile birlikte sistemdeki ortalama sanal görev sayısı ($O_{sg}{}^{s}$) aşağıdaki çıkarımla elde edilebilir,

$$O_{sg}{}^{s} = \sum_{n=0}^{N} \mathrm{n}\,(Sg_N) \tag{4.33}$$

$$O_{sg}{}^{s} = Sg_0 \sum_{n=0}^{N} \mathrm{n}\left[\frac{\lambda}{\mu}\right]^{n} \tag{4.34}$$

$$O_{sg}{}^{s} = \left[\frac{1-\left[\frac{\lambda}{\mu}\right]}{1-\left[\frac{\lambda}{\mu}\right]^{N+1}}\right]\left(\left[\frac{\lambda}{\mu}\right]\right)\left[\sum_{n=0}^{N}\frac{\partial}{\partial\left[\frac{\lambda}{\mu}\right]}\left(\left[\frac{\lambda}{\mu}\right]^{n}\right)\right] \tag{4.35}$$

$$O_{sg}{}^{s} = \left[\frac{\lambda}{\mu}\right]\left[\frac{1-(N+1)\left[\frac{\lambda}{\mu}\right]^{N}+N\left[\frac{\lambda}{\mu}\right]^{N+1}}{(1-\left[\frac{\lambda}{\mu}\right])(1-\left[\frac{\lambda}{\mu}\right]^{N+1})}\right]\left(\left[\frac{\lambda}{\mu}\right] \neq 1\ şartı\ ile\right) \tag{4.36}$$

Bununla beraber sadece kuyrukta bulunacak olan sanal görev sayısı ($O_{sg}{}^{k}$) ise genel kuyruk teoreminden bilindiği üzere,

$$O_{sg}{}^{k} = O_{sg}{}^{s} - (1 - \frac{1-\left[\frac{\lambda}{\mu}\right]}{1-\left[\frac{\lambda}{\mu}\right]^{N+1}}) \quad (4.37)$$

ifadesi ile hesaplanabilir. Her bir sanal görev talebinin sistemde ortalama bekleme süresi ($B_{sg}{}^{s}$) ve kuyrukta bekleme süresi ($B_{sg}{}^{k}$) Denklem 4.38 ve 4.39 'e göre hesaplanır,

$$B_{sg}{}^{s} = \frac{O_{sg}{}^{k}}{\lambda(1-\mathrm{Sg_N})} + \frac{1}{\mu} \quad (4.38)$$

$$B_{sg}{}^{k} = B_{sg}{}^{s} - \frac{1}{\mu} \quad (4.39)$$

Burada servis veren birim (SerOP) olup hizmet süresi doğal olarak $\boldsymbol{T_{ro}}$ 'dan küçük veya eşit olmak zorundadır.

4.6. Enerji Tüketimi ve Gecikme

Sistemin performans kriterleri arasında sistem gecikmesi ve kullanıcı sayısına göre WSN düğümlerdeki enerji sarfiyatı oldukça önemlidir. Sistemdeki gecikme ifadesi;

$$\mathrm{D_T} = \mathrm{D_i} + \mathrm{D_p} + \mathrm{D_E} \quad (4.40)$$

$$\mathrm{D_p} = \mathrm{D_{sp}} + f_{Tse}(V_{op}(f_{op_i}(q^{I_p}))) \ ) \quad (4.41)$$

Burada $\mathbf{D_T}$ toplam gecikmeyi, $\mathbf{D_i}$ istemci ile sistem arasındaki gecikmeyi göstermektedir. Denklem 4.41 'da verilen $\mathbf{D_p}$ ifadesi bir VNd 'nin yürütüm süresini temsil etmektedir. Bu ifade aynı zamanda sanal görev oluşturma ($\mathrm{D_{sp}}$) ve kuyrukta bekleme sürelerini de içerdiğinden $f_{Tse}(V_{op}(f_{op_i}(q^{I_p}))) \ ) > \boldsymbol{T_{se}}$ olması durumunda toplam sistem gecikmesi;

$$\mathrm{D_T} = \mathrm{D_i} + \mathrm{D_{sp}} + T_{se} \quad (4.42)$$

Olarak ifade edilecek ve yürütüm başarısız olarak etiketlenecektir.

Bununla birlikte yürütüm süresi ise kaynak okuma gecikmesi (D_f) ve iç yürütüm sürelerinin (D'_E) toplamına eşittir;

$$D_E = D'_E + D_f \tag{4.43}$$

$$D_f = \sum_{i=1}^{R_N} F_{Tr}(f_{op_i}(q^{I_p}))) \ ) \tag{4.44}$$

D'_E İlgili VNd nin anlık yürütüm süresini verirken D_f toplam kaynak gecikmesini temsil etmektedir. Burada $\mathbf{D_f} > \boldsymbol{T_{ro}} \geq \boldsymbol{T_r}$ olması durumunda ise toplam gecikme Denklem 4.45 'deki gibi olacaktır;

$$D_T = D_i + D_{sp} + D'_E + T_{ro} \tag{4.45}$$

Diğer önemli bir parametre düğümlerin yaşam ömürleridir. Bir düğümün başlangıç enerjisi genel olarak joule olarak ifade edilir ve bir saniye içinde harcanan güç değerini göstermektedir. Bir düğümün yaşam ömrü bu tez kapsamında gün olarak hesaplanacaktır. Düğüm yaşam ömrünün hesaplanabilmesi için iletim (tx), alım (rx), algılama (sense), yürütüm (proc) ve uyku (sleep) modlarındaki enerji sarfiyatları bilinmelidir. Bunlarla ilgili genel ifadeler şu şekildedir;

$$E_{tx} = \frac{I_{tx}\, x\, t_{tx}}{R_{ptx}}\, x\, \hat{Z}, \tag{4.46}$$

$$E_{rx} = \frac{I_{rx}\, x\, t_{rx}}{R_{prx}}\, x\, \hat{Z}, \tag{4.47}$$

$$E_{sense} = \frac{I_{sense}\, x\, t_{sense}}{R_{psense}}\, x\, \hat{Z}, \tag{4.48}$$

$$E_{proc} = \frac{I_{proc}\, x\, t_{proc}}{R_{pproc}}\, x\, \hat{Z}, \tag{4.49}$$

$$E_{sleep} = \frac{I_{sleep}\, x\, t_{sleep}}{R_{psleep}}\, x\, \hat{Z}, \tag{4.50}$$

$$\hat{Z} = K\ x\ V_{cc}, \tag{4.51}$$

Burada "V_{cc}", besleme gerilimini, "I" ilgili süreç akımını, "t" ilgili sürecin süresini ve R_p süreç frekansını vermektedir. "K" değeri yaşam ömrü birimine göre değişmekte olup, gün için $8.64\ x\ 10^4$ 'dir.

Bir düğümün yaşam ömrü Denklem 4.52 'ya göre hesaplanabilir;

$$LT_i = \frac{E_{init} - E_{thr}}{(E_{tx} + E_{rx} + E_{sens} + E_{proc}) + E_{sleep}} \tag{4.52}$$

Bu denklemde $E_{init} = V_s\,(volt)x\ I_{ah}\ (mAh)x\ 3600$ watt/sn olarak hesaplanır. E_{thr} ise düğümün yaşam ömrünü gösteren eşik değeri temsil etmektedir.

4.7. Hata Tolerans

Sistem yürütümünde VNd bazlı hata, VEngine bazlı hata ve alt WSN bazlı hatalara karşı farklı önlemler alınmıştır. VNd bazlı hatalarda hata sayısı ve güvenlik bazlı sınırlar dikkate alınmaktadır. Bir VNd 'nin Denklem 4.22-4.25 'deki şartların aşım sayısı ön tanımlı "$H^{V^{I_p}}$" değeri ile kontrol edilmektedir. Her bir şartın aşımında ilgili VNd için bu değer bir azaltılır. Bu değerin 0 olması ile VNd yürütümü durdurulur ve istemci bilgilendirilir.

$$VNd\ durum \begin{cases} 1, eğer\ H^{V^{I_p}} > 0 \\ 0, eğer\ H^{V^{I_p}} \leq 0 \end{cases} \tag{4.53}$$

VEngine bazlı hatalarda sistem OperationRunner 'da çalıştırılan her VNd ve kuyruğa alınan sanal görev için yürütüm zamanı ve kuyruk sırasını da içeren bir replikasyon verisinin kaydını başka bir sanal makinada tutmaktadır. VEngine çökme durumunda bu replikasyon verileri ile sistem yeniden başlatılmaktadır. Bu şekilde, OperationRunner ve sanal görev replikasyon formatları $Rep_{\boldsymbol{V^{I_p}}} =< \boldsymbol{V^{I_p}}, t_{\boldsymbol{V^{I_p}}} >$ ve $Rep_{\boldsymbol{V^{I_p}}} =< Sg_{\boldsymbol{V^{I_p}}}, t_{Sg_{\boldsymbol{V^{I_p}}}} >$ şeklinde olacaktır. Alt WSN ler için [63] 'deki model esas alınmıştır. Bu hata tolerans modeline göre bir alt WSN 'deki çalışan düğüm sayısı,

$$H_t = H_{t-1} - Ol_{W_k} x\, H_{t-1}\,, H_0 = |W_k{}^0|\,, \tag{4.54}$$

ile ifade edilir. Burada $|W_k{}^0|$ alt WSN ağın başlangıçtaki düğüm sayısını, Ol_{W_k} ise ilgili ağdaki hata olma olasılığını ifade etmektedir. Simülasyon esnasında hata oluşum oranı sabit alınmış ve aynı bölge için alternatif bir düğüm ölçüm yaptığı varsayılmıştır. Bu nedenle alt WSN ağlardan gelen hatalar Denklem 4.22-4.25 'de tanımlanan sonuçlara göre değerlendirilecektir.

Denklem 4.11-4.13 'de tanımı verilen problemin ispatı tezin bir sonraki OPNET ile sistem simülasyonları bölümünde denenmiştir. Bu simülasyonda Denklem 4.14 'de belirtilen yürütüm şartları dikkate alınmıştır. Kayıt prosedürlerinde, Denklem 4.15-4.17temeline dayalı ancak fazladan alt WSN parametrelerini de dikkate alan Denklem 4.18-4.21 modeli ve yardımcı enerji takip modelleri olan Denklem 4.6-4.10 modelleri OPNET platformunun "registration" modülünde kodlanmıştır. Bölüm 3.1.2 'de "OperationRunner", "SerOp" 'modüllerinde detayları verilen operasyonel çalışma şekli OPNET 'de Denklem 4.22-4.25 'e göre gerçekleştirilmiştir. Sistemde kullanılan M/M/1/N kuyruk yapısının çalışma modeli Denklem 4.26-4.39 'de verilmiştir. Simülasyonda kullanılan düğmelerde enerji sarfiyat ve yaşam ömürlerini ifade eden Denklem 4.40-52 modellerine göre gerçeğe yakın düğüm enerji tüketimi simülasyonlarda izlenmiştir. Son olarak Denklem 4.53 ve 4.54 hata tolerans yöntemlerinin esas kriterlerini oluşturmaktadır.

5. SİSTEM SİMÜLASYONU VE FVWSN YAZILIMININ FİZİKSEL ORTAMDA TEST EDİLMESİ

Bu bölüm, önerilen sistemin daha önceki bölümlerde açıklanan temel prensiplerine göre simülasyonunu ve FVWSN framework 'ünün pratikte test edilmesini ele almaktadır. Bilindiği üzere bir ağ sistemin incelenmesi dört şekilde gerçekleştirilebilir. Birincisi "Matematiksel Modelleme" ikincisi ise simülasyon platformlarının kullanılması, üçüncüsü Hibrid analiz ve dördüncüsü test-bed emülasyonudur [124]. Matematiksel modelleme hızlıdır ancak pek çok durumda yetersiz kalmaktadır. Birçok farklı durum için analitik modeller mevcut olmayabilir. Dahası, var olan modeller gerçeğe yakın yaklaşımlar sunmakta zayıf kalabilir. Örnek vermek gerekirse özellikle ağ kuyruk yapılarında kullanılan matematiksel modellerin kayıplı modeller olması gösterilebilir. Simülasyon platformları ise kullandıkları simülasyon tekniklerine göre temel olarak iki tiptir. Bunlar, TCS (Time Clock Simulation) ve DES (Discrete Event Simulation). Bunlardan TCS, basit yapılı ağ simülasyonları için kullanılan ve zaman slotları ile çalışan yavaş platformlardır. DES ise büyük ölçekli simülasyonlarda oldukça başarılı sonuç veren ve çok daha hızlı olan yaygın bir simülasyon tekniğidir. Bununla beraber güçlü bilgisayar donanımlarına ihtiyaç duyar. Üçüncü ağ sistem inceleme yöntemi ise matematiksel model ve simülasyon platformlarının beraber kullanılmasıdır ki bu daha çok spesifik durumlarda gerekli olmaktadır. Son olarak test-bed ortamlarında yapılan analizler ise özellikle büyük ölçekli sistemlerin incelenmesinde maliyet ve zaman açısından elverişsizdir.

Bu nedenlerden dolayı simülasyon platformu olarak, güçlü özelliklere sahip DES tabanlı OPNET (Optimized Network Engineering Tools) Modeler (yeni adıyla Riverbed Modeler) platformu kullanılmıştır [33]. Bu tez çalışması, ticari lisanslı olan bu simülatörün eğitim lisanslı deneme sürümünde gerçekleştirilmiştir. Bu simülasyon platformun seçilmesindeki neden, bu platformda WSN düğümlerinin katmansal yapılarının gerçeğe daha yakın olarak modellenebilmesidir. Böylece farklı teknolojik özellikleri olan WSN sistemlerin tesis edilmesi kolay olabilecektir. OPNET Modeler platformu diğer ağ teknolojilerine sağladığı alt yapı desteğini WSN sistemlere sunmamaktadır. Bu nedenle özgün WSN sistemlerinin oluşturulması kullanıcılara kalmıştır. OPNET Modeler simülatörünün sağlamış olduğu temel simülasyon bileşenleri, düğüm(node), süreç(process), iletişim hattı (link), paket yapısı (packet format) ile her bir ağ bileşeni tasarlanabilmektedir. Programlama dili olarak ProtoC dilini kullanmaktadır. Bu tez çalışmasının odaklanmış olduğu heterojen WSN yapılarının

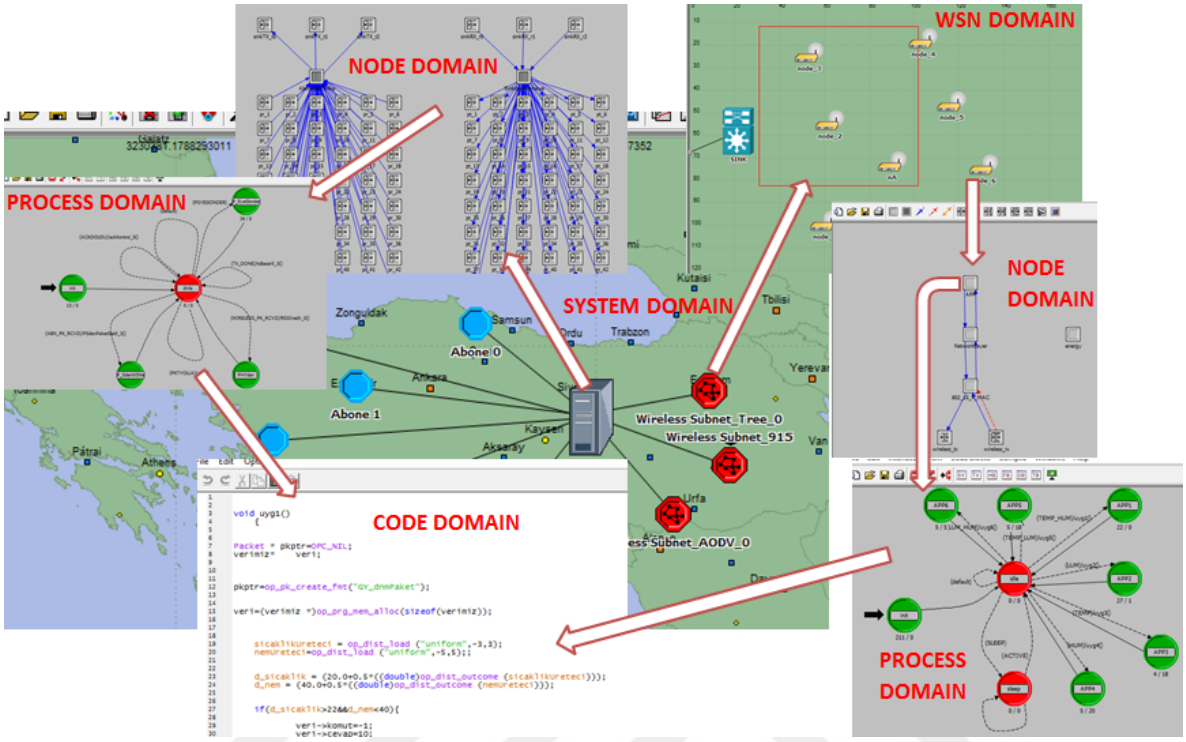
oluşturulması için öncelikle WSN düğüm, sink düğüm yapıları ve diğer sistem bileşenleri Bölüm 4 'deki modellemelere göre OPNET ortamında programlanmış ve genel ağ yapıları bu simülatör bileşenleri vasıtasıyla oluşturulmuştur. Bu süreç ve detaylı bilgi EK-1 de verilmiştir. Bu bölümde doğrudan sisteme ait simülasyon platformları açıklanmıştır.

Bu bölümde, temel amaç sadece önerilen IoT/CPS WSN sitemin benzetiminin yapılması değil, DBS (Düğüm bazlı sanallaştırma) ve ABS (Ağ Bazlı Sanallaştırma) gibi diğer WSN sanallaştırma teknikleri ile karşılaştırılmasının da aynı ortam ve şartlar için yapılabilmesidir. Bu nedenle ilk olarak DBS sistemlerin analizi gerçekleştirilmiştir.

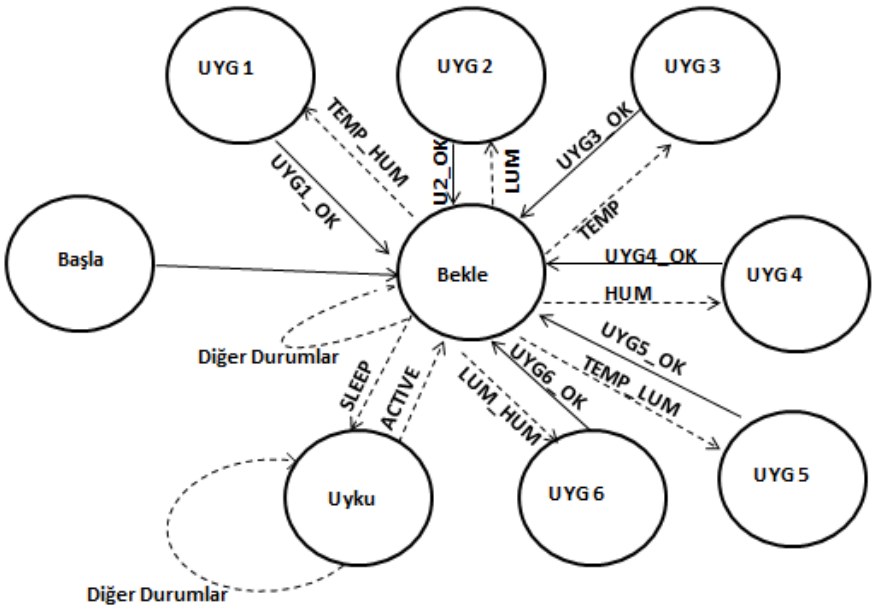
5.1. DBS temelli IoT Kaynak Paylaşımı Simülasyonu

Önceki bölümlerde bahsedildiği üzere DBS sistemler bir düğüm üzerinde birden fazla istemci uygulamanın koşturulması prensibine dayanmaktadır. Bu teknikte değerlendirme ve karar verme noktası WSN içi düğümlerdir. Bu teknik EOS 'a bağlı olarak üç şekilde gerçekleştirilebilir. Bunlar ardışıl (sequential), olay tabanlı (event-driven) ve çoklu iş parçacık temelli (multithread). Bunlardan üçüncüsü kısıtlı donanımsal özelliklerden dolayı çoğu düğüm teknolojisi tarafından tercih edilmez. En çok tercih edilen event-driven temelli beraber çalışabilirliktir. Bu deneyde hem sequential hem de event-driven temelli kaynak paylaşımı senaryoları denenmiştir. Bunun için ilk olarak geliştirilmiş olan WSN düğümlerinin uygulama katmanları yeniden dizayn edilmiştir. [64] 'daki kısıtlar dikkate alınarak en fazla altı farklı uygulama için simülasyonlar gerçekleştirilmiştir. Düğüm uygulama katmanın çalışmasını gösteren Moore tipi sonlu durum makinesi (FSM) yapısı aşağıda verilmiştir. Bu simülasyonda WSN düğümlerin üzerinde üç adet sensör (ısı, sıcaklık, nem) ve bir aktuatörün olduğu varsayılmıştır. Sensör değerleri normal dağılıma göre belli değerler üreten kaynaklardır. Her bir uygulama bu sensör çiftelerinden ikisinin belirlenen olay şartlarını sağlaması durumunda ilgili olay durumuna geçiş sağlanacaktır. Örneğin sıcaklığı 40 dereceyi nemin % 20 'nin altına inmesi durumunda "UYG1" durumuna geçiş yapılması gösterilebilir. Her bir uygulama O(n^2) karmaşıklığına sahip bir skaler uygulama kodu çalıştırmaktadır. FSM diyagramlarındaki kesikli çizgiler, senaryo olay şartlarını ifade eden geçişleri, düz çizgiler ise ilgili durum (state) kodlarının yürütümlerinin tamamlanmış olması şartlarındaki geçişleri temsil etmektedir. Kendi üzerine dönen geçişler "Diğer Durumlar " olarak etiketlenmiştir. Bunun anlamı senaryo olay şartlarının ve ilgili geçiş şartları dışında meydana gelebilecek her türlü şartlarda durum değişiminin yaşanmayacağıdır. Örneğin sistem en düşük enerji modunda ise olay tetiklenmesi gelse bile uyku modundan çıkılmaması gibi. Şekil 5.1 OPNET platformunda hazırlanan DBS

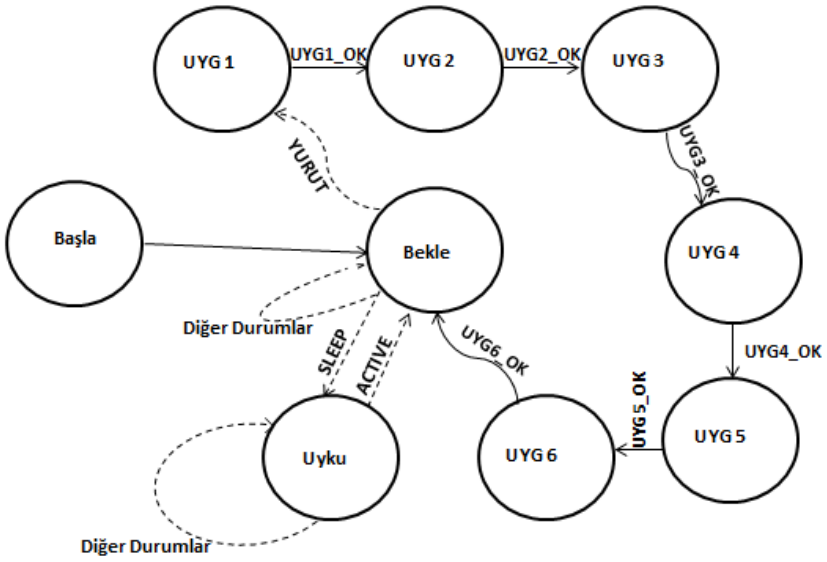
simülasyonu platformunun genel yapısını ve bileşenlerini, Şekil 5.2 ise WSN düğümlerin event-driven ve sequential uygulama katman çalışma FSM ‘lerini göstermektedir.



Şekil 5.1.DBS genel simülasyon yapısı ve bileşenleri



a) Event-Driven Mod



b) Sequential Mod

Şekil 5.2. DBS simülasyonları için WSN düğüm uygulama katman çalışma FSM 'leri

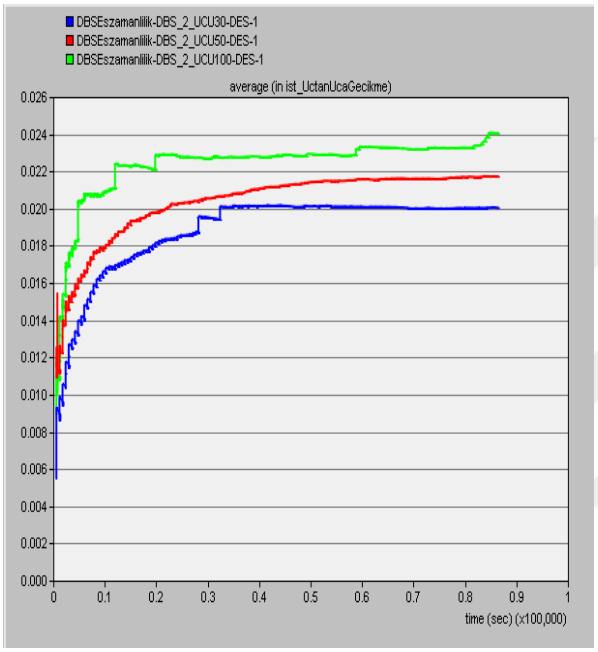
Event-driven DBS simülasyonlarında durum geçiş şartları ayrık, yani aynı anda birden fazla şartı sağlamayacak şekilde seçilmiştir. DBS simülasyonları genel olarak sistemi üç ana parçadan oluşmaktadır. Bunlar aboneler, Pub/Sub MW sistem ve alt WSN sistemlerdir. DBS simülasyonlarında alt WSN sistemlerdeki her bir düğümün uygulama katmanında her aboneye ait state 'ler (durumlar) tanımlıdır ve bunlar seçilen çalışma moduna göre koşturulmaktadır. Senaryo gereği her uygulama değerlendirme sonucunda ağdaki başka bir düğümdeki aktuatörü aktif/pasif edecektir. PubSub MW sistem, abonelere kendi uygulamalarının koşturulması sonucunu veren arakatman yazılımını içeren sistemdir. Bu deneyde abonelerin sisteme bir etkileri olmayıp sadece sonuç değerlerinin alınması ve istatistiklerinin tutulmasını sağlamaktadır. Sistem performans metriklerinden olan "gecikme süresi değeri", bir abonenin ilgili kod yürütümlerinin tamamlanması ve aktuatörün güncellenmesi için geçen süreyi göstermektedir. İkinci performans metriği ise düğümlerin kalan enerji seviyeleridir. Üçüncü ve son performans metriği ise her bir alt WSN 'deki ağ çıktısı (throughput) 'dur. Tablo 5.1 DBS simülasyon parametrelerini göstermektedir. Bu ve bundan sonraki ABS ve FVWSN sistem simülasyonlarında sonuçların değerlendirilmesine ve alt WSN 'deki çalışma mantığı hakkında fikir vermesi açısından sondaj usulü seçilen alt WSN düğüm istatistikleri verilecek daha sonra ise tüm sistemin ortalama değerlerini gösteren grafik veriler sunulacaktır.

Tablo 5.1. DBS Simülasyonları için kullanılan parametreler

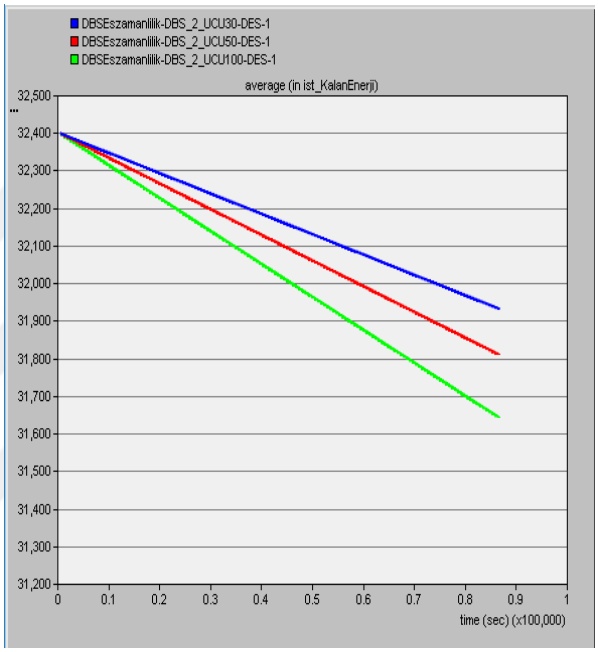
Uygulama Katmanı	
Uygulama Sayısı	2 - 6
Eş zamanlı olay tetiklenme aralığı (sn)	0.05 – 1.5
Normal WSN ağ çalışma periyodu (sn)	600
Uygulama Analiz Tipi	O(n^2) karmaşıklığına sahip skaler korelasyon
Uygulama Yönetim	WSN1 / WSN3= ZigBee Private Profile WSN2 = String tabanlı komut/sorgu
Ağ Katmanı	
Yönlendirme Tipleri	Ağaç / MESH
MAC Katmanı	
MAC tipi	802.15.4/Yarışma Temelli
Ortama Erişim	CSMA/CA
ACK	Evet
Min. Backoff Exponent	3
Max. Backoff Sayısı	4
Fiziksel Katman	
WSN 'lerin Çalışma Frekans Bandları (MHz)	868 / 915 / 2400
Kanal Sayısı	1-16
Band genişliği (KHz)	800 - 2000
Modülasyon	BPSK/QPSK
Spread Spectrum	DSSS
Veri Hızı (Kbps)	250
Enerji	
Besleme Tipi	Batarya
Başlangıç Seviyesi (Joule)	32400
Fiziksel Ortam	
WSN çalışma alanı ebatları (m x m)	150 x 150
Toplam Alt WSN saysı	3
Düğüm sayıları	10 / 10 /10
Ortam tipi	Kampüs Açık Alan

Simülasyonda her bir WSN aynı zamanda kendine ait bir periyodik çalışma da yapmaktadır. Bu periyodik çalışma değeri düğümlere ait atanmış attribute (öznitelik) 'lar vasıtası ile ayarlanabilmektedir. Simülasyon farklı senaryolar için ayrı ayrı koşturulmuştur.

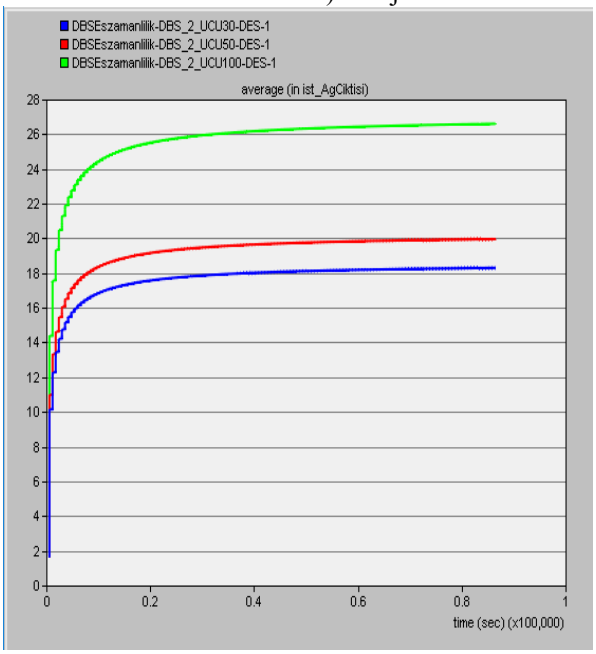
Event-driven senaryolarda üç temel parametre kullanılmıştır. Bunlar olayların birbirine ne kadar yakın zamanda gerçekleştiği, bu olaylardan maksimum sayıda olayın tetiklenmesi durumunda sistemin nasıl etkilediği, ikincisi yüklü uygulama sayısının etkisi ve üçüncüsü ise ağ kaynaklarının yüzde kaçının paylaşıldığıdır. Örnek bir senaryo olarak düğüm üzerinde 4 uygulama, ağ üzerinde %30 kaynak paylaşımı ve farklı uygulamaları tetikleyen olay gerçekleşmelerinin 0.05 sn içinde oluşması verilebilir. Aşağıda şekillerde ilk olarak alt WSN ve düğümlerinin davranışları hakkında fikir sağlanması için seçilen düğüm istatistik grafikleri verilmiştir.



a) Uçtan uca gecikme



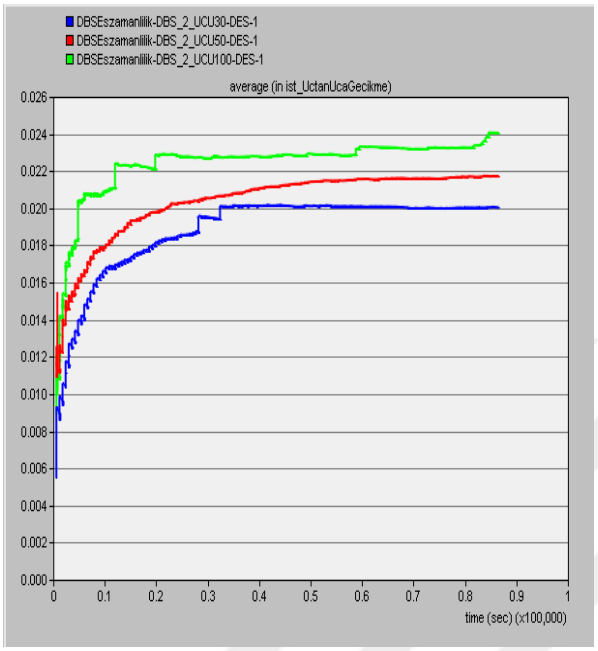
b) Enerji Tüketimi



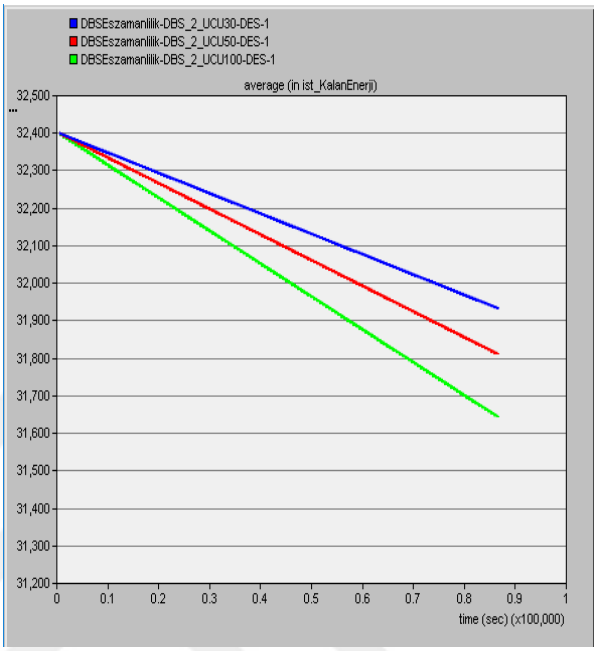
c) Ağ çıktısı (bit/sn)

Şekil 5.3. Uyg.Say: 2, Kay.Pay(%): 30 (mavi),50 (kırmızı),100 (yeşil), Olay Ara.:1.5 sn

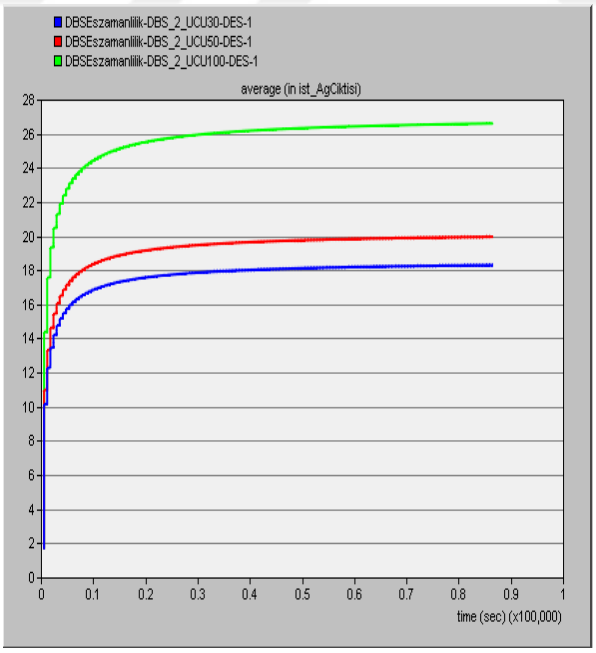
Şekil 5.3 ‘de, düğüm başına 2 uygulamanın yüklü olduğu ve bu uygulamaları süren olayların olma olasılığının 1.5 sn içerisinde gerçekleştiği senaryonun, alt WSN ağ kaynaklarının %30 (mavi), 50 (kırmızı) ve 100 (yeşil) paylaşılması durumunda, sondaj usulü seçilen bir alt WSN düğümü için elde edilen simülasyon performans sonuçları verilmiştir.



a) Uçtan uca gecikme



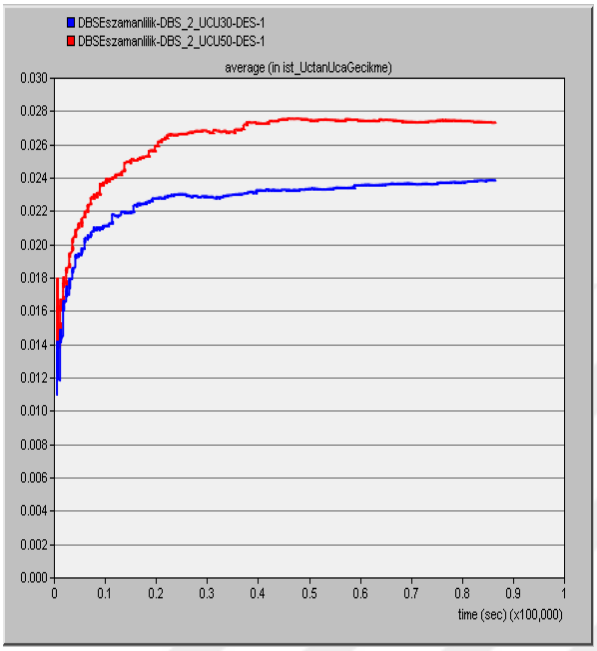
b) Enerji Tüketimi



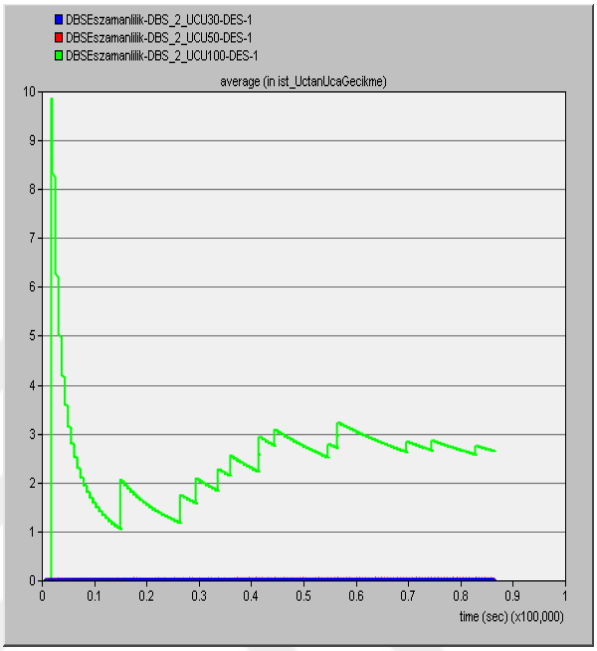
c) Ağ çıktısı (bit/sn)

Şekil 5.4. Uyg.Say: 2, Kay.Pay(%): 30 (mavi),50 (kırmızı),100 (yeşil), Olay Ara.:1 sn

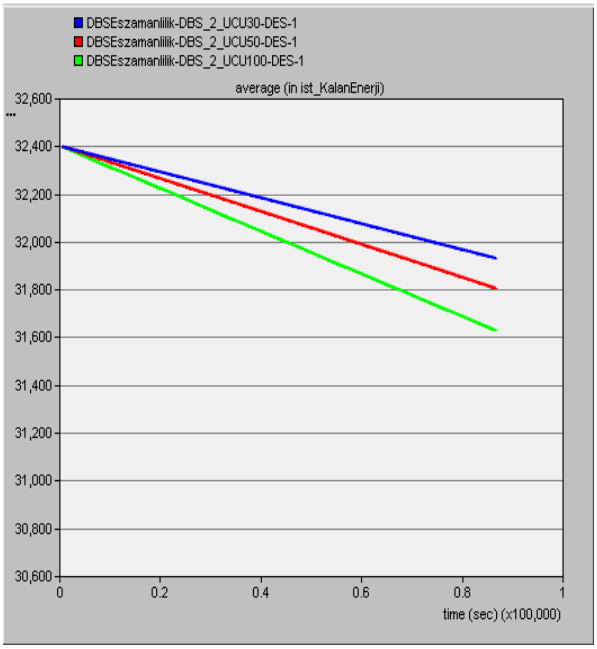
Şekil 5.4 ‘de, düğüm başına 2 uygulamanın yüklü olduğu ve bu uygulamaları süren olayların olma olasılığının 1 sn içerisinde gerçekleştiği senaryonun, alt WSN ağ kaynaklarının %30 (mavi), 50 (kırmızı) ve 100 (yeşil) paylaşılması durumunda, sondaj usulü seçilen bir alt WSN düğümü için elde edilen simülasyon performans sonuçları verilmiştir. Görüleceği üzere, bu olay olma olasığı aralık değeri için Şekil 5.3 ile büyük bir fark gözlemlenmemiştir.



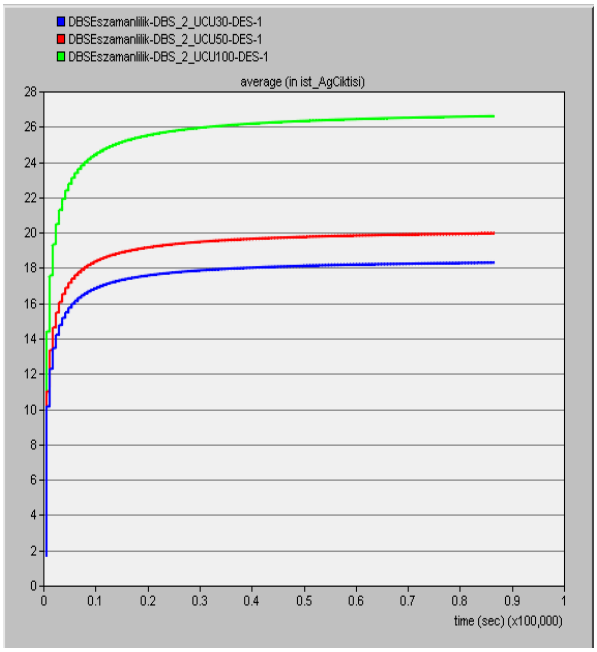
a) %30 ve %50 kaynak paylaşımı gecikme



b) %100 kaynak paylaşımı gecikme



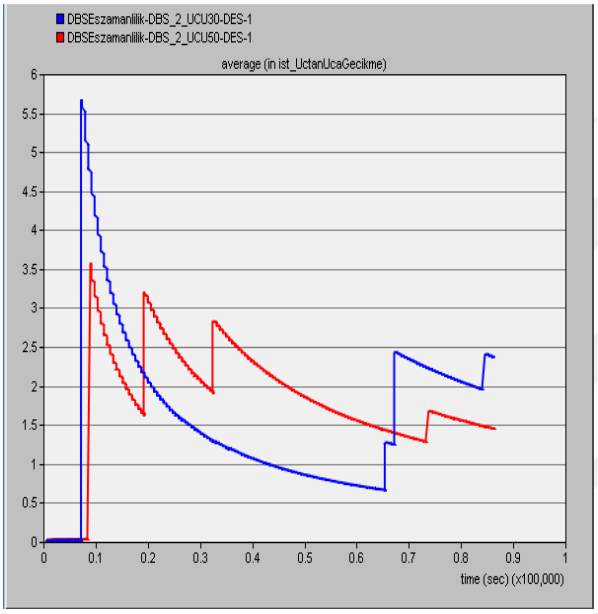
c) Enerji tüketimi



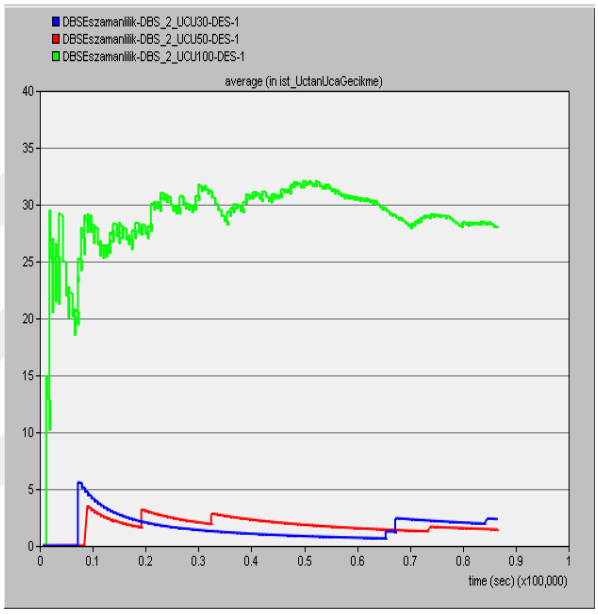
d) Ağ çıktısı (bit/sn)

Şekil 5.5. Uyg.Say: 2, Kay.Pay(%): 30 (mavi),50 (kırmızı),100 (yeşil), Olay Ara.:0.15 sn

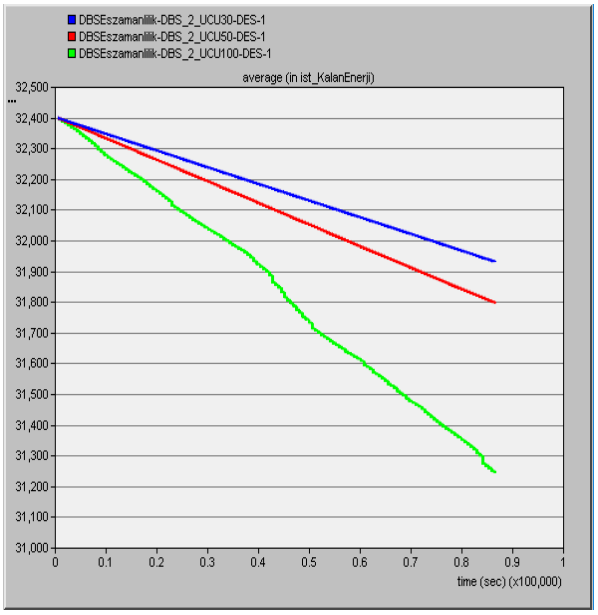
Şekil 5.5 ‘de, düğüm başına 2 uygulamanın yüklü olduğu ve bu uygulamaları süren olayların olma olasılığının 0.15 sn içerisinde gerçekleştiği senaryonun, alt WSN ağ kaynaklarının %30 (mavi), 50 (kırmızı) ve 100 (yeşil) paylaşılması durumunda, sondaj usulü seçilen bir alt WSN düğümü için elde edilen simülasyon performans sonuçları verilmiştir. Şekil 5.5 b ‘de görüldüğü üzere bu senaryoda %100 kaynak paylaşımı durumunda, ağdaki uçtan uca gecikme değeri Şekil 5.5 a’ da gösterilen %30 ve 50 değerlerine kıyasla oldukça yüksektir. Tezde bu değer Kritik Olay Olma Süresi – KOOS- olarak adlandırılmıştır. Uçtan uca gecikme değerin %100 kaynak paylaşımı için yüksek olma nedeni, ağ genelinde detayları bu tez kapsamında sunulmayan WSN iç dinamiklerinden kaynaklanmaktadır.



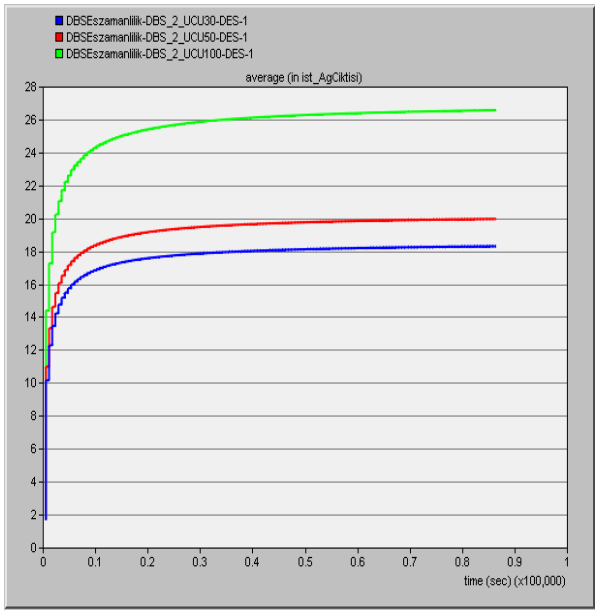
a) %30-50 kaynak paylaşımı gecikme



b) %100 kaynak paylaşımı için gecikme



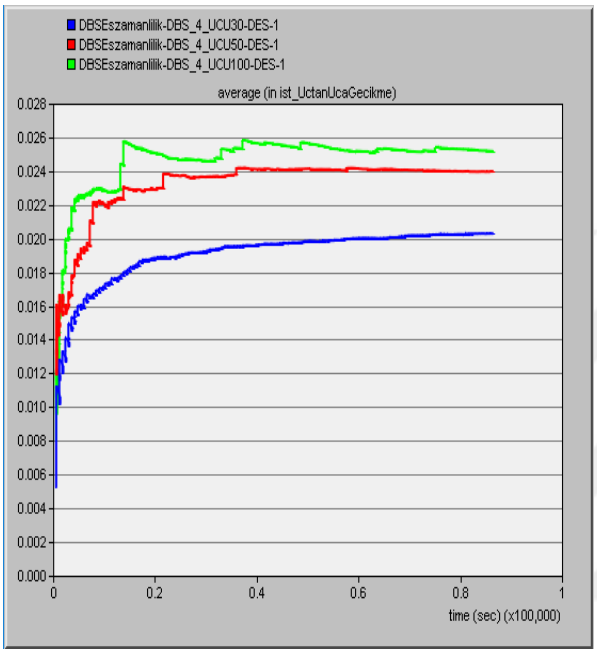
c) Enerji Tüketimi



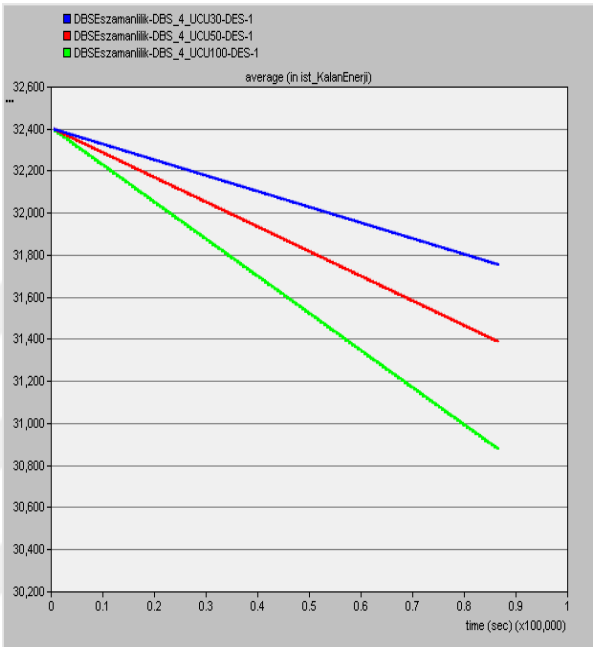
d) Ağ Çıktısı (bit/sn)

Şekil 5.6. Uyg.Say: 2, Kay.Pay(%): 30 (mavi),50 (kırmızı),100 (yeşil), Olay Ara.:0.05 sn

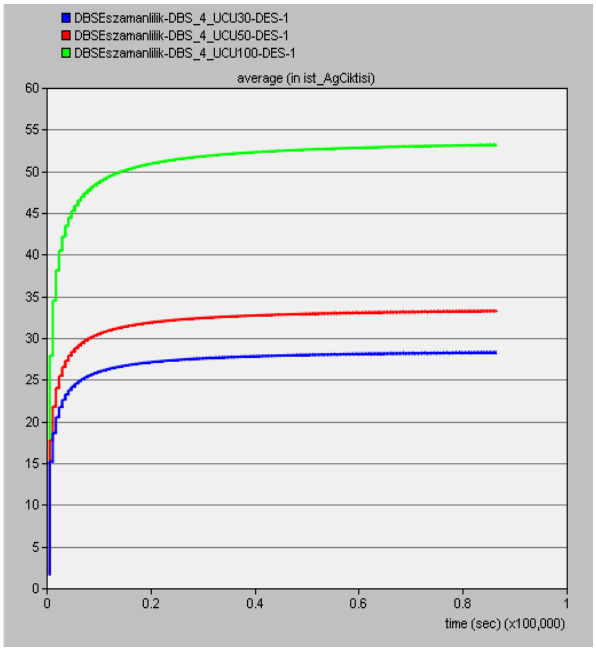
Şekil 5.6 ‘da, düğüm başına 2 uygulamanın yüklü olduğu ve bu uygulamaları süren olayların olma olasılığının 0.05 sn içerisinde gerçekleştiği senaryonun, alt WSN ağ kaynaklarının %30 (mavi), 50 (kırmızı) ve 100 (yeşil) paylaşılması durumunda, sondaj usulü seçilen bir alt WSN düğümü için elde edilen simülasyon performans sonuçları verilmiştir. Görüleceği üzere KOOS değerleri tüm kaynak paylaşım durumlarında etkin olmuştur.



a) Uçtan uca gecikme



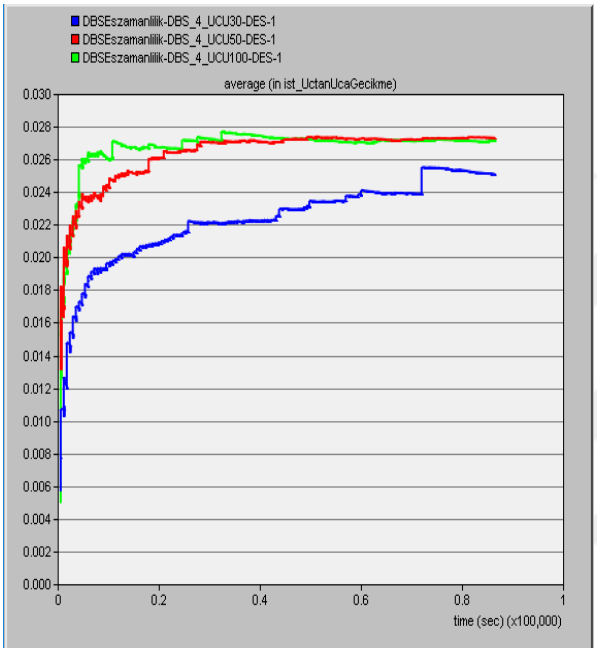
b) Enerji tüketimi



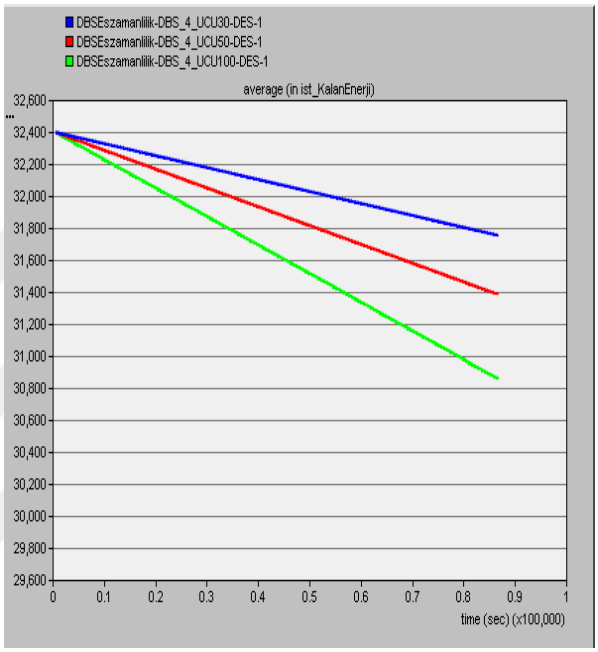
c) Ağ çıktısı

Şekil 5.7. Uyg.Say: 4, Kay.Pay(%): 30 (mavi),50 (kırmızı),100 (yeşil), Olay Ara.:1.5 sn

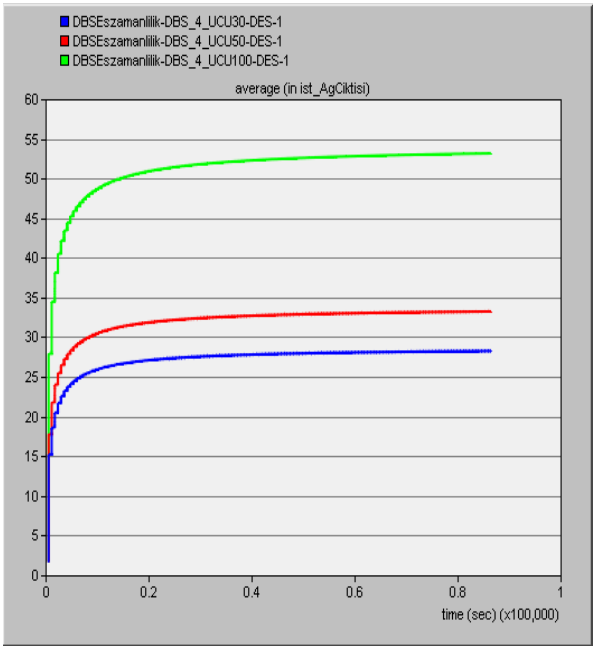
Şekil 5.7 ‘de, düğüm başına 4 uygulamanın yüklü olduğu ve bu uygulamaları süren olayların olma olasılığının 1.5 sn içerisinde gerçekleştiği senaryonun, alt WSN ağ kaynaklarının %30 (mavi), 50 (kırmızı) ve 100 (yeşil) paylaşılması durumunda, sondaj usulü seçilen bir alt WSN düğümü için elde edilen simülasyon performans sonuçları verilmiştir. Görüleceği üzere KOOS değerleri hiçbir kaynak paylaşım durumunda etken olmamış Şekil 5.3 ‘de değerlere benzerlik göstermiştir. Uçtan uca gecikme süresi ortalama 20 msn civarında olmakla birlikte enerji sarfiyatı ve ağ çıktısı değerlerinde önemli bir değişiklik gözlemlenmemektedir.



a) Uçtan uca gecikme



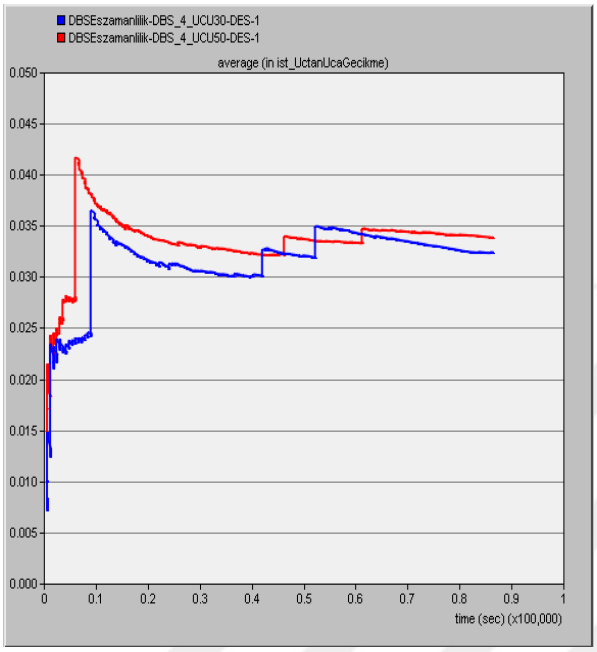
b) Enerji tüketimi



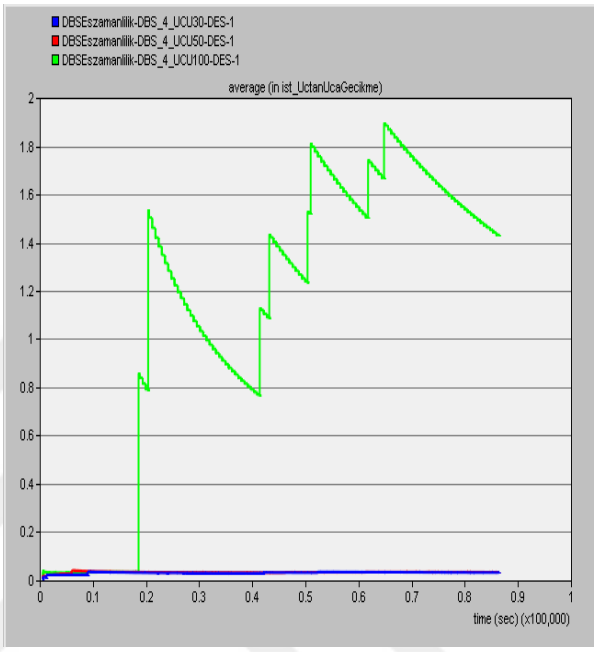
c) Ağ çıktısı

Şekil 5.8. Uyg.Say: 4, Kay.Pay(%): 30 (mavi),50 (kırmızı),100 (yeşil), Olay Ara.:0.5 sn

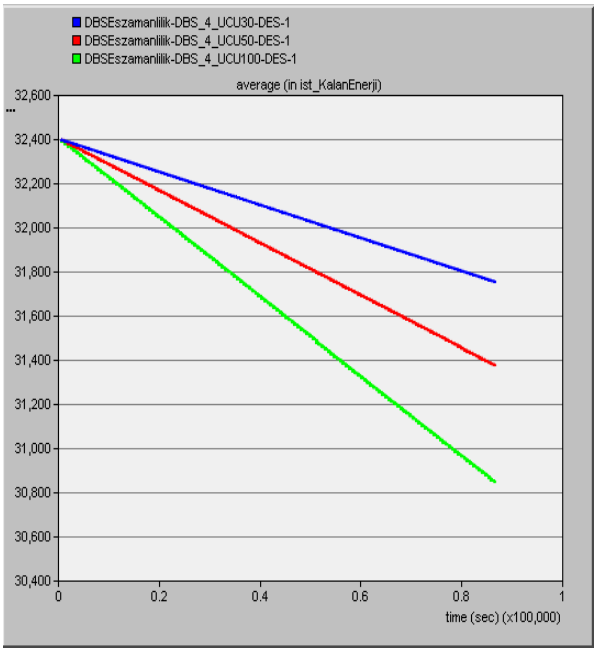
Şekil 5.8 ‘de, düğüm başına 4 uygulamanın yüklü olduğu ve bu uygulamaları süren olayların olma olasılığının 0.5 sn içerisinde gerçekleştiği senaryonun, alt WSN ağ kaynaklarının %30 (mavi), 50 (kırmızı) ve 100 (yeşil) paylaşılması durumunda, sondaj usulü seçilen bir alt WSN düğümü için elde edilen simülasyon performans sonuçları verilmiştir.



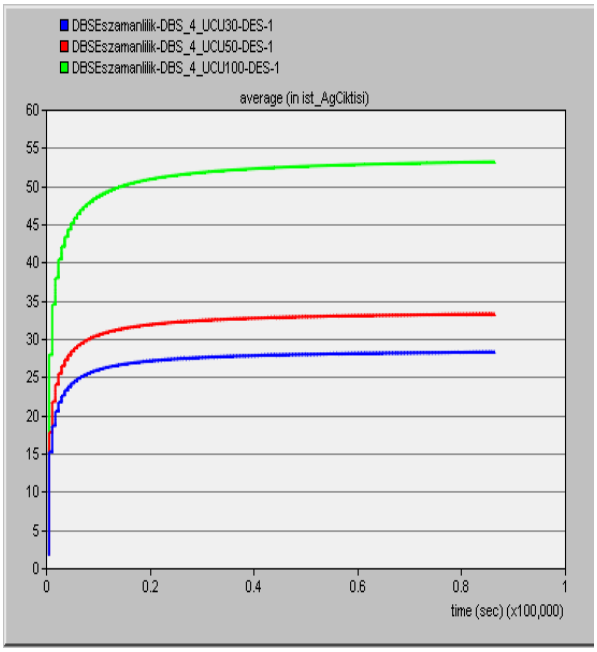
a) % 30-50 kaynak paylaşımı gecikme



b) %100 kaynak paylaşımında gecikme



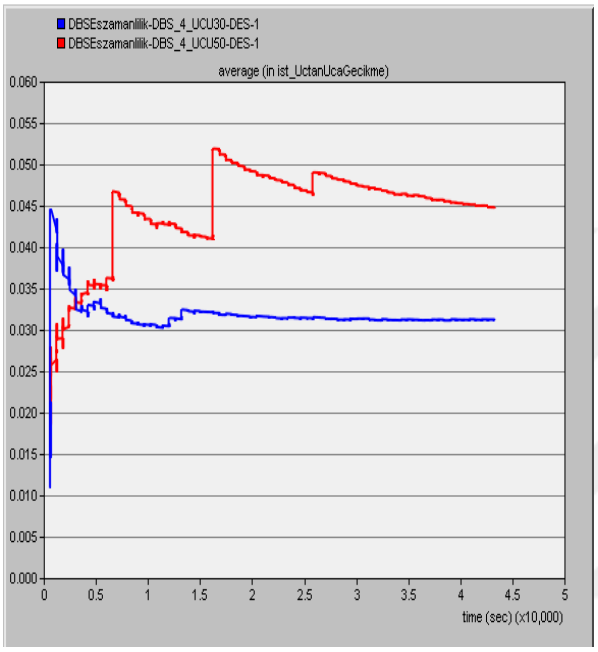
c) Enerji Tüketimi



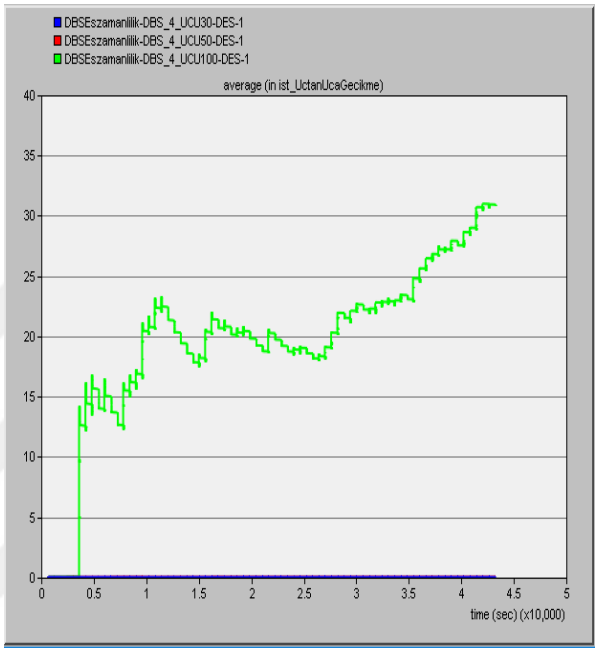
d) Ağ Çıktısı (bit/sn)

Şekil 5.9. Uyg.Say: 4, Kay.Pay(%): 30 (mavi),50 (kırmızı),100 (yeşil), Olay Ara.:0.25 sn

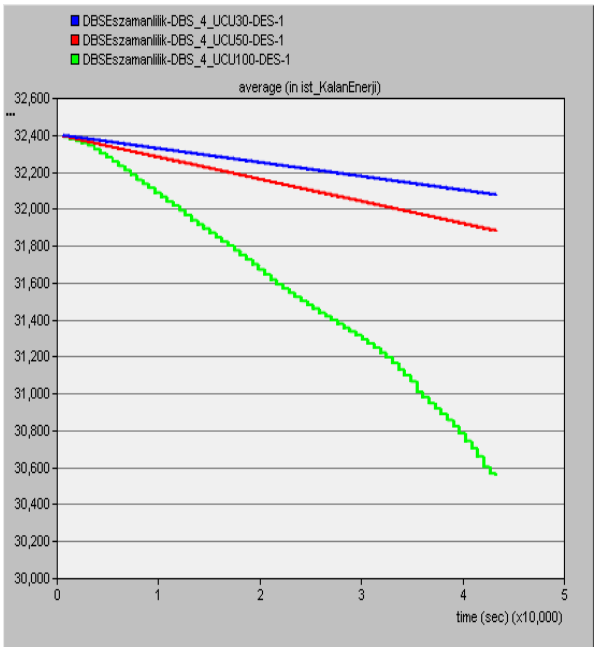
Şekil 5.9 ‘de, düğüm başına 4 uygulamanın yüklü olduğu ve bu uygulamaları süren olayların olma olasılığının 0.25 sn içerisinde gerçekleştiği senaryonun, alt WSN ağ kaynaklarının %30 (mavi), 50 (kırmızı) ve 100 (yeşil) paylaşılması durumunda, sondaj usulü seçilen bir alt WSN düğümü için elde edilen simülasyon performans sonuçları verilmiştir. Görüleceği üzere KOOS değeri %100 için uygulama sayısı 2 olan senaryoya göre daha da yükselmiştir.



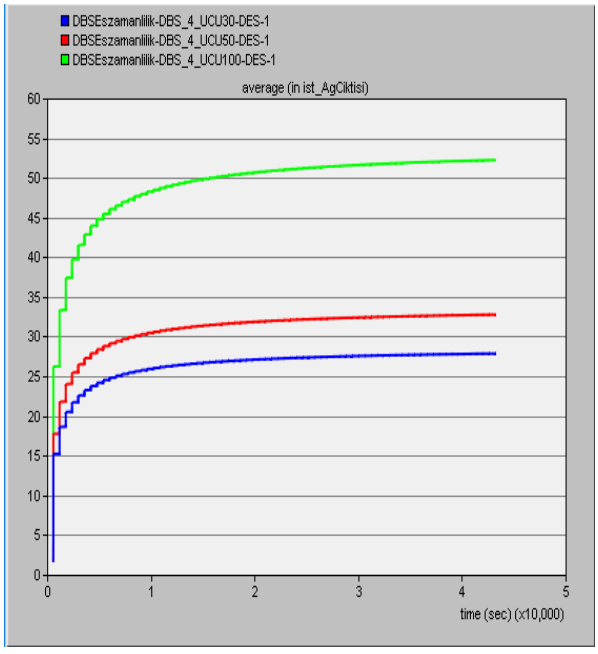
a) % 30-50 kaynak paylaşımı gecikme



b) %100 kaynak paylaşımında gecikme



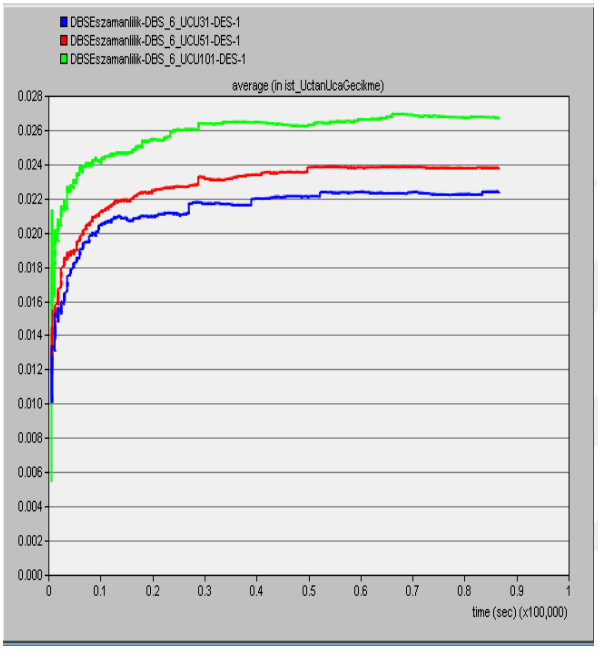
c) Enerji Tüketimi



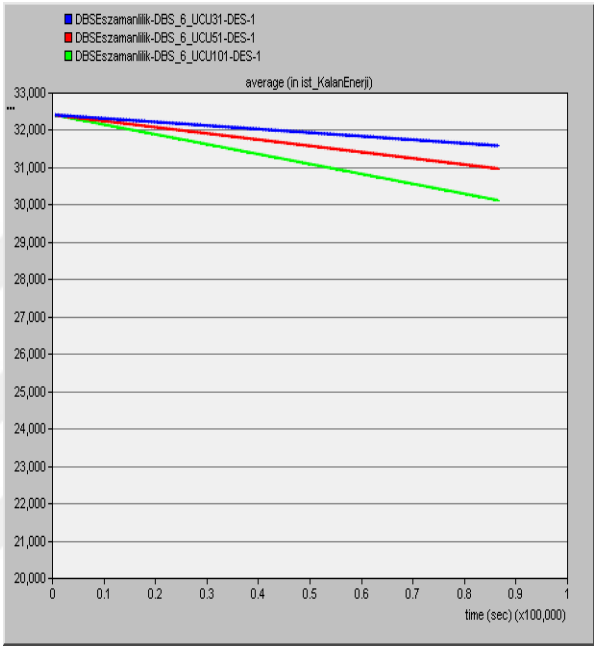
d) Ağ Çıktısı (bit/sn)

Şekil 5.10. Uyg.Say: 4, Kay.Pay(%): 30 (mavi),50 (kırmızı),100 (yeşil), Olay Ara.:0.15 sn

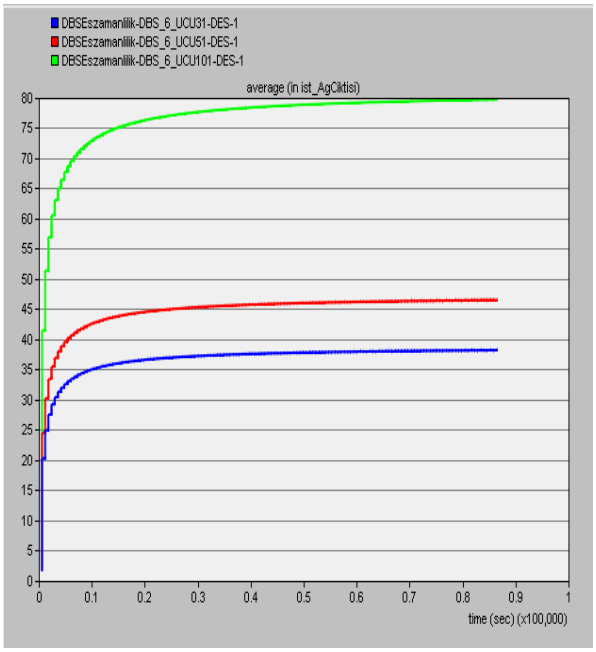
Şekil 5.10 ‘da, düğüm başına 4 uygulamanın yüklü olduğu ve bu uygulamaları süren olayların olma olasılığının 0.15 sn içerisinde gerçekleştiği senaryonun, alt WSN ağ kaynaklarının %30 (mavi), 50 (kırmızı) ve 100 (yeşil) paylaşılması durumunda, sondaj usulü seçilen bir alt WSN düğümü için elde edilen simülasyon performans sonuçları verilmiştir. Görüleceği üzere KOOS değeri halen %100 için etkindir. %30 ve 50 kaynak paylaşımında bir miktar yükselme olmuş olsa da bu oran çok fazla olmamaıştır.



a) Uçtan uca gecikme



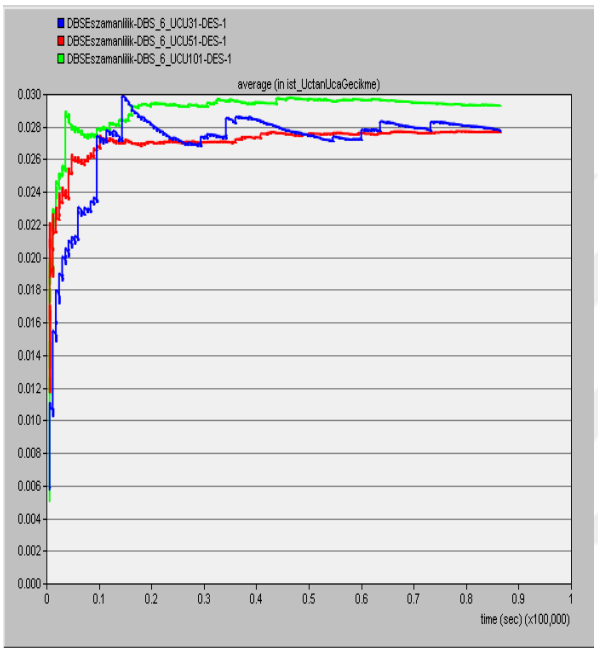
b) Enerji tüketimi



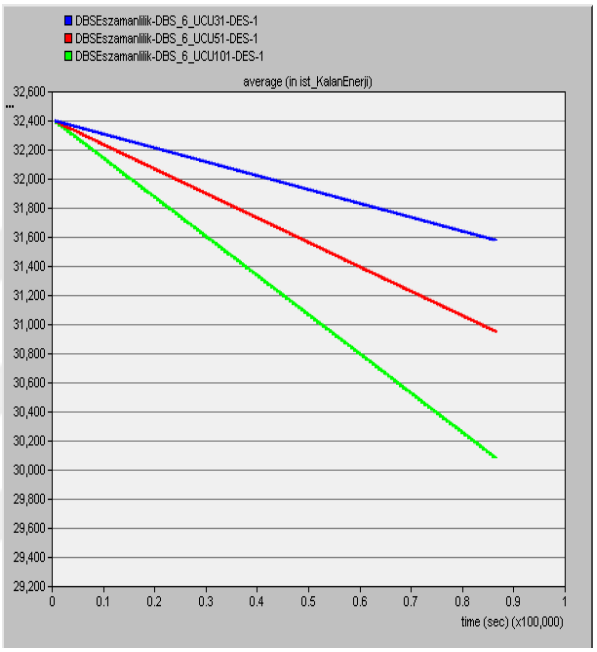
c) Ağ çıktısı

Şekil 5.11. Uyg.Say: 6, Kay.Pay(%): 30 (mavi),50 (kırmızı),100 (yeşil), Olay Ara.:1.5 sn

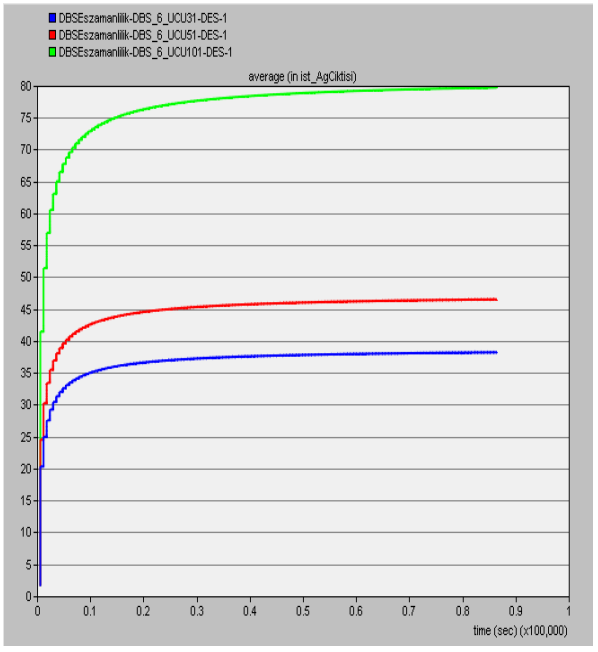
Şekil 5.11 ‘da, düğüm başına 6 uygulamanın yüklü olduğu ve bu uygulamaları süren olayların olma olasılığının 1.5 sn içerisinde gerçekleştiği senaryonun, alt WSN ağ kaynaklarının %30 (mavi), 50 (kırmızı) ve 100 (yeşil) paylaşılması durumunda, sondaj usulü seçilen bir alt WSN düğümü için elde edilen simülasyon performans sonuçları verilmiştir. Görüleceği üzere KOOS değeri, hiçbir durumda etken olmamıştır ve değerler normal seyrinde devam etmiştir.



a) Uçtan uca gecikme



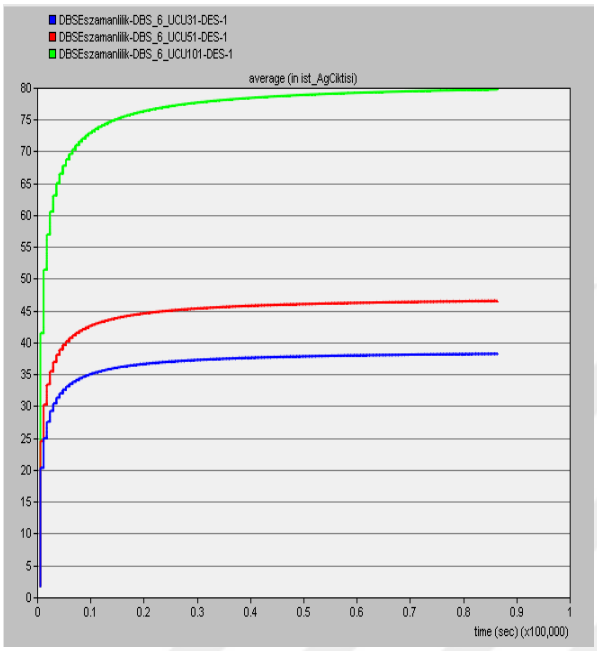
b) Enerji tüketimi



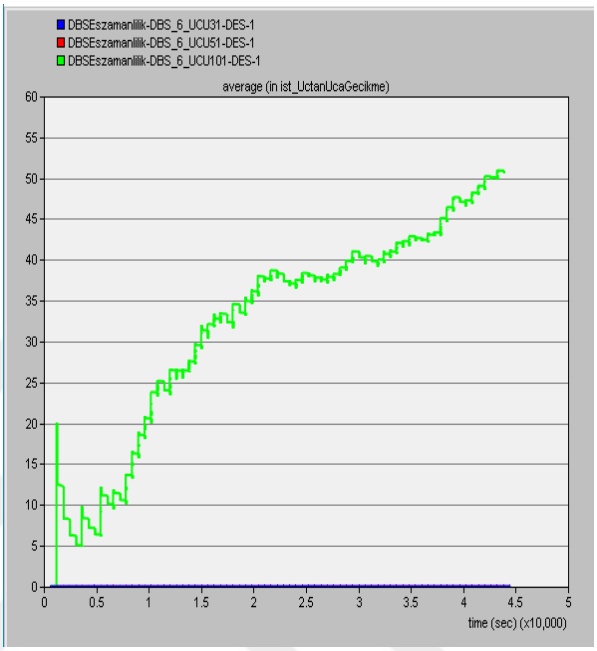
c) Ağ çıktısı

Şekil 5.12. Uyg.Say: 6, Kay.Pay(%): 30 (mavi),50 (kırmızı),100 (yeşil), Olay Ara.:0.5 sn

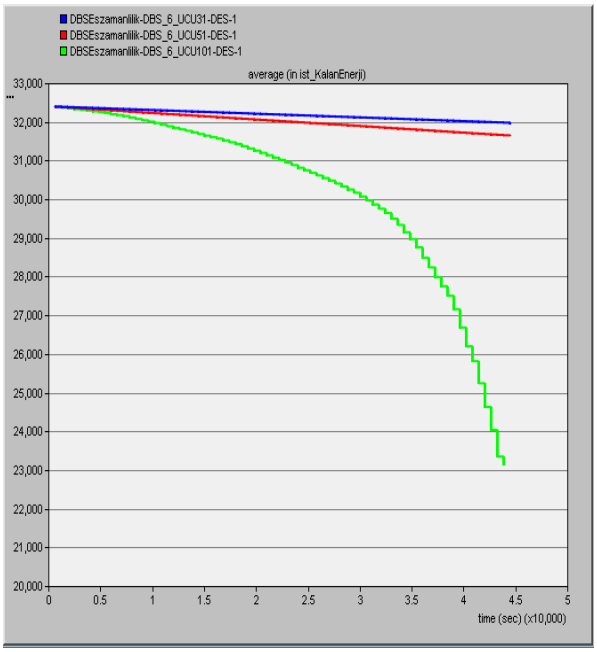
Şekil 5.12 ‘de, düğüm başına 4 uygulamanın yüklü olduğu ve bu uygulamaları süren olayların olma olasılığının 0.5 sn içerisinde gerçekleştiği senaryonun, alt WSN ağ kaynaklarının %30 (mavi), 50 (kırmızı) ve 100 (yeşil) paylaşılması durumunda, sondaj usulü seçilen bir alt WSN düğümü için elde edilen simülasyon performans sonuçları verilmiştir.



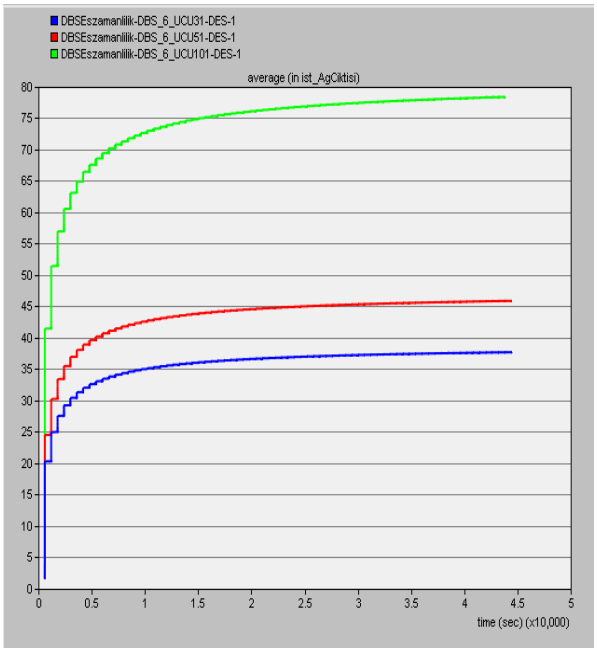
a) % 30-50 kaynak paylaşımı gecikme



b) %100 kaynak paylaşımında gecikme



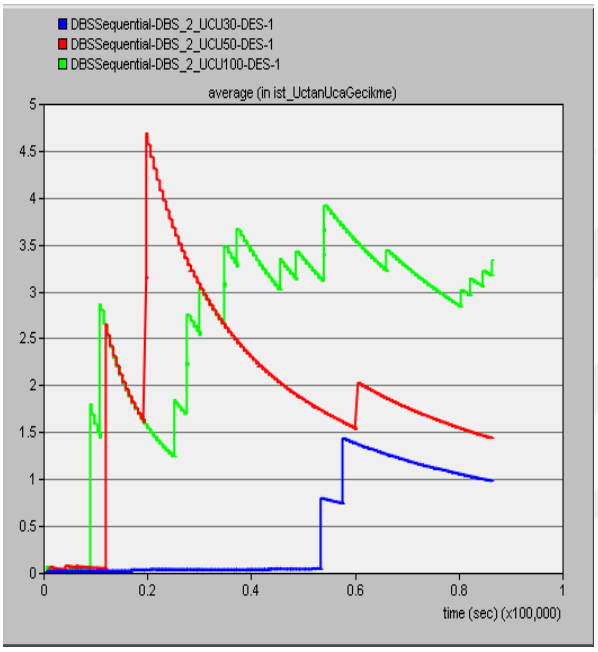
c) Enerji Tüketimi



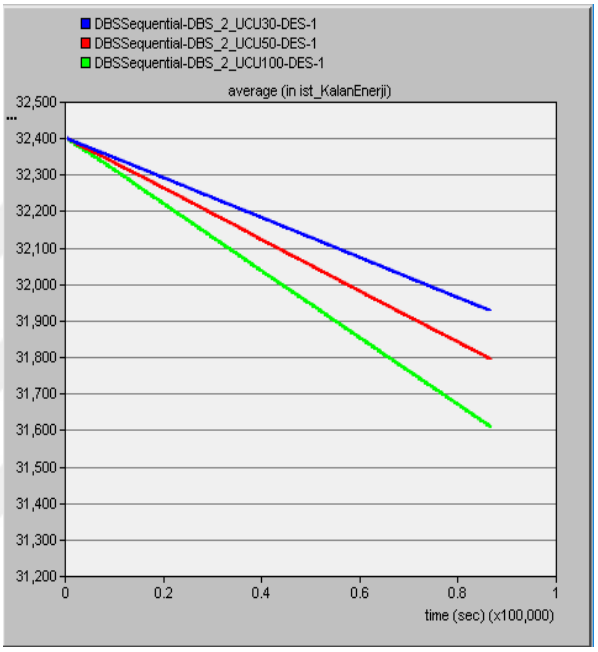
d) Ağ Çıktısı (bit/sn)

Şekil 5.13. Uyg.Say: 6, Kay.Pay(%): 30 (mavi),50 (kırmızı),100 (yeşil), Olay Ara.:0.25 sn

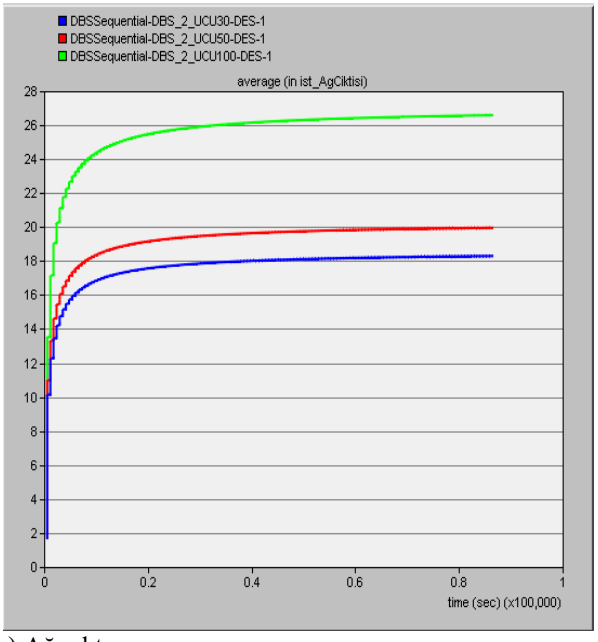
Şekil 5.13 de görüldüğü üzere benzer şekilde altı uygulamalı senaryoda da %100 kaynak paylaşımı olma durumunda, belli bir sürenin altındaki (KOOS=0.25) olay etkileşiminde ağ çıktısının değişmemesine rağmen gecikme ve enerji tüketimi oldukça olumsuz etkilenmektedir. DBS'de diğer bir yöntem ise ardışıl (seqential) çalışma yönteminin kullanılmasıdır. Bu yöntem uygulamaların arka arkaya çalışması ve kaynak verilerine göre yürütümlerinin yapılıp yapılamayacağı mantığı üzerine çalışmaktadır. Event-driven yönteme göre daha az tercih edilirler. Aşağıda sequential (ardışıl) DBS senaryolarının yürütüm sonuçları verilmiştir,



a) Uçtan uca gecikme



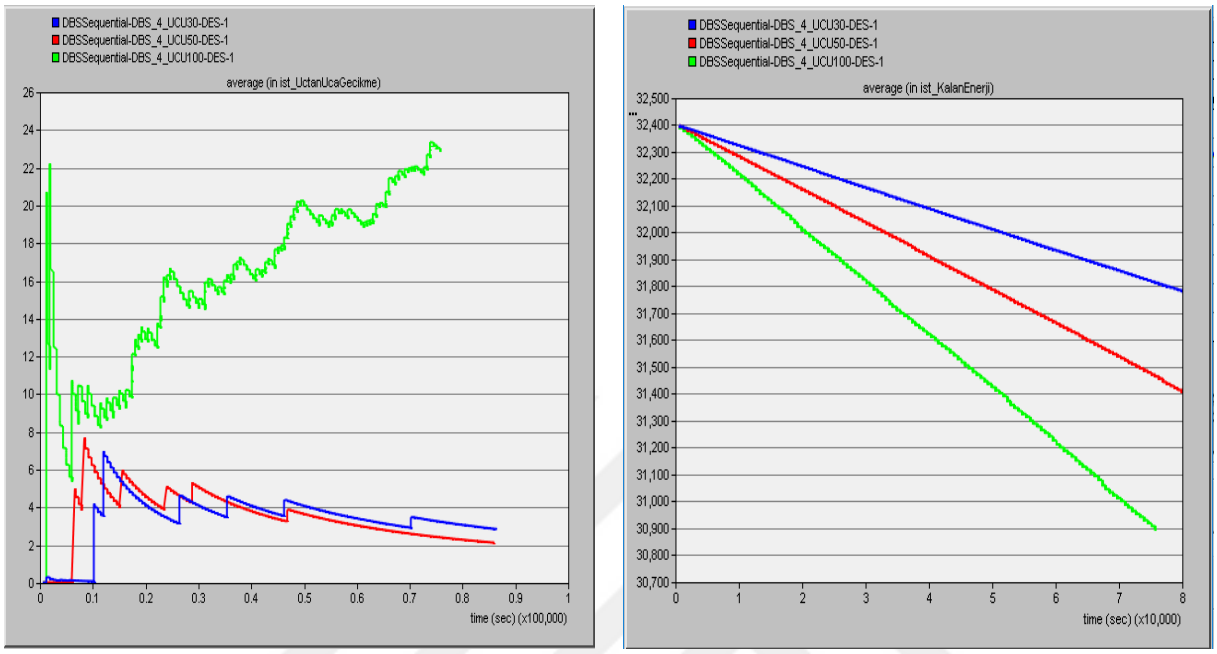
b) Enerji tüketimi



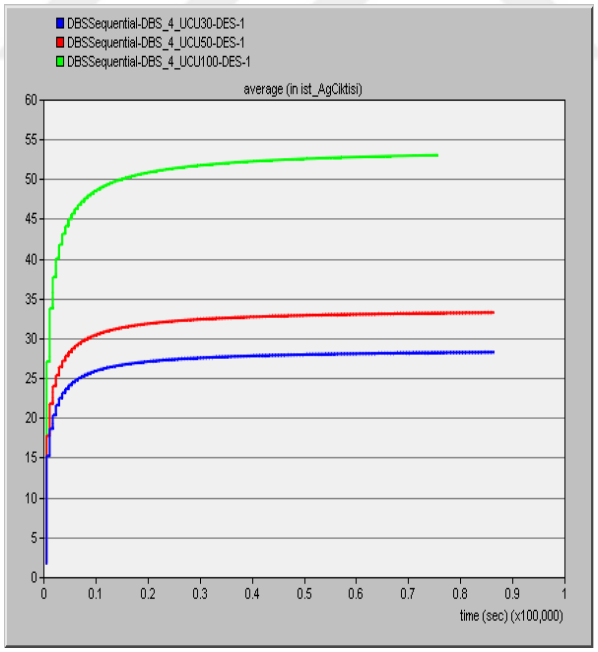
c) Ağ çıktısı

Şekil 5.14. Uyg.Say: 2, Kay.Pay(%): 30 (mavi),50 (kırmızı),100 (yeşil), Mod: Sequential

DBS metodunun 2 uyulama için ardışıl modda çalıştırılması durumunda sondaj usulü seçilen bir düğümden elde edilen veriler Şekil 5.14 ‘de gösterilmiştir. Görüldüğü üzere ardışıl modda uçtan uca gecikme ve enerji sarfiyatı olay tabanlı moda göre daha kötüdür.



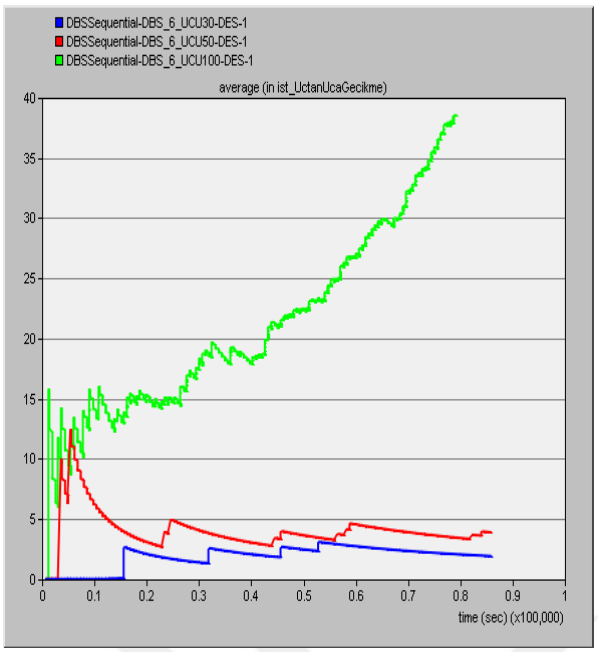
a) Uçtan uca gecikme b) Enerji tüketimi



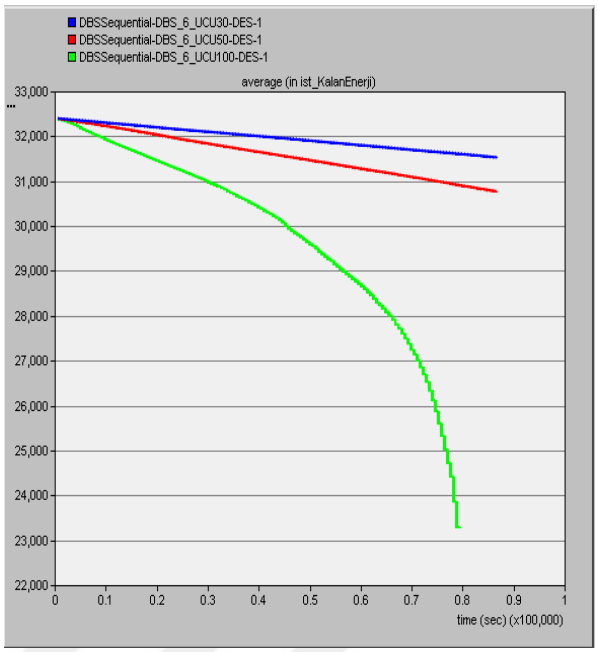
c) Ağ çıktısı

Şekil 5.15. Uyg.Say: 4, Kay.Pay(%): 30 (mavi),50 (kırmızı),100 (yeşil), Mod: Sequential

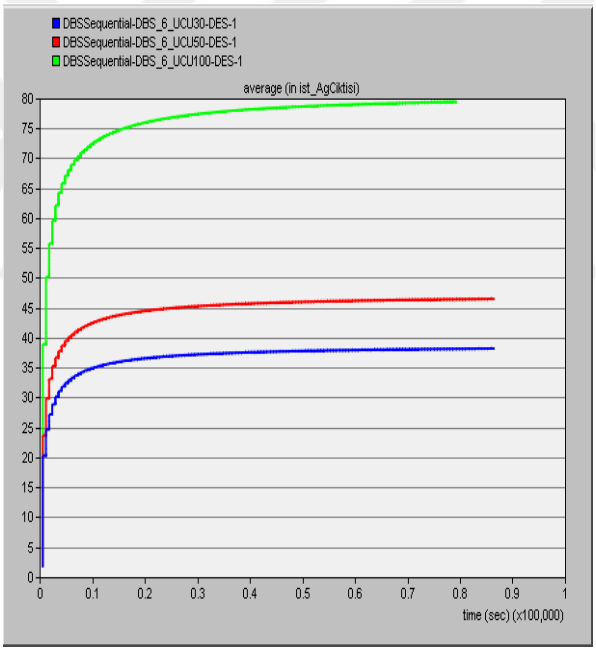
Şekil 5.15 ‘de aynı durum 4 uyulama için de devam etmekte hatta %100 kaynak paylaşımı durumunda daha da olumsuz bir hal almaktadır.



a) Uçtan uca gecikme



b) Enerji tüketimi

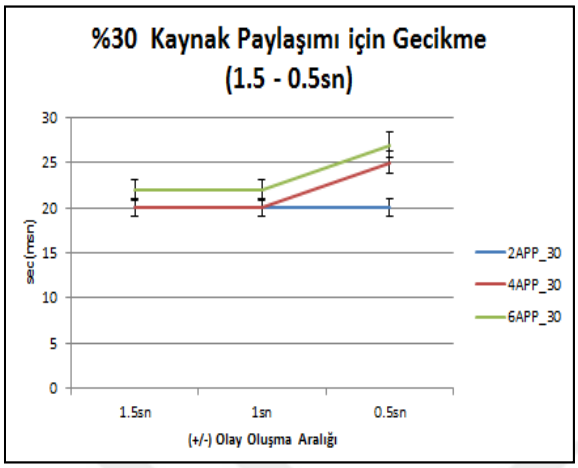


c) Ağ çıktısı

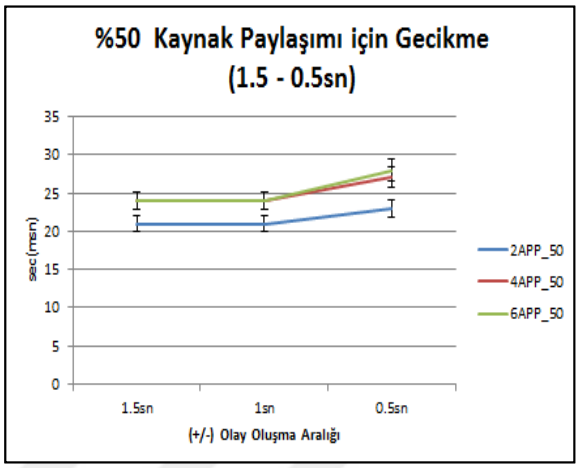
Şekil 5.16. Uyg.Say: 6, Kay.Pay(%): 30 (mavi),50 (kırmızı),100 (yeşil), Mod: Sequential

Şekil 5.16 ‘dan görüleceği üzere sequential çalışma durumunda her ne kadar ağ çıktısı aynı kalsa da gecikme ve enerji tüketimi event-driven çalışma moduna göre daha kötüdür. Bu nedenle DBS teknikleri içerisinde çok tercih edilmemektedir. Yapılan DBS simülasyonlarının genel ortalama sonuçları ve değerlendirilmesi ise aşağıda verilmiştir. İlk olarak Şekil 5.17’ de Event-Driven modunda olay oluşma aralığı ve uçtan uca gecikme değerlerini veren sonuçlar sunulmuştur. Bu sonuçlar olayların oluşma aralığı değerlerine ve kaynak paylaşım oranlarına

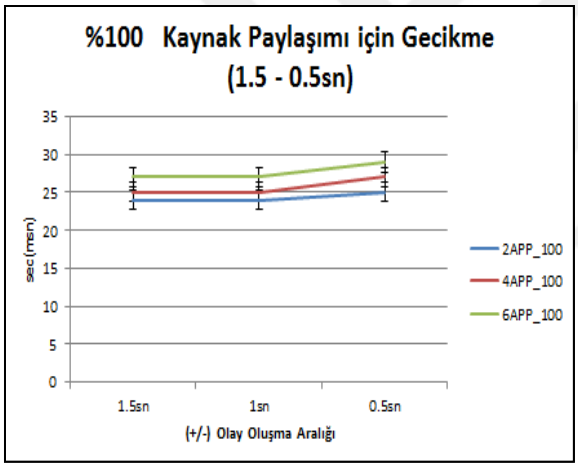
göre iki bölümde verilmiştir. Birinci bölüm (a, b ve c) kritik olay oluşma sıklığına kadar olan gecikme sonuçları (KOOS), ikincisi ise KOOS 'dan sonraki gecikme sonuçlarıdır (d, e ve f).



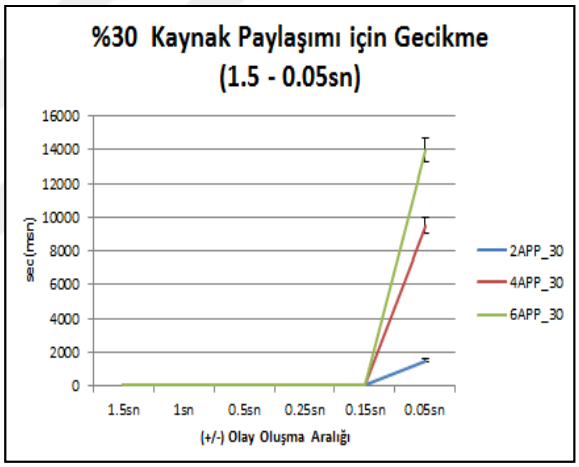
a)



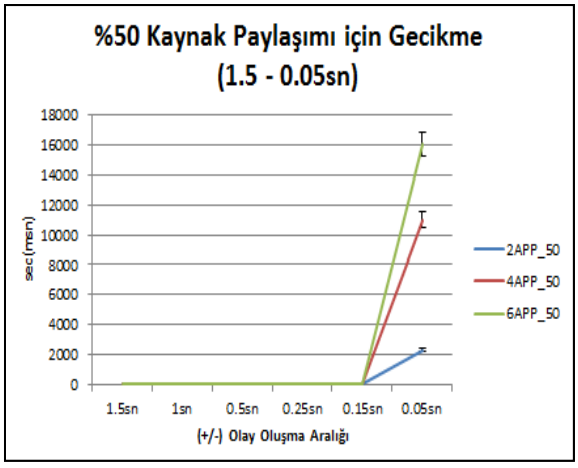
b)



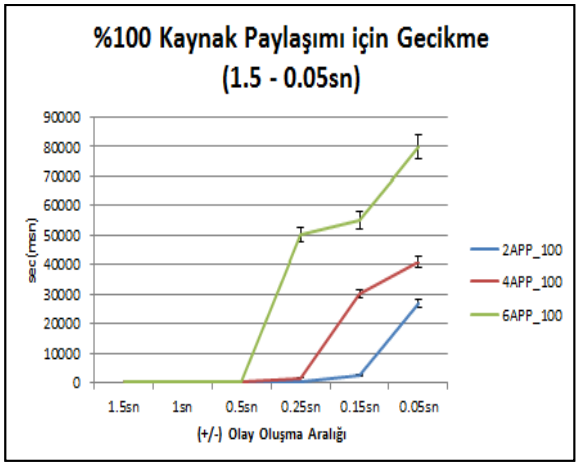
c)



d)



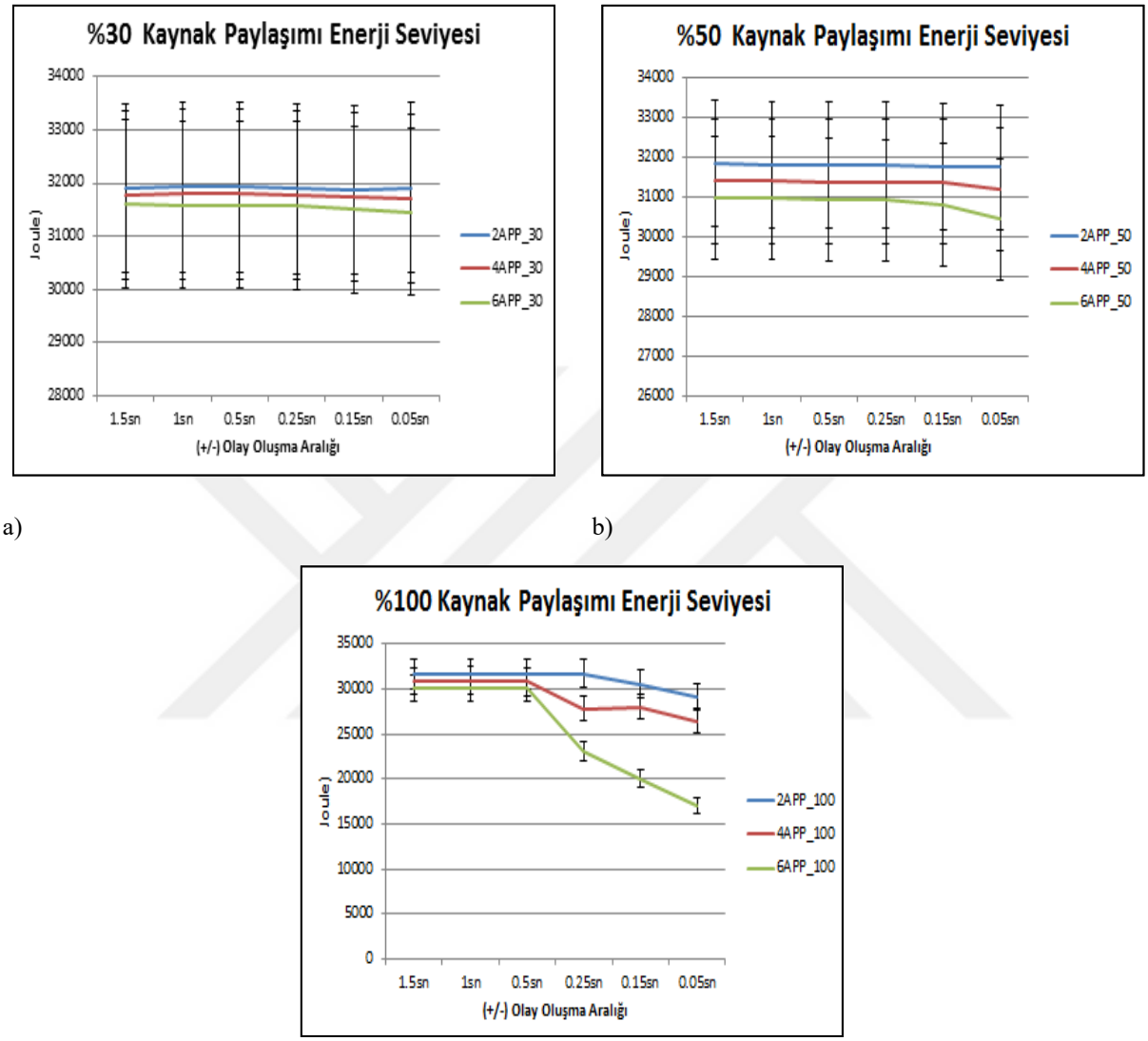
e)



f)

Şekil 5.17. KOSS değeri öncesi ve sonrası uçtan uca gecikme değerlerini kaynak paylaşım oranlarına göre gösteren sonuç grafikleri

Şekil 5.18, event-driven DBS simülasyonlarında genel ortalama enerji tüketimi durumlarını göstermektedir.



a) b)

c)

Şekil 5.18. Kaynak paylaşım oranlarına ve olay oluşma aralıklarına göre event drivent DBS simülasyon sonuçlarını gösteren grafikler

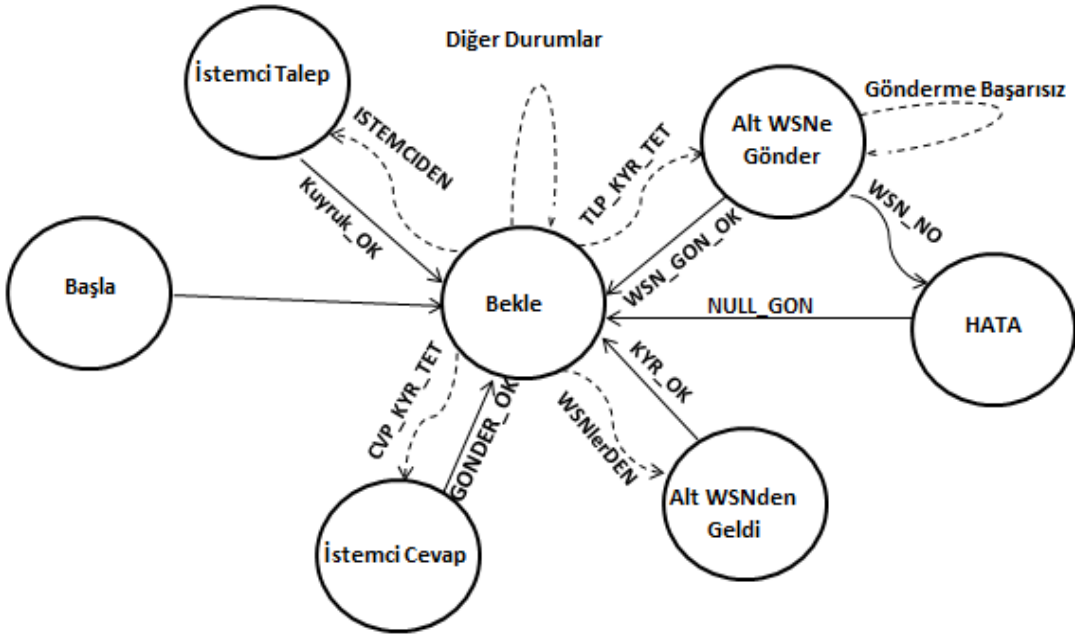
Tüm DBS simülasyonlarından görüldüğü üzere KOOS ‘e kadar ortalama uçtan uca gecikme değeri 20-30 msn iken KOOS ‘den sonra bu değer 20.000 ile 80.000 msn değerine ulaşabilmektedir. KOOS değeri kaynak paylaşım oranına göre değişmektedir. Bu da daha önce açıklanan DBS tekniğinin dezavantajlarına bir yenisini katmaktadır. Aynı şekilde nihai enerji seviyesi daha çok kaynak paylaşım oranının artması ile düşüş göstermektedir. Gecikmedeki ve enerji sarfiyatındaki bu değer artışı, event –driven mod ‘un sequential

çalışma mod ‘a yaklaşmasından ziyade ağ yapısının kullanmış olduğu yarışma temelli, ACK modlu MAC ‘den kaynaklanmaktadır. Farklı MAC modları için incelenmesi gereken bir problem olarak ortaya konulan bu durum tez kapsamı dahilinde olmadığından daha sonraki çalışma konuları içerisinde yer alacaktır. Burada dikkat edilmesi gereken nokta, sonraki kaynak paylaşım/sanallaştırma teknikleri ile kıyaslanması için istemci sayısı ve normal çalışma durumunda elde edilen ortalama gecikme ve enerji seviyeleridir.

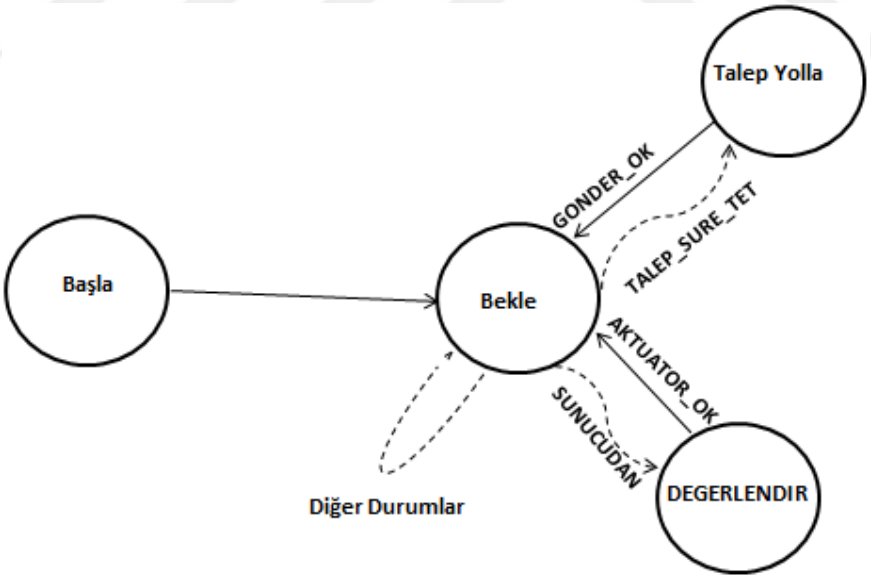
5.2. ABS temelli IoT Kaynak Paylaşımı Simülasyonu

Diğer bir WSN kaynak paylaşımı/sanallaştırma tekniği ağ bazlı sanallaştırmadır. Bu teknikte değerlendirme ve karar verme yeri istemcilerdir. Bu teknik temelde iki şekilde uygulanmaktadır. Birincisi özel P2P protokoller vasıtasıyla örtüşen veya kümesel WSN ağlar oluşturmak, ikincisi ise uzaktan fonksiyon çağrımları ile farklı WSN ‘lerden ortak bir amaç için faydalanmaktır. Bunlardan ilki tüm WSN teknolojisi tarafından desteklenmez. Daha çok sınırlı sayıda WSN ‘ler arasında spesifik amaçlara yönelik uygulamaları bulunmaktadır. İkinci ABS yöntemi ise uzun yıllardır bilinen API teknolojileri kullanmaktadır. Bu teknolojilerde internet alt yapısını kullanırlar ancak daha basit bir işleyiş yapıları vardır. Veri yönetimi, depolama ve filtreleme gibi gelişmiş alt yapı hizmetleri yoktur. API bazlı yöntemlerin uygulanabilirliğinin daha geniş olması, bir önceki DBS simülasyonları ile daha kolay kıyaslama yapılabilir olması nedeni ile bu bölümdeki simülasyon senaryosu bu yöntemi baz almıştır. Bu ABS simülasyonunda, bir önceki DBS simülasyonlardaki WSN yapıları değiştirilmemiş sadece sunucu yapısı API moda göre düzenlenmiştir. Senaryo gereği istemciler talepleri doğrultusunda sunucu tarafından sunulan API ‘ler vasıtası ile bir veya daha fazla farklı WSN kaynaklarından veri talep edebilmektedir. Alınan verilerin değerlendirmesi istemci tarafta yapılmakta ve değerlendirmenin sonucunda gerekli ise sunucu taraftaki başka bir kaynakla (örneğin aktuatör) etkileşim gerçekleştirilmektedir. Bu senaryoda alınan verilerin değerlendirilmesi istemci üzerinde O(n^2) karmaşıklığındaki bir algoritma ile yapılmakta ve değerlendirme sonucunun belirlenen eşik değerlerin üzerinde çıkması durumunda farklı bir WSN ağdaki aktuatörün set edilmesi sağlanmaktadır. İstemcilerin veri talep etmesi ve değerlendirme sonucunda ilgili aktuatörün set edilmesine kadar olan ortalama gecikme, birinci performans metriğidir. İkinci metrik WSN düğümler üzerindeki ortalama enerji tüketimi, üçüncüsü ise ağ çıktısı (throughput) olacaktır. Yine bir önceki simülasyonlarda olduğu gibi kaynak paylaşımlarının %30-50-100 oranında yapıldığı senaryo örnekleri ile çalışılmıştır. Şekil 5.19 da sunucu sisteme ve istemci uygulama katmanına ait

FSM diyagramları verilmiştir. Sistem pratikte olduğu üzere eş zamanlı talepler ($Talep_{i+1} - Talep_i < \tau,\ \tau = eş\ zamanlılık\ eşiği$) için ön bellek verilerini kullanmaktadır. Bu alt WSN 'lerdeki enerji tasarrufu için sunucu sistemlerde kullanılır.



a) Sunucu sistem FSM

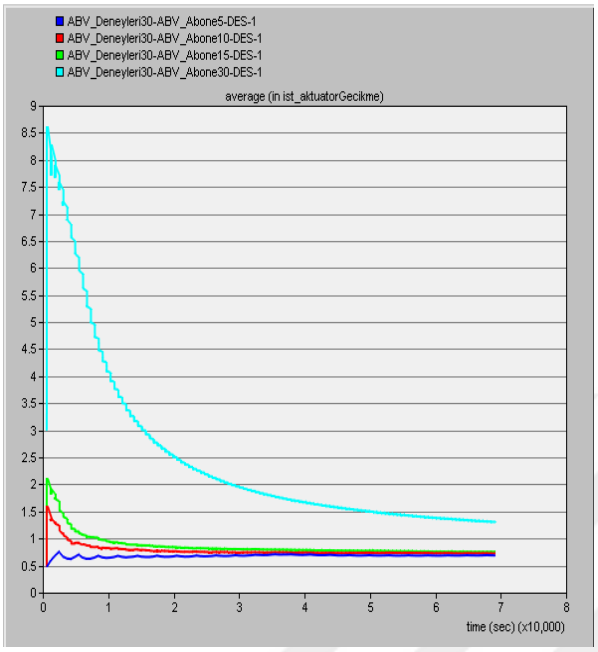


b) İstemci uygulama FSM

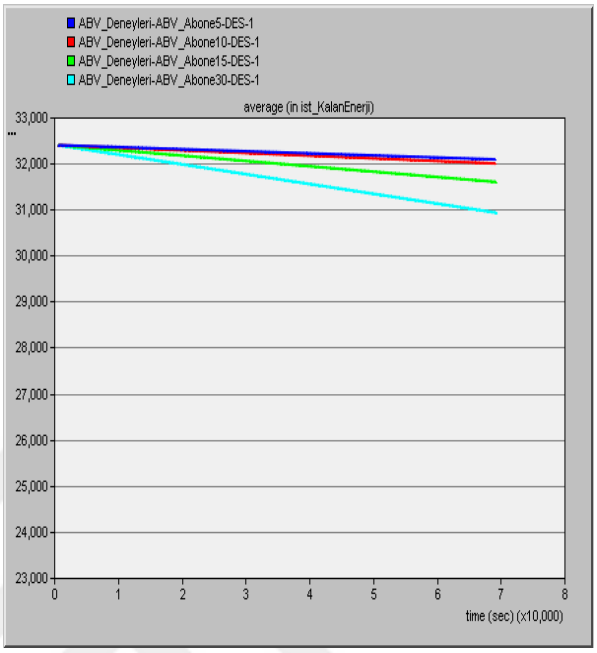
Şekil 5.19. ABS Sunucu sistem ve istemci uygulama katmanı FSM

Şekil 5.20-22 'de DBS simülasyonlarındaki aynı örnek WSN düğümlerinden alınan performans metrik grafikleri sunulmuş ve Şekil 5.23 'de ise ortalama genel metrik değerleri

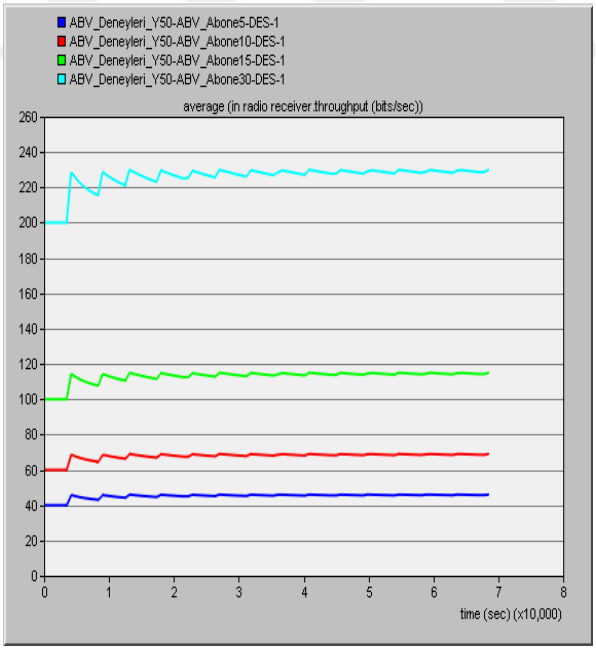
paylaşılmıştır. Yapılan ABS deneyleri sırasıyla 5, 10, 15 ve 30 istemci için % 30, 50 ve 100 kaynak paylaşımları senaryoları için denenmiştir. Tüm ABS sonuç grafiklerinde açık mavi 30 istemci, yeşil 15 istemci, kırmızı 10 istemci ve lacivert ise 5 istemciye ait sonuçları göstermektedir.



a) Uçtan uca gecikme



b) Enerji tüketimi

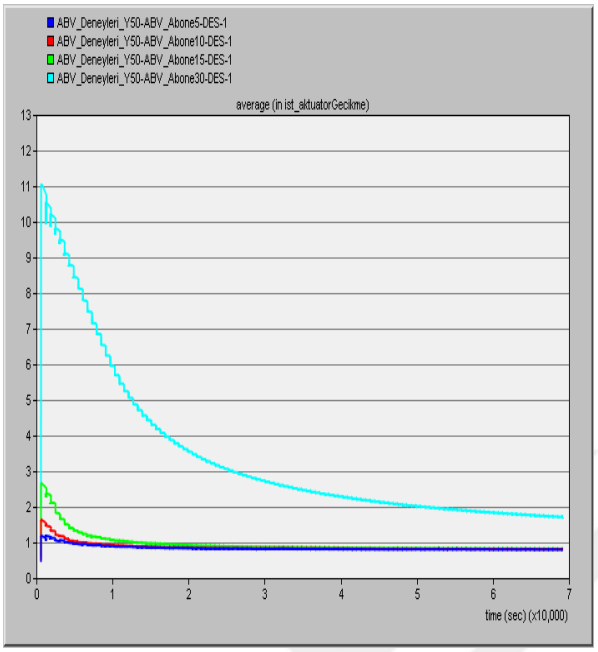


c) Ağ çıktısı

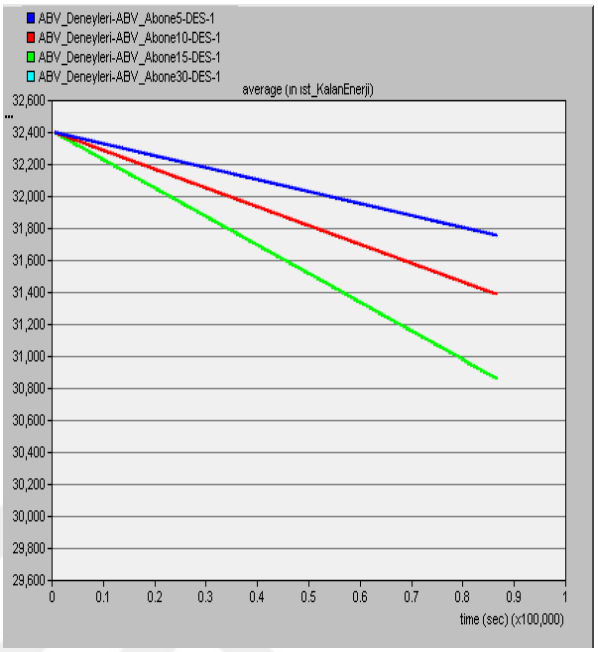
Şekil 5.20. ABS 5-30 istemci için %30 kaynak paylaşımı için metrik sonuçları

Şekil 5.20 ‘de farklı kullanıcı sayıları için %30 kaynak paylaşımı durumunda sondaj usulü seçilen alt WSN düğümünden elde edilen sonuçları göstermektedir. Sonuçlardan da

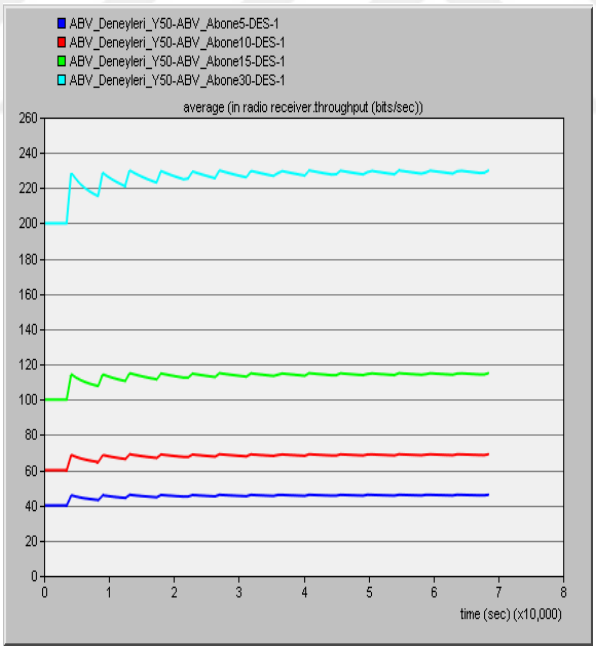
görüleceği üzere gecikme değerleri KOOS durumu olmayan DBS ‘ye göre oldukça yüksektir. Kullanıcı sayısının artması ile gecikme ve enerji sarfiyatı da artmakdadır. Ön bellekten veri okuma tüm ABS senaryolarında geçerlidir.



a) Uçtan uca gecikme



b) Enerji tüketimi

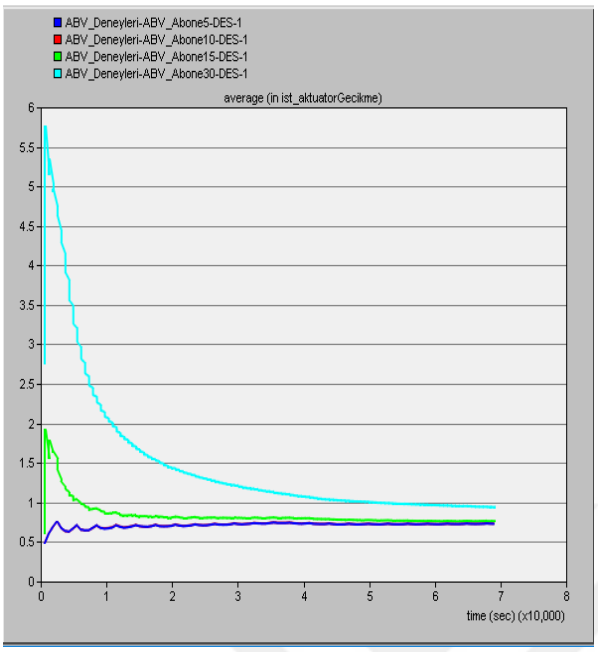


c) Ağ çıktısı

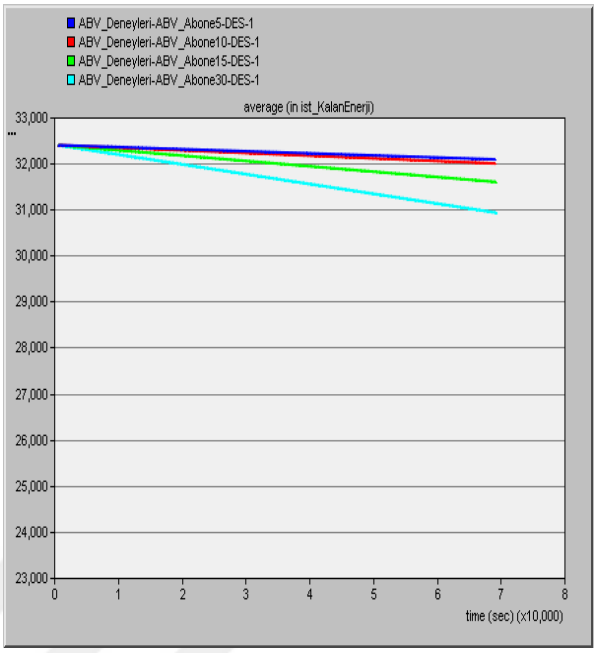
Şekil 5.21. ABS5-30 istemci için %50 kaynak paylaşımı için metrik sonuçları

Şekil 5.21 ‘de farklı kullanıcı sayıları için %50 kaynak paylaşımı durumunda sondaj usulü seçilen alt WSN düğümünden elde edilen sonuçları göstermektedir. Sonuçlardan da

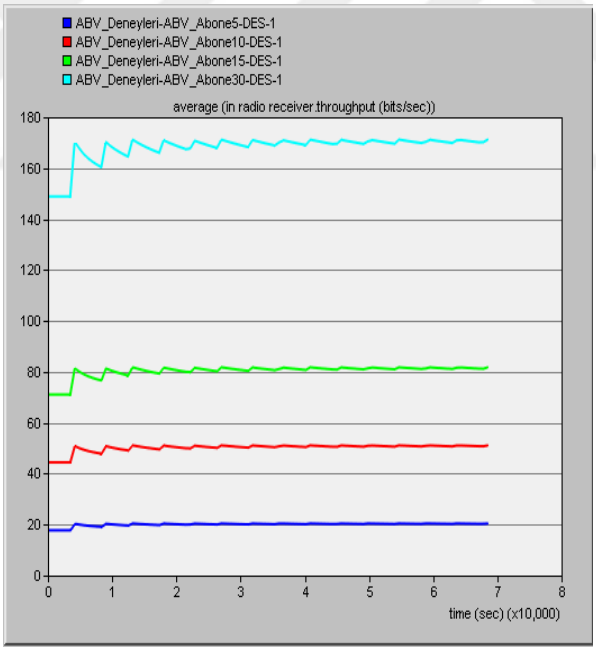
görüleceği üzere gecikme değerleri KOOS durumu olmayan DBS ‘ye göre halen oldukça yüksektir. Kullanıcı sayısının artması ile gecikme ve enerji sarfiyatı da artmakdadır.



a) Uçtan uca gecikme



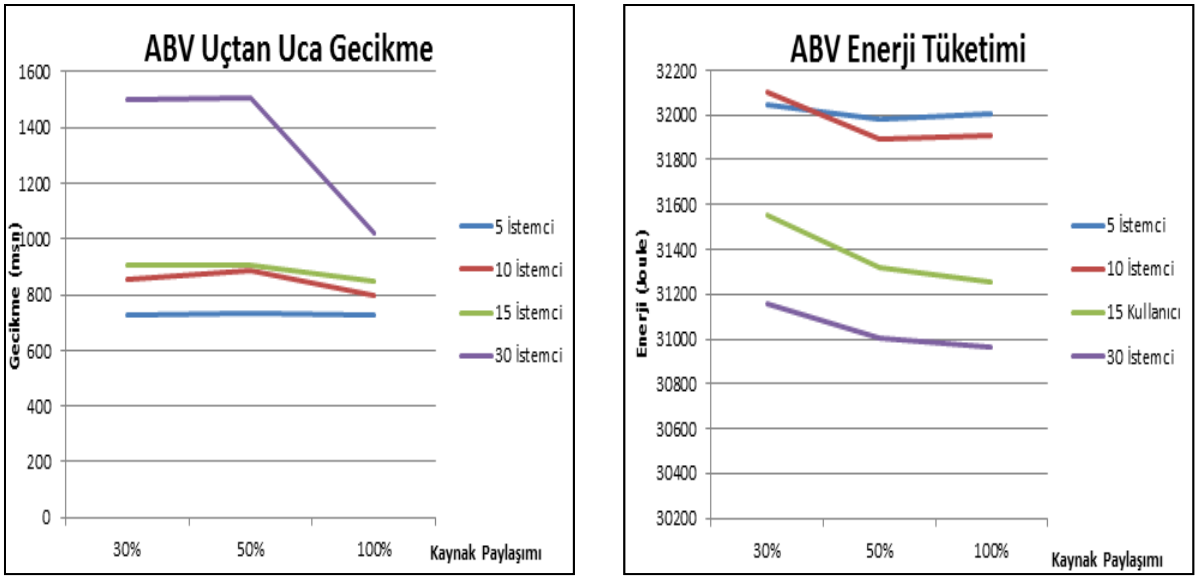
b) Enerji tüketimi



c) Ağ çıktısı

Şekil 5.22. ABS 5-30 istemci için %100 kaynak paylaşımı için metrik sonuçları

Şekil 5.22 ‘de farklı kullanıcı sayıları için %100 kaynak paylaşımı durumunda sondaj usulü seçilen alt WSN düğümünden elde edilen sonuçları göstermektedir. Sonuçlardan da görüleceği üzere gecikme değerleri yüksek olmakla beraber kaynak paylaşım oranlarının değişmesi ile çok da fazla değişmemiştir.



a) b)

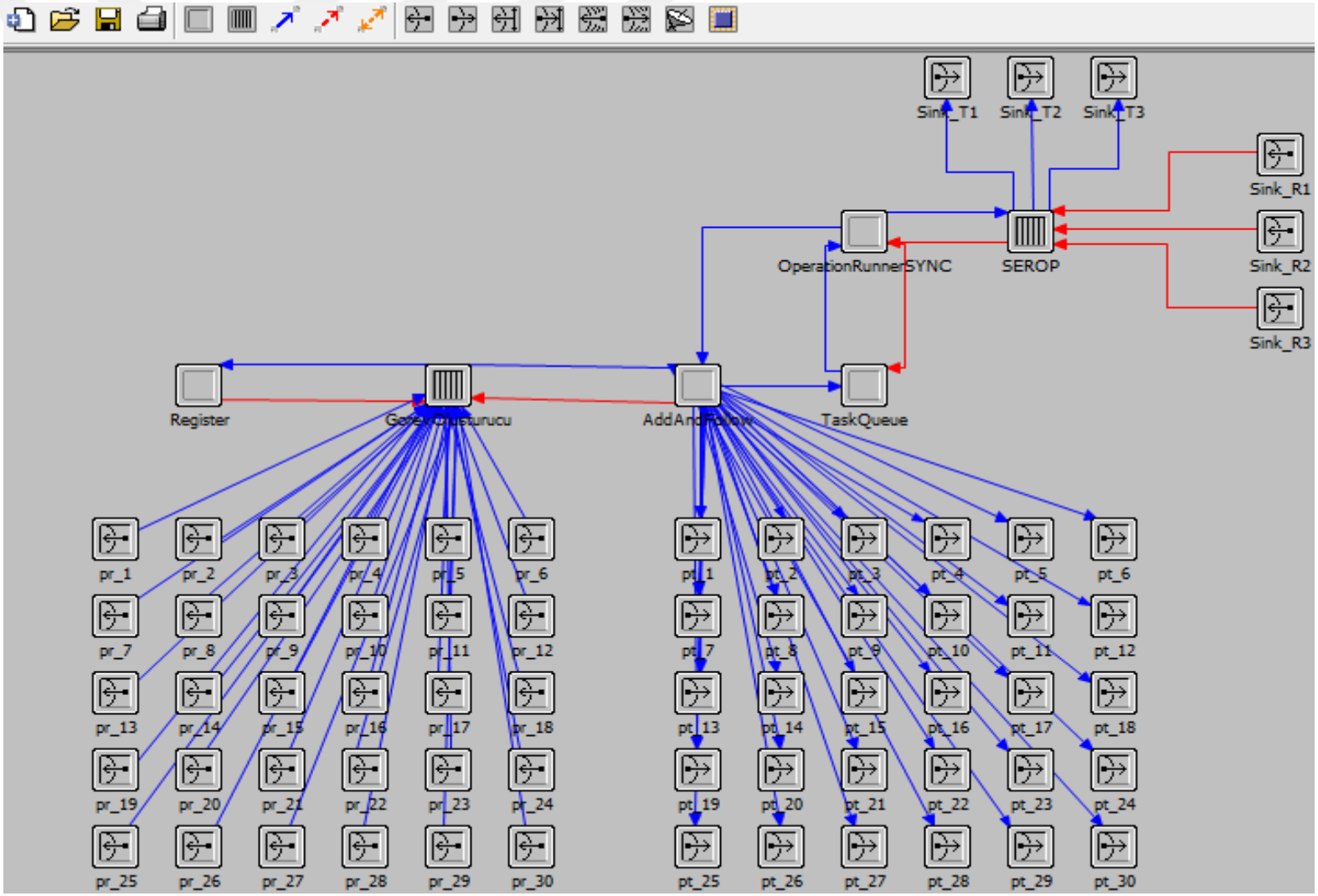
Şekil 5.23. ABS ortalama genel metrik sonuçları

ABS simülasyonları %30-50-100 kaynak paylaşımlarında, 5-10-15-30 istemci için gerçekleştirilmiştir. Görüleceği üzere ABS, uçtan uca gecikmede, DBS 'ye göre en iyi ve en kötü durumda yaklaşık 40 ile 70 kat arası bir artış söz konusu olabilmektedir. Öte yandan enerji tüketimi açısından 30 kullanıcıdaki enerji tüketimi bile yaklaşık olarak DBS 'nin %30 kaynak paylaşımındaki değere yakın olmaktadır. Bir diğer değişle gecikmede her ne kadar bir artış olsa da bu yöntemle daha fazla kullanıcıya kaynak paylaşımı yapılabilmektedir. Buradaki kaynak paylaşım oranlarının düşmesi daha fazla kullanıcının daha az kaynağa talep göstermesi ve alt WSN ağlarında bölgesel gecikmenin artmasına yol açmaktadır. Bu etki özellikle 30 istemcide kendini daha da göstermektedir. Kaynak paylaşım oranı arttıkça alt ağlardaki talep dengesi daha tutarlı olduğundan gecikme değeri daha da düşebilmektedir. Bu düşüşe sebep olan bir diğer etken ise, eş zamanlı taleplerin ön bellek sonuçları ile karşılanmasıdır. Bu sonuçlardan da görüleceği üzere sunucu tabanlı kaynak paylaşımları DBS sistemlere göre enerji verimi açısından daha uygun iken zaman kritik uygulamalar için DBS sistemler daha etkindir.

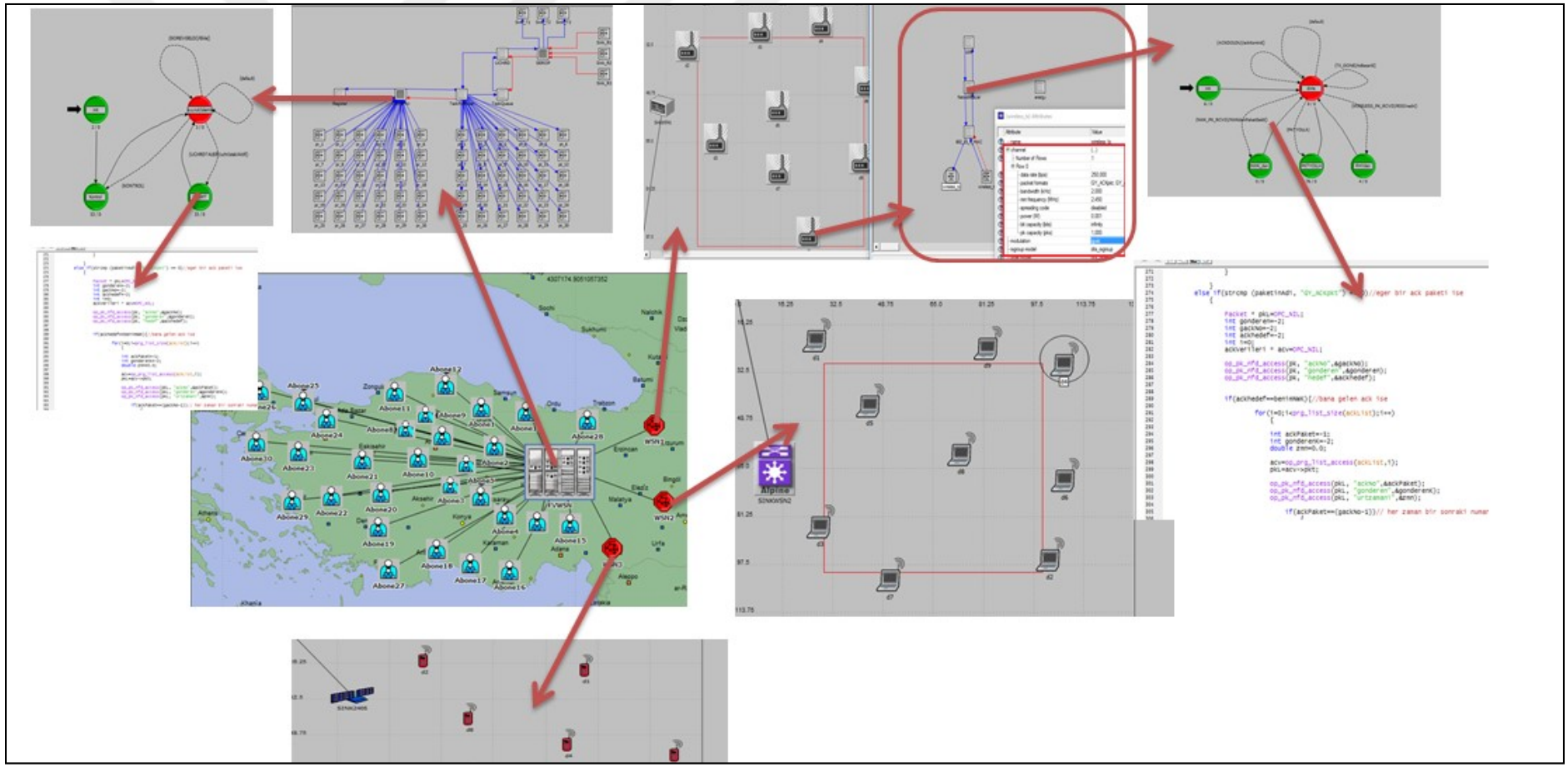
5.3. FVWSN temelli IoT/CPS WSN Sağlayıcı Sistem Simülasyonu

Bu bölümde önceki bölümlerde detayları ve modellemesi verilmiş olan önerilen FVWSN temelli ATBS sistemin simülasyonu OPNET platformunda yapılarak DBS ve ABS 'ye göre karşılaştırılması yapılacaktır. Genel blok yapısı Bölüm 3 'de Şekil 3.22 'de verilen sistemin

OPNET Modeler platformunda gerçekleştirme şekli Şekil 5.24 ‘de ve genel simülasyon yapısı ise Şekil 5.25 ‘de verilmiştir. Her bir modüle ait çalışma yapısı Bölüm 3 ve 4 ‘deki temel prensiplere göre OPNET Modeler platformunda gerçekleştirilmiştir. Önerilen sistemin yürütümünde Bölüm 4 ‘de verilen Denklem 4.22-4.25 ve 4.52-4.53 modelleri esas alınmıştır. Alt WSN sistemlerin yapısı daha önceki DBS ve ABS simülasyonlarında kullanılanların aynısıdır. İstemcilerin kendi komut ve sorgularını işletebilecek ilgili VNd ‘ler “child process” olarak sunucu modülde tanımlanmıştır. FVWSN temelli WSN sağlayıcı sistemin sonuçlarının irdelenmesinde, diğer tekniklerde olduğu gibi öncelikle alt WSN düğümlerinin davranışları hakkında fikir vermesi açısından aynı kaynak düğümlerden alınan veriler sunulacaktır. Daha sonra tüm sistem kaynaklarından elde edilen ortalama genel sonuçlar paylaşılacak ve diğer tekniklerde elde edilen sonuçlar ile karşılaştırılması yapılacaktır. Burada yine %30-50-100 kaynak paylaşımında 5-10-15-30 istemci için sistem davranışları incelenecektir. Kullanılan sistem metrikleri önceki deneylerle aynıdır. Simülasyonlarda istemciler komut/sorguları ile alt WSN kaynaklardan elde edilen veriler sunucu da ilgili VNd ’lerdeki O(n^2) algoritmalar ile değerlendirilecek ve sonuçta farklı WSN ‘lerdeki bir aktuatör uygun değere set edilecektir.

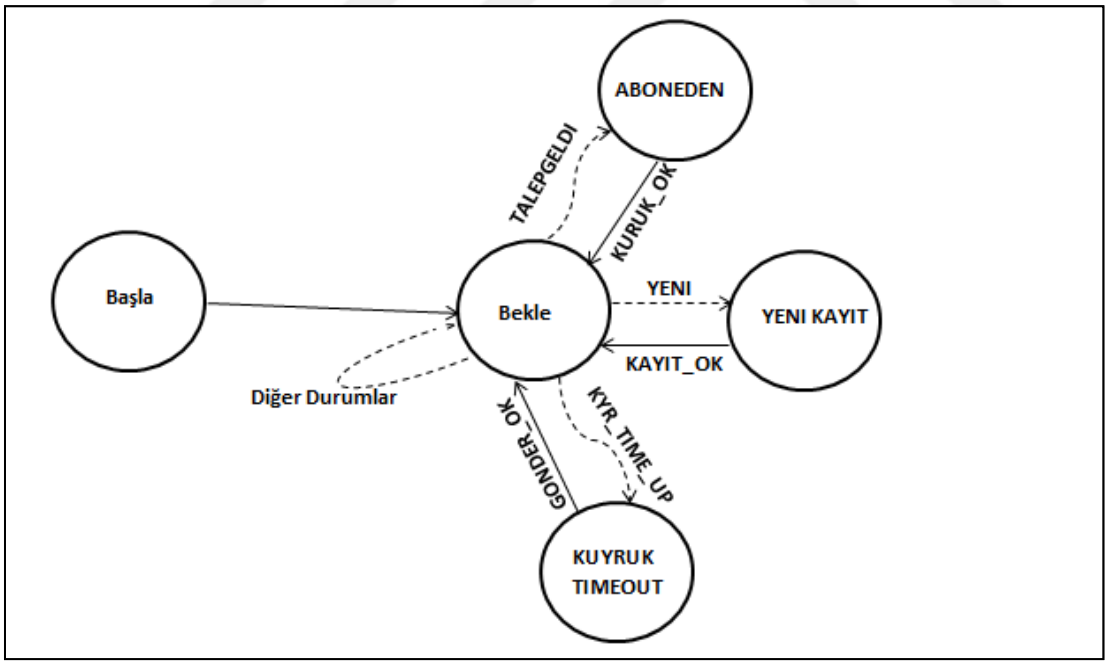


Şekil 5.24. Önerilen sistem blok diyagramının OPNET Modeler ‘de gerçeklenmesi

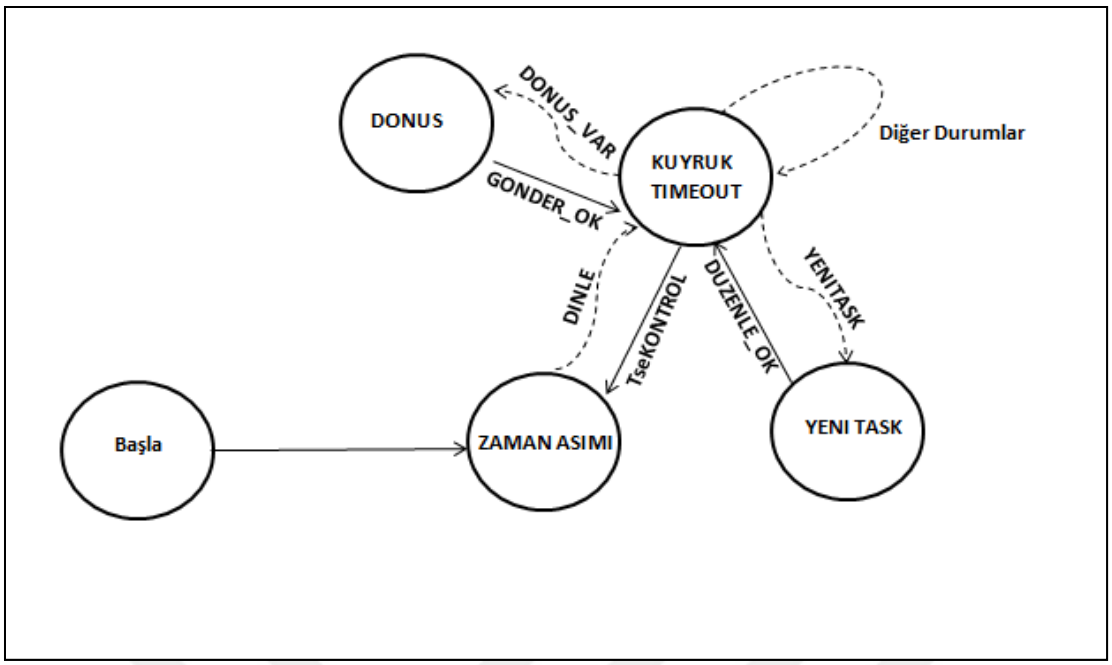


Şekil 5.25. FVWSN için genel simülasyon diyagramı

Görev Oluşturucu ve AddAndFollow birimlerinin yaptığı işlem, alınan talepler için sanal görevler oluşturmak ve bunları gerekli bilgilerle düzenleyip görev kuyruğuna bırakmaktır. Ön işlemci modülü "TALEPGELDI" şartı ile ilgili talebin sanal göreve dönüştürüldüğü duruma geçer. İstemcinin sisteme kayıt olma komutunun gelmesi durumda istemci verileri "REGISTER (KAYIT)" modülüne yönlendirilir. İç zaman kesmesinin aktif olması ile birlikte sanal görev bir sonraki modüle gönderilir. AddAndFollow modülünde gelen sanal görevler gerekli istatistik bilgiler ile ayarlanarak görev kuyruğuna bırakılır ve bunların durumları takip edilir. Bu modül aynı zamanda "Dönen Görev Listesi" 'ni de içermektedir. Modül ilk olarak sanal görevlerin kendilerine ayrılan süre içerisinde işlem görüp görmediklerini kontrol için zaman aşımı durumundan geçmektedirler. Bu kontrol belli aralıklarla devam etmektedir. Bu süreyi aşan sanal görevler ilgili istemcilere başarısız (RESEND) olarak bildirilirler. Görev sürelerini başarı ile tamamlayan sanal görevler, "DONUS" durumunda işlendikten sonra kendi istemcilerine bilgi vermek için ya M2M ya da Pub/Sub MW 'e gönderilirler. Yeni sanal görevlerin gelmesi durumunda bu süreçler tekrar etmektedir. Bu iki modüle ait çalışma FSM diyagramları Şekil 5.26 'da gösterilmiştir.



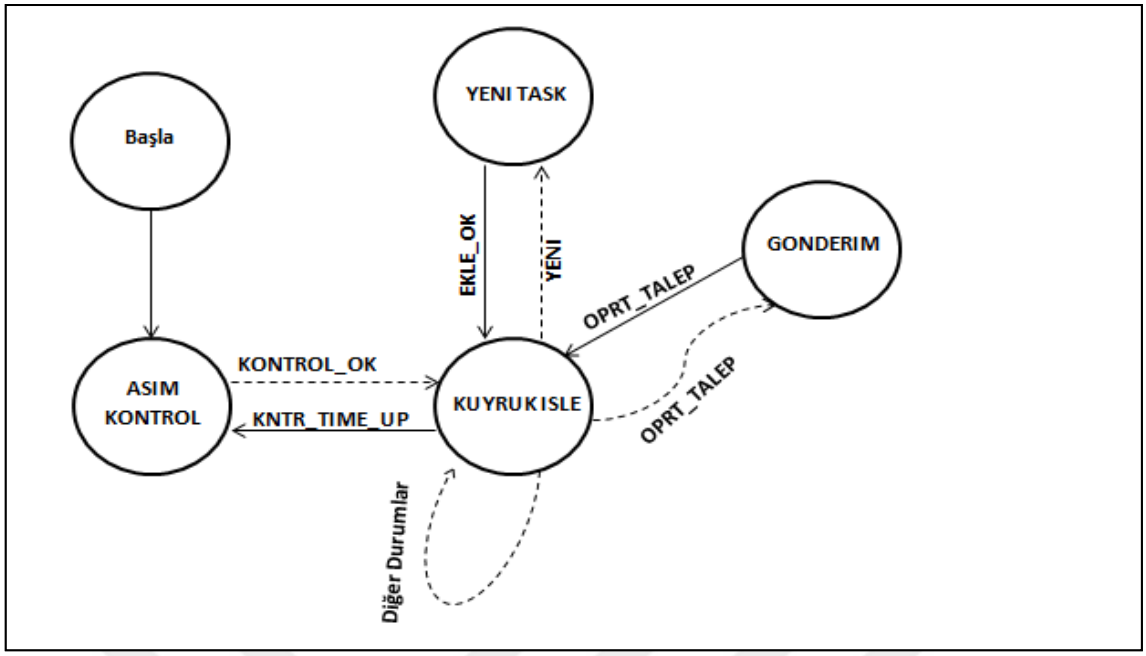
a) Ön İşlemci



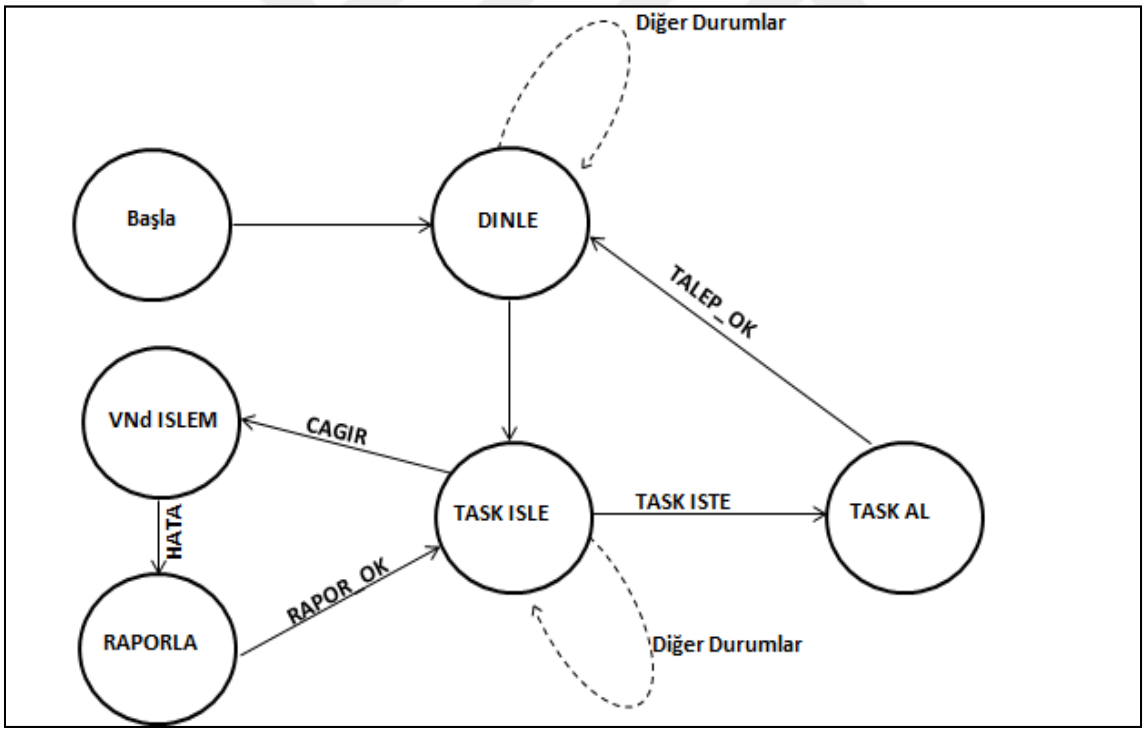
b) AddAndFollow

Şekil 5.26. Ön işlemci ve AddAndFollow FSM Diyagramları

Görev Kuyruğu modülü alınan sanal görevlerin "OperationRunner" a iletilmesinden sorumludur. Bunu yaparken Bölüm 3'de açıklanan kuyruk modelini uygulamaktadır. Kuyruğun tıkanması bir zaman çerçevesi ile izlenmektedir. Bu kontrol "ASIMKONTROL" durumunda gerçekleştirilmekte ve sistem kesmesi ile geçiş sağlanmaktadır. TaskQueue 'dan görev çıkartılması iki şekilde olmaktadır. Bunlardan birincisi zaman çerçevesi dışında kalmak, ikincisi ise OperationRunner 'dan kesme almaktır. İkinci durumda sıradaki sanal görev sonraki modüle gönderilir. OperationRunner modülü alınan sanal görevin detay bilgilerine göre ilgili VNd 'yi yürütmek ve bu VNd 'nin ihtiyaç duyduğu kaynak verilerini SerOP modülünden istemekle görevlidir. VNd operasyonları veya çağrımları esnasında oluşan hatalar raporlanmakta ve AddAndFollow modülü haberdar edilmektedir. OperationRunner modülü varsayılan olarak SYNC modunda çalışmaktadır. Görevi tamamlanan VNd hafızadan çıkarılmakta ve yeni görev kesmesi aktif edilmektedir. Bu süreçler "TASKAL" ve "DINLE" durumlarında gerçekleştirilir. TaskQueue ve OperationRunner modüllerine ait FSM diyagramları Şekil 5.27 'de verilmiştir.



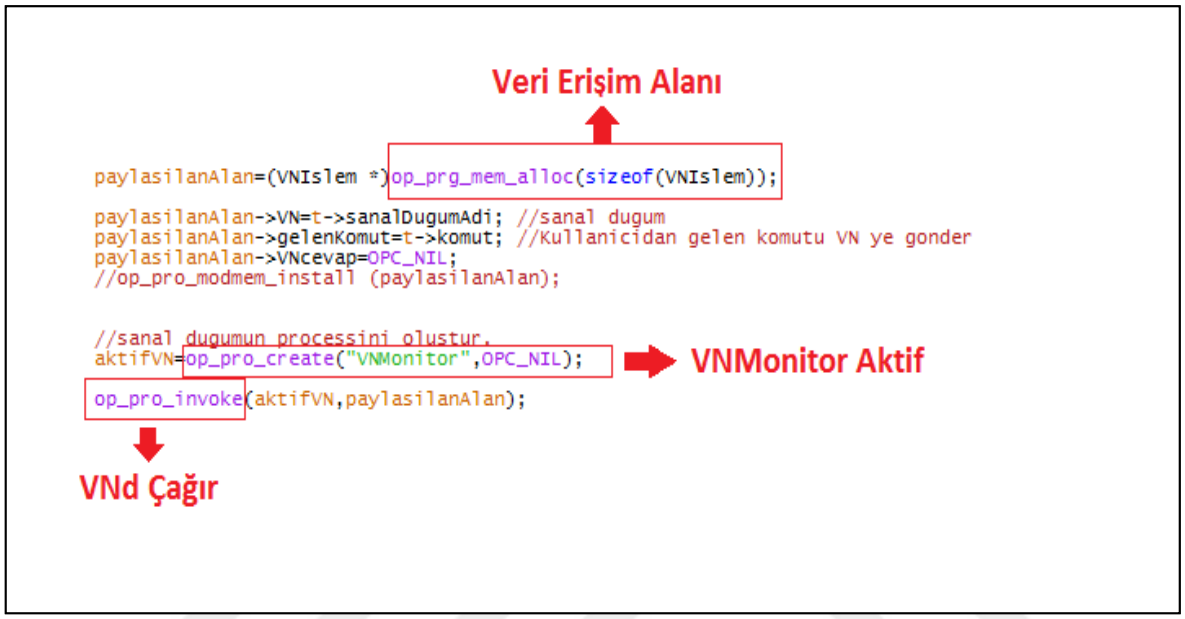
a) TaskQueue (Görev Kuyruğu)



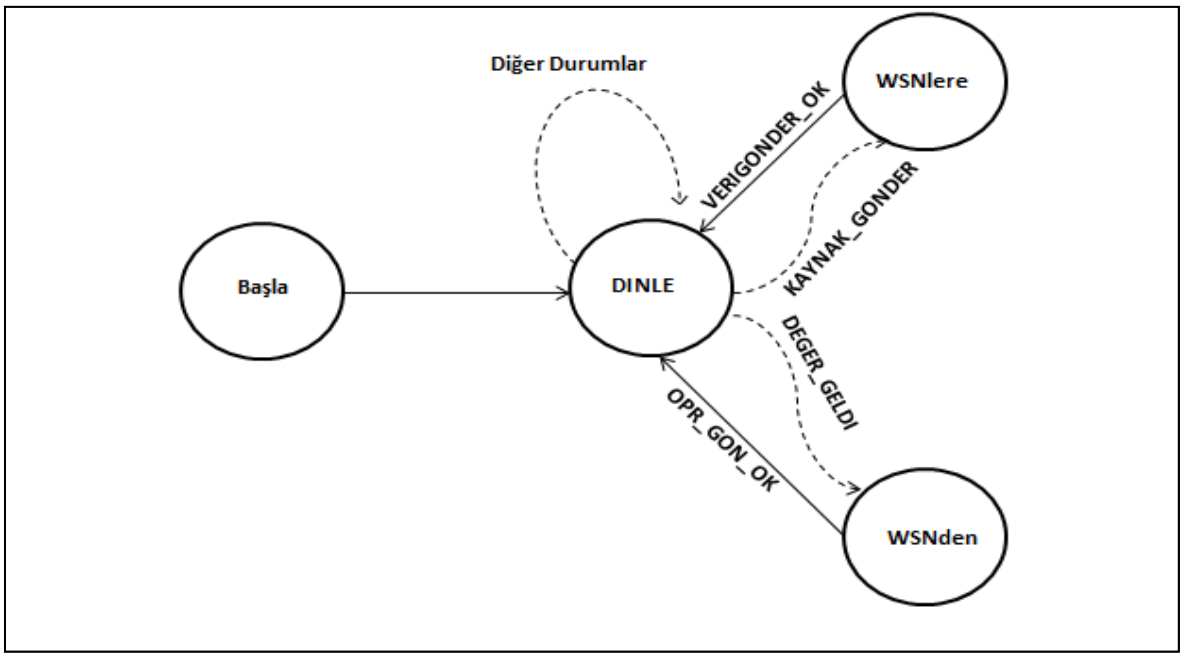
b) OperationRunner

Şekil 5.27. TaskQueue ve OperationRunner FSM Diyagramları

Bir VNd ‘nin OPNET Modeler programında nasıl çağrıldığı ve aktif edildiği Şekil 5.28 a ‘da gösterilmektedir. Bunun için önce veri paylaşım alanın tanımlanması ve VNMonitor ‘ün aktif edilmesi gerekmektedir. Daha sonra önceden oluşturulmuş VNd “child process” çağrımı ile ilgili VNd yürütümü gerçekleştirilir. Önerilen sistemin alt WSN ‘lerle olan bağlantısı ise SerOP ile sağlamaktadır. Bu modül alt WSN ‘lerden kaynak talep eden ve gelen verileri OperationRunner ‘a gönderen iki durumdan oluşmaktadır.



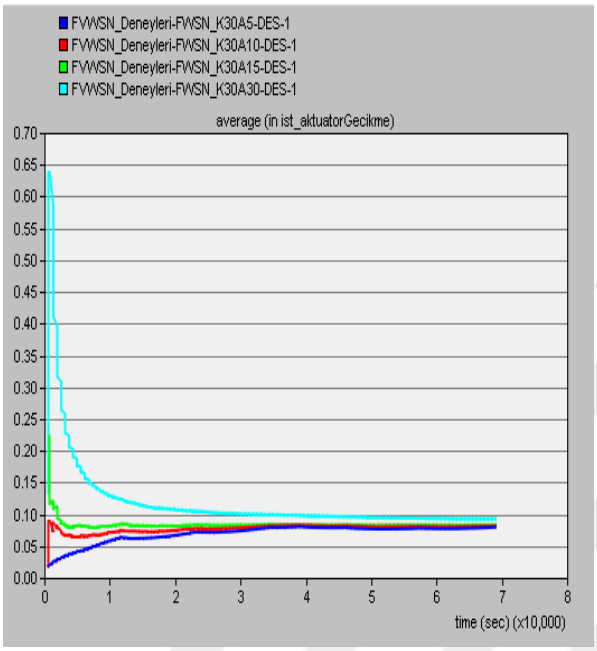
a) Child prosses VNd çağırma



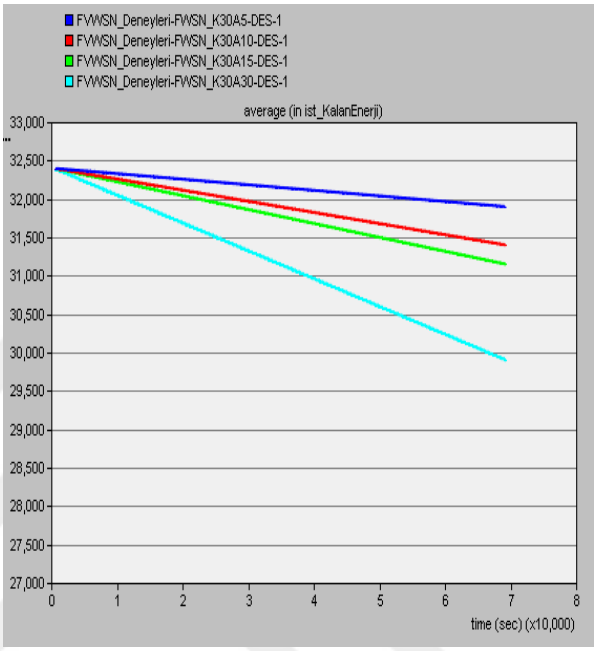
b) SerOP

Şekil 5.28. VNd çağrımı yöntemi ve SerOP FSM

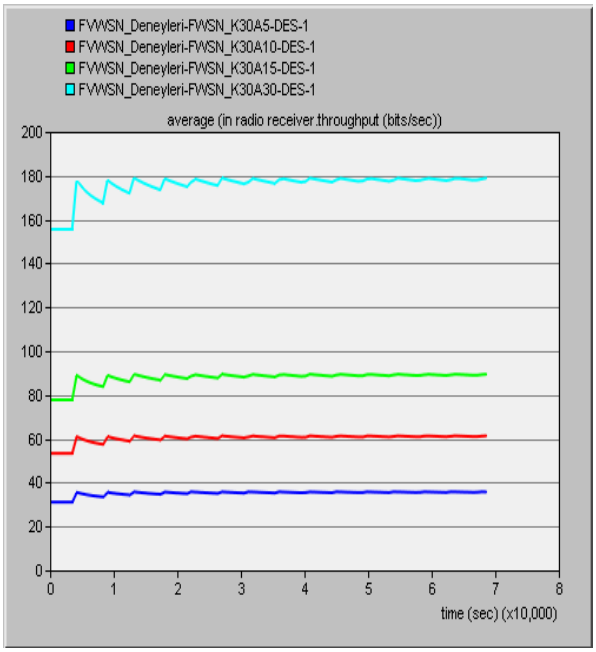
Önerilen sistemin OPNET Modeler ortamında yapılan simülasyonlarında kullanılan kod blokları Bölüm 3 ve Bölüm 4 ‘deki temel ilkelere göre hazırlanmıştır. Simülasyonlarda, numune alt WSN düğümlerinden alınan değerler için % 30 kaynak paylaşımlı sonuçlar Şekil 5.29 ‘da, % 50 kaynak paylaşımlı duruma ait sonuçlar Şekil 5.30 ‘da ve son olarak %100 kaynak paylaşımlı sonuçlar ise Şekil 5.31 ‘da sunulmuştur.



a) Uçtan uca gecikme



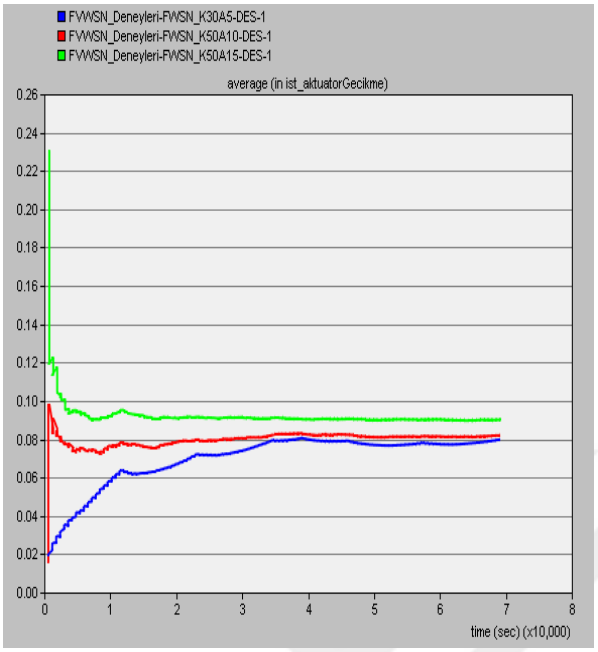
b) Enerji tüketimi



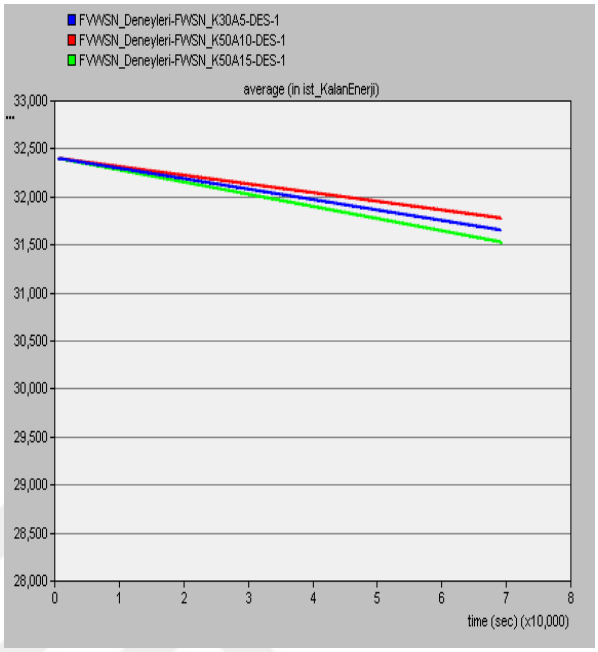
c) Ağ çıktısı

Şekil 5.29. İstemci :5-30, Kaynak paylaşım oranı %30

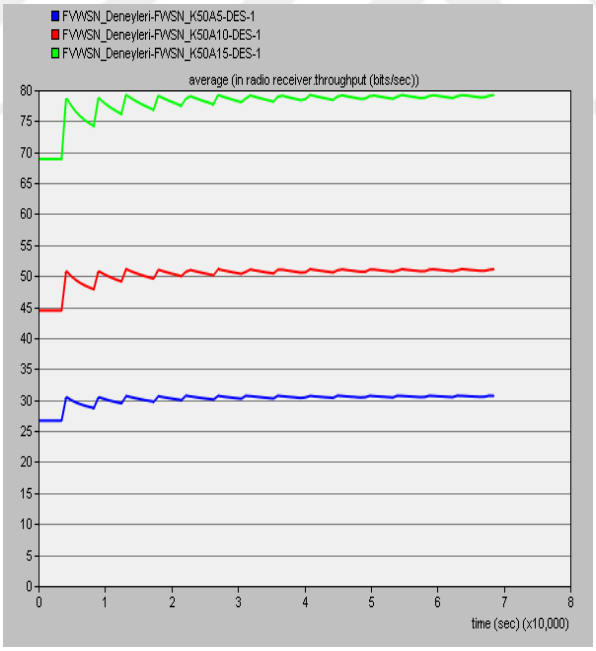
Kaynak paylaşım oranın %30 olması durumunda elde edilen değerlerin ABS temelli yöntemlere göre çok daha iyi olduğu Şekil 5.29 ‘da görülmektedir.



a) Uçtan uca gecikme



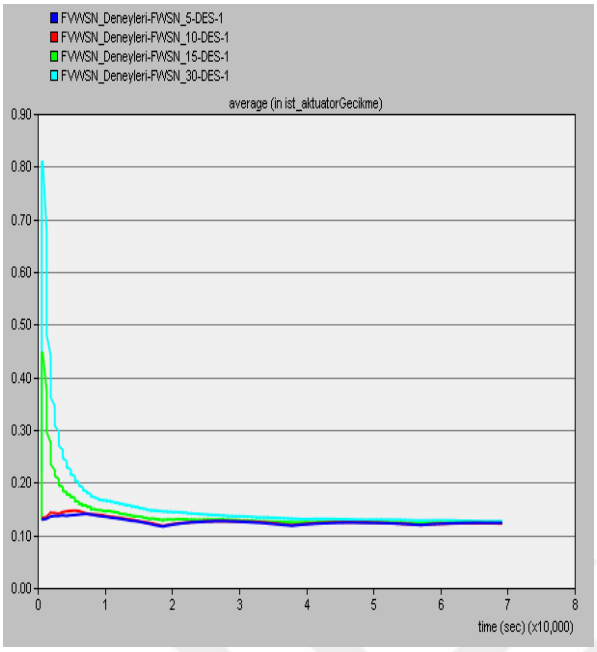
b) Enerji tüketimi



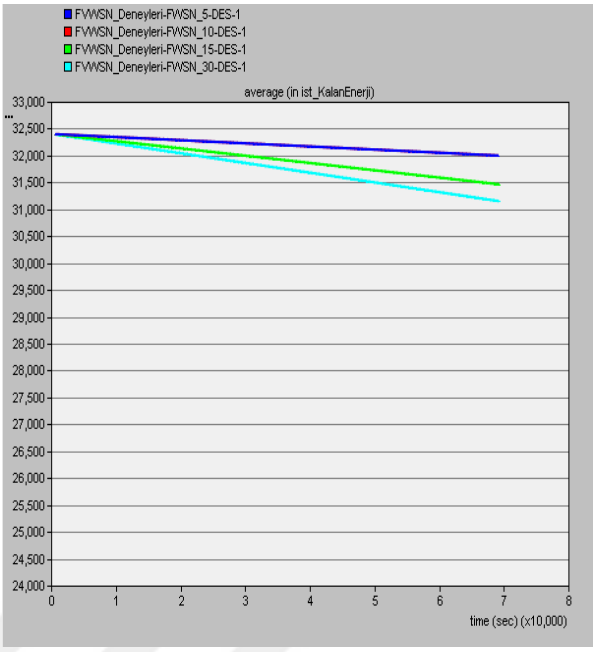
c) Ağ çıktısı

Şekil 5.30. İstemci :5-30, Kaynak paylaşım oranı %50

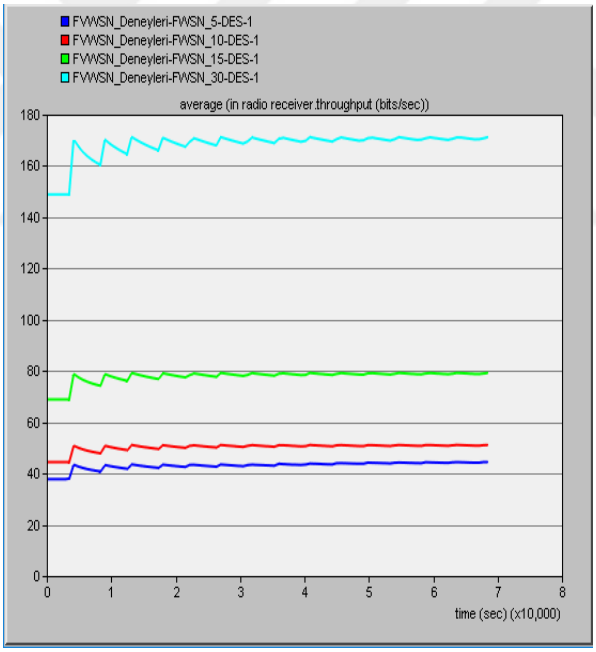
Kaynak paylaşım oranın %50 olması durumunda elde edilen değerlerin aynı kaynak paylaşım oranlı ABS temelli yöntem verilerine göre çok daha iyi olduğu Şekil 5.30 ‘da açıkça görülmektedir.



a) Uçtan uca gecikme



b) Enerji tüketimi

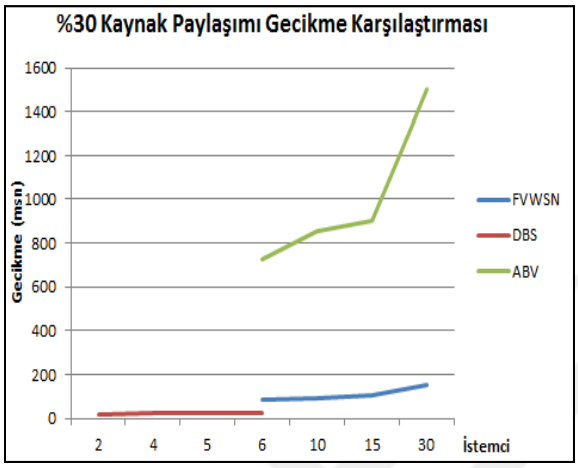


c) Ağ çıktısı

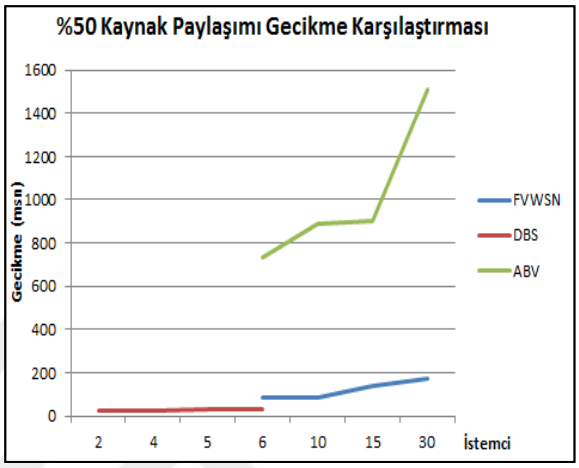
Şekil 5.31. İstemci :5-30, Kaynak paylaşım oranı %100

Bu başarım Şekil 5.31 ‘de görüldüğü üzere devam etmektedir. Bütün alt ağ metriklerinden alınan ortalama genel sonuçlar ise Şekil 5.32 ‘de verilmiştir. Aynı zamanda bu şekillere DBS ve ABS simülasyonlarındaki verilerde eklenerek genel bir mukayese ortamı sağlanmıştır. Kıyaslamaların daha net yapılması için her kaynak paylaşım oranı için grafikler ayrı ayrı verilmiştir. Şekillerden de görüleceği üzere FVWSN temelli ATBS sistem enerji tüketimi ile

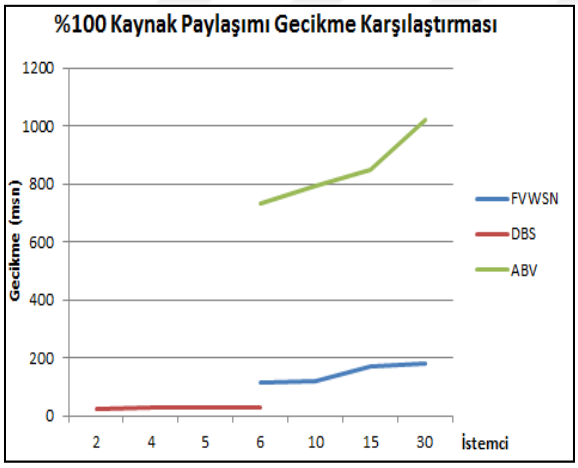
ABS ile hemen hemen aynı iken daha kısa cevap/etki süresine sahiptir. Bunun yanında ABS ve FVWSN ‘nin çok daha fazla istemciye daha az enerji tüketimi ile hizmet vererek bu konuda DBS ‘ye göre verimli oldukları gözlemlenmektedir. Öte yandan en iyi cevap/etki süresi ise DBS ‘ye aittir. Burada dikkat edilmesi gereken bir diğer nokta ise FVWSN ‘nin fonksiyon çağrımlarının istemci tanımlı olabilmesidir.



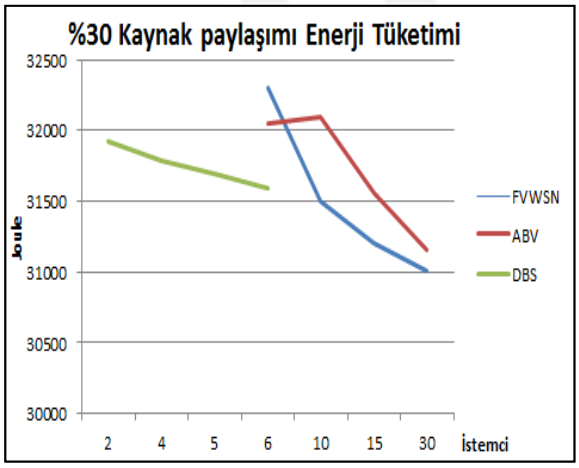
a) %30 Kaynak Paylaşımı Gecikme



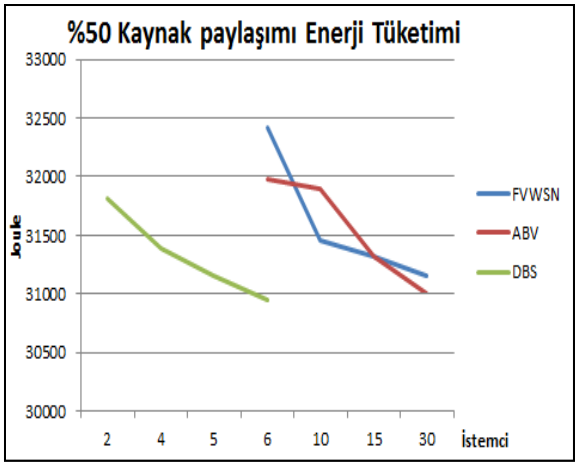
b) %50 Kaynak Paylaşımı Gecikme



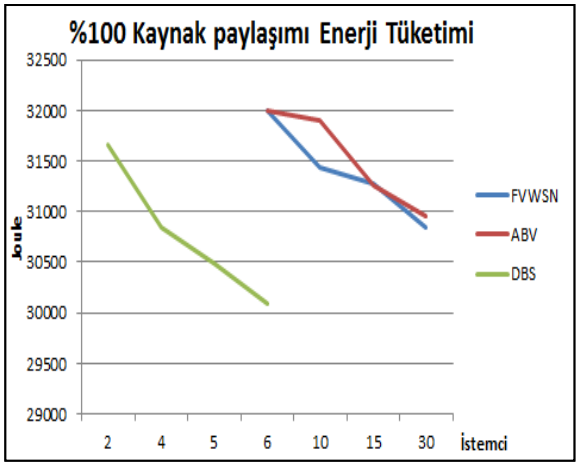
c) %100 Kaynak Paylaşımı Gecikme



d) %30 Kaynak Paylaşımı Enerji



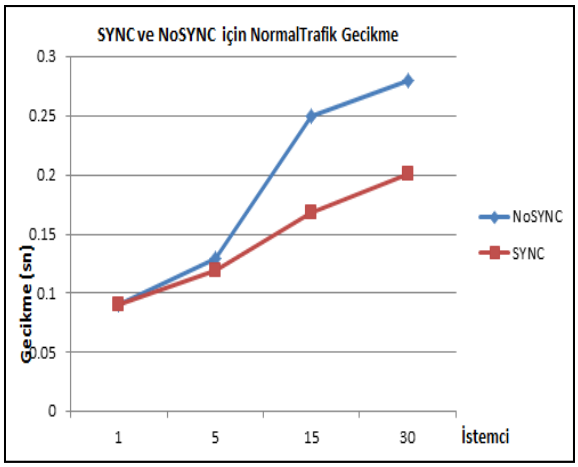
e) %50 Kaynak Paylaşımı Enerji



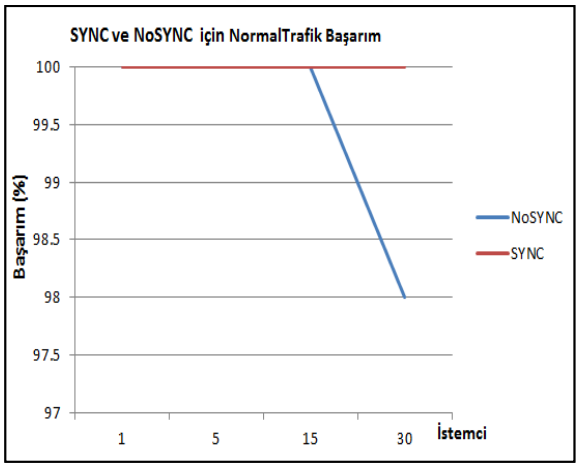
f) %100 Kaynak Paylaşımı Enerji

Şekil 5.32. DBS, ABS ve FVWSN sistem gecikme ve enerji tüketimi karşılaştırması

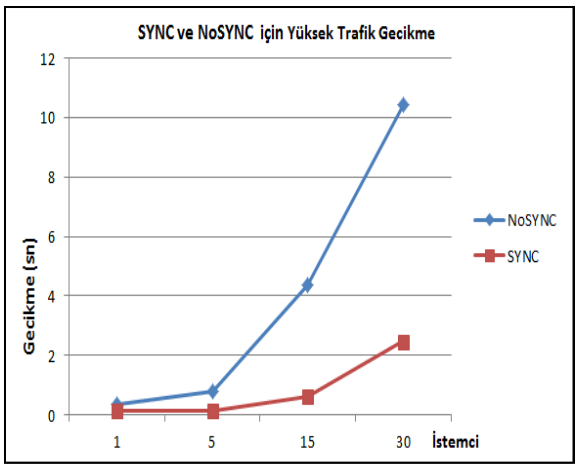
Önerilen sistemin bir diğer değerlendirme parametresi ise başarım oranıdır. Başarım oranı istemci tarafından yapılan ve başarı ile sonuçlandırılan talep sayısını göstermektedir. Bu kriteri ölçmek için %100 kaynak paylaşımı olan iki senaryo, OPNET Modeler 'da uygulanmıştır.



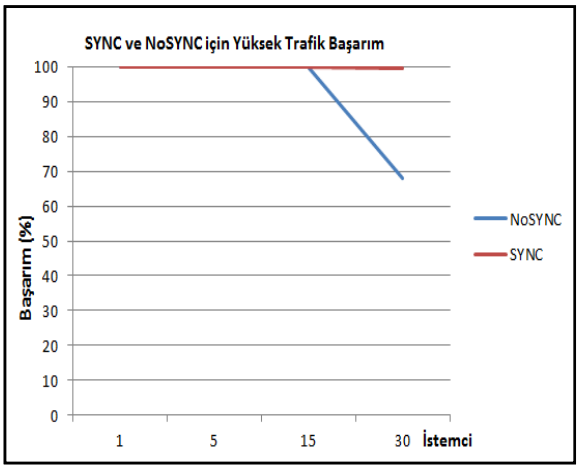
a) Normal trafik gecikme



b) Normal trafik başarım



c) Yüksek trafik gecikme



d) Yüksek trafik başarım

Şekil 5.33. FVWSN sistemin normal ve yüksek trafikte gecikme ve başarım sonuçları

Birincisi her istemci standart sapması 2 sn olan bir normal dağılımda gönderdikleri 100 talebin kaçının başarılı sonuçlandığı ve ne kadar sürede bu cevabı alabildikleri senaryodur (Normal trafik). İkincisi ise standart sapması 0.1 sn bir dağılıma sahip ve tüm istemcilerin aynı kaynakları talep ettiği bir yüksek trafikte başarım ve gecikme oranlarının tespitidir. Bu senaryolar hem SYNC hem de NoSYNC modu için ayrı ayrı denenmiştir. Bu simülasyonda $C_{sr}/R_{max}/T_{se}/T_r$ değerleri 3/2/10/4 dür. Şekil 5.33 normal ve yüksek trafik için elde edilen ortalama genel sonuçları göstermektedir.

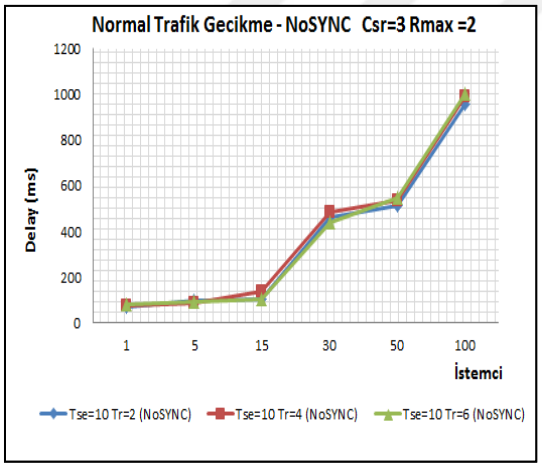
5.4. FVWSN temelli IoT/CPS WSN Sağlayıcı Sistemin Gerçek Ortamda Test Edilmesi

FVWSN 'nin pratik olarak test edilmesi Fırat Üniversitesi Kablosuz Sensör Ağları Laboratuvarında gerçekleştirilmiştir. JAVA 'da yazılan ve SOAP temelli Web Servis üzerinden çalışan FVWSN sanallaştırma yazılımı iki ayrı senaryo için UBUNTU Server 14.04 LTS işletim sistemli, i5 işlemcili ve 4 GB hafızalı bir bilgisayar üzerinde denenmiştir. Alt WSN olarak laboratuvardaki ZigBee düğümlerden, Libelium firmasına ait WaspMote, 802.15.4 temelli düğmelerden ve TelosB düğümlerden oluşturulan ağlar kullanılmıştır. Toplamda 100 adet VNd oluşturulmuş ve sisteme yüklenmiştir. Simülasyonda olduğu gibi ilk senaryoda normal şartlarda istemcilerin taleplerinin ne kadar sürede cevaplandığı ve başarım oranının ne olduğu incelenmiş, ikinci senaryoda ise her istemcinin arka arkaya eş zamanlı 100 talep gönderdiği yüksek trafikteki durumu gözlemlenmiştir. Test platformu parametreleri Tablo 5.2'de verilmiştir.

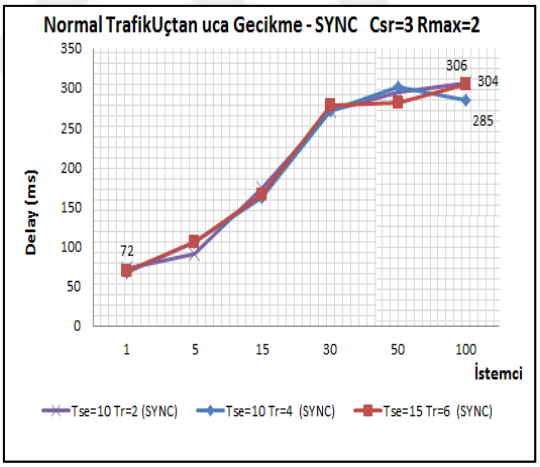
Tablo 5.2. Test platformu parametreleri

Parametre	Değer
C_{sr} *(Maksimum kaynak okuma sayısı)*	3
R_{max} *(Maksimum VNd kaynak sayısı)*	2
İstemci Tipi	Tip A
Toplam Kaynak ve Paylaşım Oranı	64 - %100
T_{se}*(Maksimum VNd yürütüm süresi)*	10
T_r*(Kaynağa erişim süresi)*	2/4/6

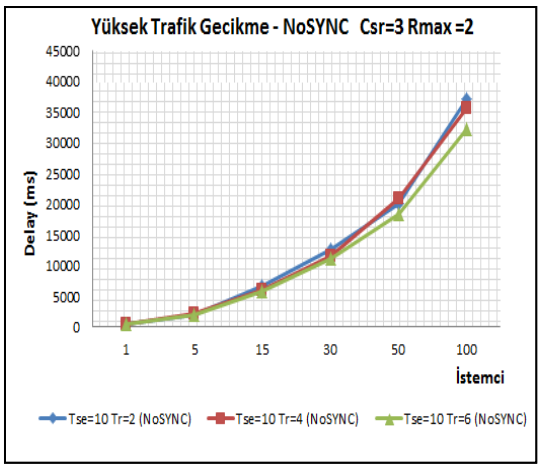
Şekil 5.34 ‘de, normal ve yüksek trafikte elde edilen test sonuçları verilmiştir. Bu sonuçlardan da görüleceği üzere sistemin pratik ortamda çalışması ve simülasyonda elde edilen değerler tutarlılık göstermektedir. Varsayılan durum modu olan SYNC özellikle istemci sayısındaki artışında başarım oranını ortalama %99 ‘larda tutmayı başarmıştır. NoSYNC durumları sadece öncelik derecesi yüksek olan veri paketlerinde kullanılmasına rağmen performans kriterlerini değerlendirme açısından bu testlerde denenmiştir.



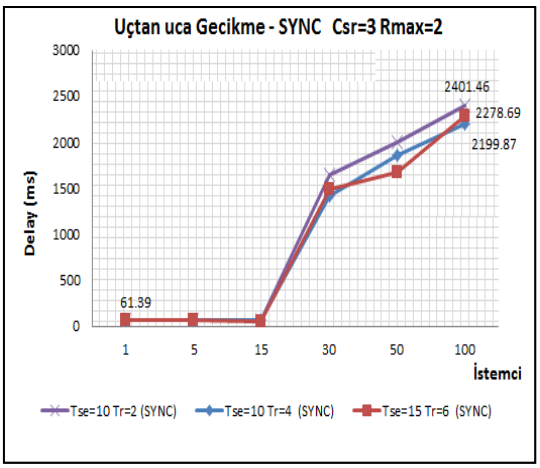
a) Normal Trafik Gecikme (NoSYNC)



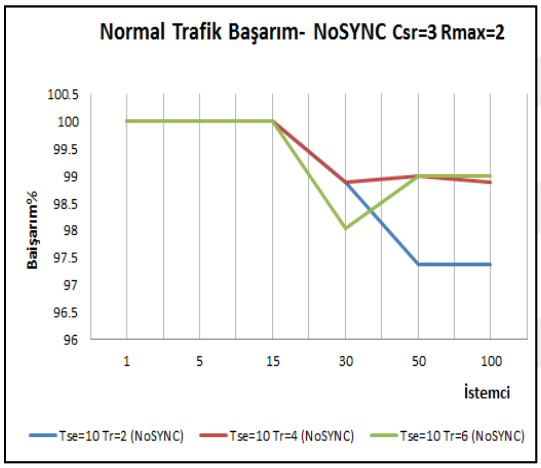
b) Normal Trafik Gecikme (SYNC)



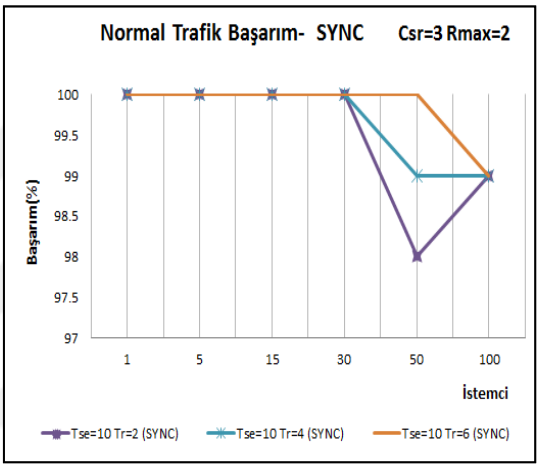
c) Yüksek Trafik Gecikme (NoSYNC)



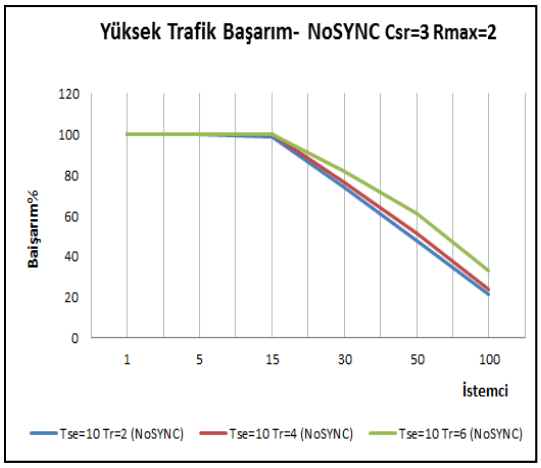
d) Yüksek Trafik Gecikme (SYNC)



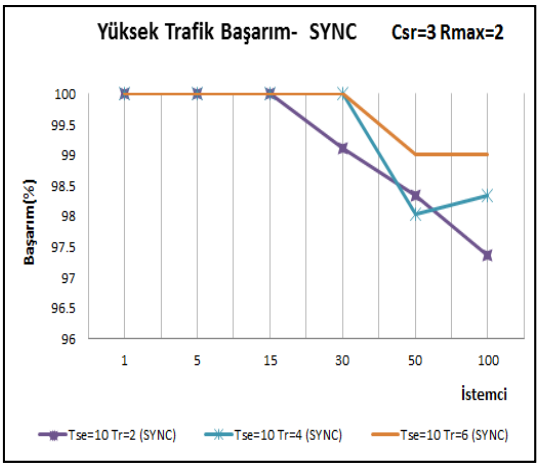
e) Normal Trafik Başarım (NoSYNC)



f) Normal Trafik Başarım (SYNC)



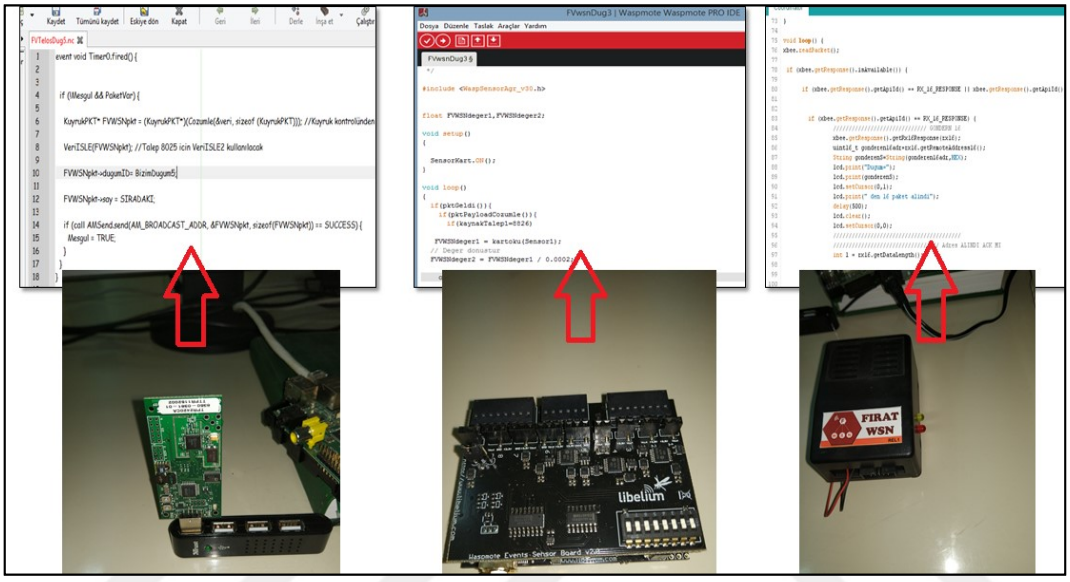
g) Yüksek Trafik Başarım (NoSYNC)



h) Yüksek Trafik Başarım (SYNC)

Şekil 5.34. FVWSN sistemin test sonuçları (T_{se} sabit, T_r değişken)

Önerilen sistemin SYNC modunda çalışması özellikle yüksek trafik durumunda zorunludur. Yüksek trafikte gelen taleplerin öncelik derecelerinin sıralanması ve gecikmenin artmaması için bazı öncelikli taleplerin SYNC modda değerlendirmesi gerekebilir. FVWSN framework ‘ünün pratik olarak test edilmesi başarı ile gerçekleştirilmiş ve temel sunucu yapısına entegre edilebileceği görülmüştür. Deneylerde kullanılan bazı düğümler ve kodlamalarından birer ekran görüntüsü Şekil 5.35 a ve b ‘de verilmiştir.



a) Düğüm tipleri ve kodları

b) ZigBee, Telos, Waspmote düğümleri

Şekil 5.35. WSN laboratuvarında pratik uygulama için kullanılan düğüm tiplerinden bazıları

6. BÜYÜK VERİ İŞLEYEBİLEN FVWSN TEMELLİ ATBS SİSTEM MİMARİSİ VE GERÇEKLEŞTİRİLMESİ

Bu bölüm, IoT ve CPS sistemler için önerilen büyük veri işleyebilen FVWSN temelli ATBS sistemin mimari detayları ve fiziksel gerçekleştirilmesinin nasıl yapıldığını içermektedir. Bu tez kapsamında önerilen sistemin temel olarak üç ana işlevi destekleyebilmesi amaçlanmıştır. Bunlardan birincisi, farklı heterojen WSN 'leri tek çatı altında birleştirerek istemcilere bu heterojen sistemlerdeki kaynakları hiçbir teknik alt yapı bilgisine ihtiyaç duymadan paylaştırabilmektir. Aynı zamanda arka taraftaki bu heterojen yapıyı tek bir sistem gibi gösterebilmektir. İkinci temel işlev, çok sayıda WSN kaynağından veya istemciden zaman içinde alınan/toplanan verilerin büyük veri boyutuna ulaşması durumunda gerekli büyük veri hizmetlerinin verilebilmesidir. Büyük verilerin önemli kaynaklarından olan WSN sistemler, sürekli olarak oldukça farklı veri tipleri içeren bilgi akışı sağlayabilirler. Bu veriler istemcilere doğrudan dağıtabilindiği gibi sistem üzerinde de depolanabilirler. Örneğin akıllı şehir, akıllı şebeke gibi büyük ölçekli sistemlerde uzun süreler boyunca ölçüm verileri toplanabilir ve belli bir zaman sonra toplanmış olan bu büyük ölçüm verileri istatistiki veya akademik olarak incelenebilirler. Buna ek olarak, sisteme sağlayıcılık yapan alt WSN sistemlerden gelen verilerin optimize veriler olmayacağı ve gereksiz/fazlalık veriler içerebileceği de dikkate alınmalıdır. Gereksiz veri önleme fonksiyonlarının büyük veri sistemleri ile birlikte kullanılması bu bağlamda önemli bir avantaj sağlayacaktır. Bu sebeplerden dolayı önerilen sistem, verimli bir büyük veri teknolojisi desteği de sağlamak zorundadır. Üçüncü işlev ise önerilen sistemin internet ölçeğinde ölçeklenebilir ve yürütülebilir olmasını sağlamaktır. Birinci işlev FVWSN ve yardımcı Pub/Sub arakatman yazılımı ile sağlanırken ikinci işlev ise günümüz büyük veri teknolojilerinden Hadoop ile başarılacaktır. Bahsedilen bu sistemlerin internet ve büyük CPS projeler ölçeğinde sürdürülebilir olabilmesi ise bulut teknolojisi kullanılarak kotarılacaktır. Önerilen sistem, birinci ve ikinci işlevleri, üçüncü işlev olan bulut teknolojisi içerisine alarak istemcilere farklı teknolojileri ve servisleri tek bir yapı altında sunabilecektir.

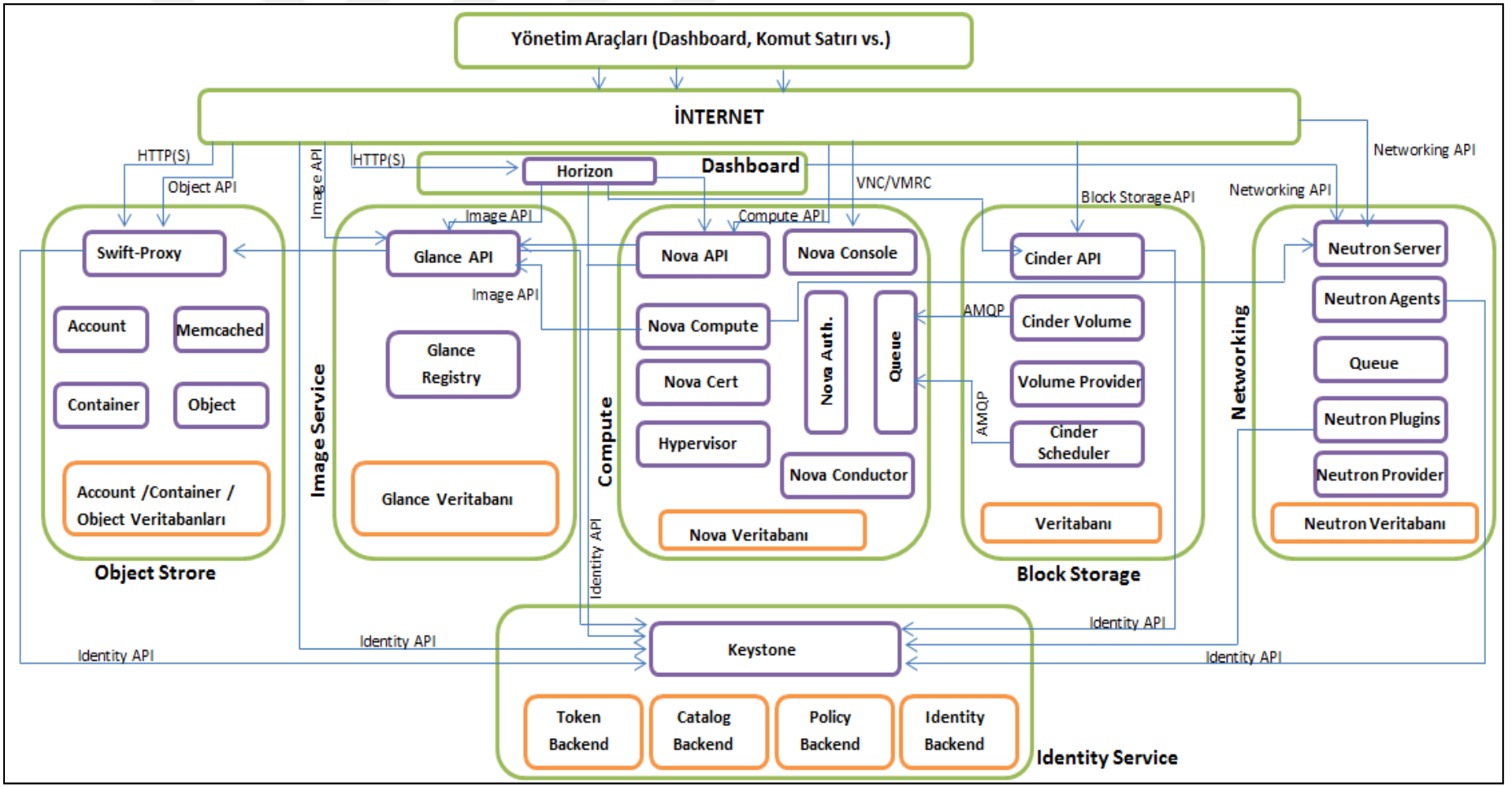
Bilindiği üzere bulut teknolojileri farklı donanımsal kaynakları, tam bir şeffaflıkla kendi yapısı altında bir araya getiren, onları yönetebilen ve farklı amaçlar için paylaşılabilir olmalarını sağlayan sistemlerdir. Bulut teknolojileri bu hizmetleri "Bulut İşletim Sistemleri" yardımıyla yapmaktadırlar. Önerilen IoT/CPS WSN sağlayıcı sistem, her ne kadar bireysel

donanımlar üzerinde çalışabilecek şekilde tasarlanmış olasa da asıl amaç onun bir bulut sistem hizmeti olarak çalışabilir olmasını sağlamaktdır. Bu nedenle önerilen sistem, oluşturulan bir bulut sistemine entegre edilmiştir. Geliştirilen bulut sistem için kullanılan işeltim sistemi OpenStack ‘dir [125]. Önerilen sistem, bu büyük veri desteğini geliştirilen bulut sisteme entegre ettiği Hadoop teknolojisi ile sağlamaktadır. Hadoop ‘un sağlamış olduğu hata tolerans, MapReduce framework ‘u ve diğer dağıtık veri işleme teknolojileri OpenStack ile tam bir uyum içerisinde çalışabilmektedir.

Bu bölümde öncelikle OpenStack yapısı ve servisleri hakkında kısa bilgi verilecek daha sonra sistemin fiziksel detayları, FVWSN framework, Hadoop, gereksiz veri optimizasyon fonksiyonu ve Pub/Sub arakatman yazılımı ile entegrasyonu hakkında bilgi verilecektir.

6.1. OpenStack İşletim Sistemi

OpenStack büyük donanımsal ve ağ kaynaklarını (işlemci, hafıza, depolama vb.) yönetebilen, bu yönetilebilen kaynaklar ile paylaşımlı servisler sunulmasına imkân veren bir bulut işletim sistemidir. OpenStack, dünya genelinde binlerce geliştiricinin destek verdiği ve 150 ‘nin üzerinde ülkede yaygın bir şekilde kullanılan bir teknolojidir. Pek çok teknoloji devi ve büyük proje yürütümleri bu platformun sağladığı avantajlardan faydalanmaktadır. Hypervisor olarak KVM, Xen, VirtualBox, VMware ve Hyper-V teknolojilerini desteklemektedir. OpenStack alt yapı hizmetlerini bir biri ile etkileşimli olan servisler vasıtasıyla gerçekleştirir. Bu etkileşim ise uygun API ‘ler ile sağlanmaktadır. Her ne kadar kurulacak servisler uygulama tipine göre seçilse de bazı temel servisler, bütün uygulama tiplerinde bulunmak zorundadır. Şekil 6.1 OpenStack ‘de ki genel servis mimarisini göstermektedir. Dashboard servisi olan Horizon web tabanlı bir kontrol portalı sağlamaktadır. Bu servis sayesinde diğer temel servislere erişim, konfigürasyon ve yönetimsel işlemler yapılabilmektedir. Hesaplama (Compute) servisi Nova en önemli alt yapı servislerindendir. Tüm “compute instance” ların yönetiminden sorumludur.



Şekil 6.1. OpenStack genel servis mimarisi

Talep üzerine oluşturulan sanal makinaların yürütüm düzenlemeleri bu servis üzerinden gerçekleştirilir. Ağ yönetim servislerinden sorumlu olan Neutron, OpenStack servisler tarafından yönetilen arayüz aygıtları arasında "network connectivity as service" hizmetlerini sağlamaktadır. Swift bir nesne depolama servisidir (Object Storage Service). Web servislerden veya http temelli API 'lerden alınan yapısal olmayan (unstructured) veri nesnelerinin depolanmasını/yönetilmesini sağlar. Replikasyon yoluyla yüksek bir hata toleransı sağlamaktadır. Cinder, block storage aygıtların yönetilebilmesini sağlayan servistir. OpenStack 'in en kritik servislerinden olan Keystone ise bütün servisler için yetkilendirme ve yetki kontrol servislerinin yürütümünü gerçekleştirmektedir. Glance servisi, sanal makine disk imajlarının depolanması ve alınması hizmetlerinden sorumludur. Compute servisi, bu servisi "instance" işlemlerinde sıklıkla kullanmaktadır. Bu temel servislerin dışında OpenStack 'in sahip olduğu farklı servislerde bulunmaktadır. Bu servislerden Ceilometer, OpenStack için faturalandırma, benchmarking, ölçeklenebilirlik ve diğer istatistik verilerin gözlemlenmesinde kullandığı hizmettir. Heat servisi birden fazla yürütülen bulut uygulamalarının yönetilebilirliğini düzenleyen hizmetleri sağlarken, Trove servisi, veritabanı motorları için "Cloud Database as a Service" fonksiyonlarını sunar. Hadoop cluster 'larının ölçeklenmesine yönelik temel parametreler Sahara servisi ile kotarılmaktadır [125].

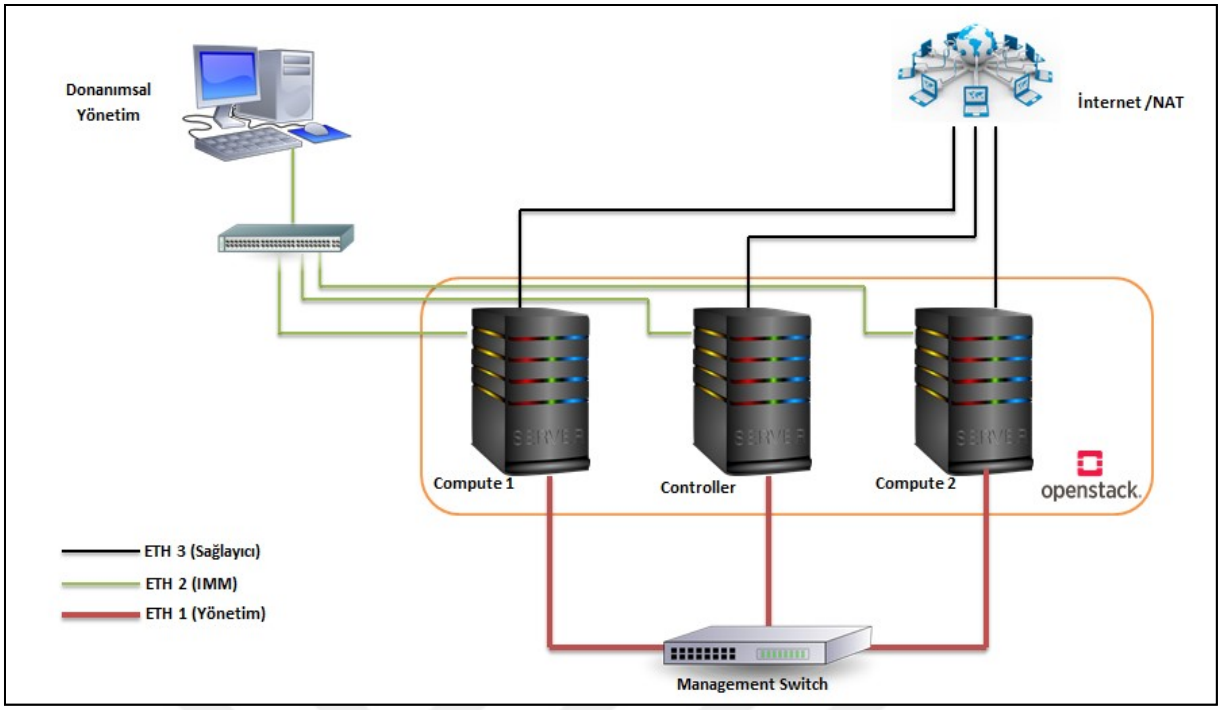
OpenStack 'de bütün servisler, ortak kullanılan Keystone servisi ile yetkilendirilmektedir. Özel yönetici komut yürütümleri hariç, OpenStack servisleri bir birleri ile public API 'ler vasıtası ile etkileşmektedirler. Bütün bu servisler birkaç alt process 'den oluşmaktadırlar. Bu process 'ler API taleplerini dinlemek, ve onları gerektiğinde diğer servislere aktarmakla görevlidirler. Bir servisin process 'leri arasındaki iletişim için AMPQ mesaj broker kullanılmaktadır. Kullanıcılar OpenStack e' web tabanlı bir kullanıcı ara yüzü olan Horizon ile erişebilmektedir. Ancak bu tek seçenek olmayıp farklı browser API 'leri ve komut satırı üzerinden de bu erişim gerçekleştirilebilmektedir. Aynı zamanda bulut üzerindeki uygulamalar için kullanılabilecek Java, Python ve Node.Js gibi platformlar için SDK 'ları da sağlanmaktadır.

Bu tez çalışmasında OpenStack bulut işletim sistemi üç adet sunucu üzerinde çalıştırılırmıştır. Bu sunuculara ve diğer donanımlara ait teknik özellikler Tablo 6.1 'de verilmiştir. Sunucu bilgisayarlar Ubuntu 16.04 LTS işletim sistemi çalıştırmaktadır. Fiziksel sunuculardan biri Controller diğer ikisi ise Compute sunucu olarak kullanılmak üzere ayarlanmıştır.

Tablo 6.1. Fiziksel donanımlara ait teknik özellikler

Parametre	**Özellik**
Sunucu	3 x Lenovo x3650
İşlemci	Intel Xeon
İşlemci Hızı (GHz)	2.4 (+2.4)
Core	6
Thread	12
L1 / L2/ L3	12 KB / 34 KB / 15360 KB
Güç Kaynağı (adet)	2
Sabit Disk Kapasitesi (GB)	3 x 600
RAID	1
Hafıza (GB)	48 / 32 / 16 (Controller)
RAM Hız	2400 MHz
Uzaktan Yönetim Modülü	IMM
Kasa	2U
İşletim Sistemi	Ubuntu 16.04. LTS
Bulut İşletim Sistemi	OpenStack Newton
Ethernet Port	4
Compute Sunucu	2

Sistemin donanımsal teknik blok yapısı Şekil 6.2 ‘de verilmiştir. Şekilden de görüleceği üzere sistem sunucuları üç adet Ethernet çıkışını kullanmaktadır. Bunlardan ilki Controller ve Compute sunucular arasındaki iç trafiği/yürütümü sağlayacak olan ağ çıkışıdır. Bu ağ bir “Management Switch” üzerinden oluşturulmuştur. Bu ağda, sunucular arasında “heart beat” kontrolleri, instance operasyonları gibi iç yürütümler sağlanmaktadır ve bu ağda “private IP” ler kullanılmaktadır. İkinci Ethernet çıkışı yine yönetimsel amaçlıdır ve IMM desteğinden faydalanılmak için kullanılır. Bu da “private IP” ile çalışmaktadır. Son Ethernet bağlantısı ise internet bağlantısının sağladığı sistem çıkışıdır. Sağlayıcı sistemin internet çıkışı bu çalışmada NAT ile gerçekleştirilmiştir.



Şekil 6.2. Bulut sistemin fiziksel kurulum blok şeması

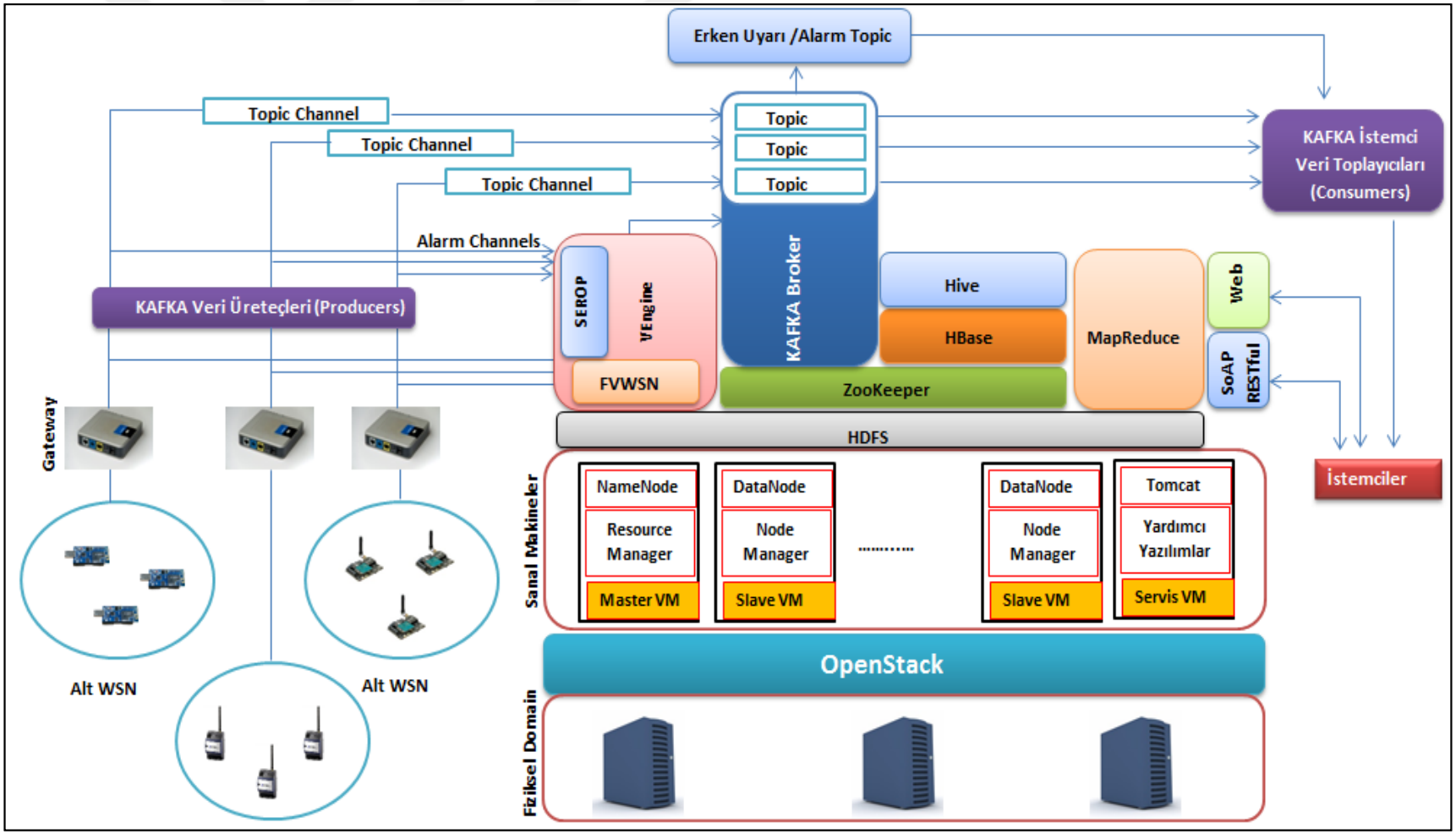
Önerilen sistemin gerçekleştirilmesinde öncelikle OpenStack işletim sisteminin kurulumu yapılmıştır. Sistem yapısı ve planlaması bakımından bütün OpenStack servisleri kurulmamıştır. Kurulumda öncelikle Neutron, Nova, Glance ve Sahara servislerinin kurulumu gerçekleştirilmiştir. Dahili servisler arasındaki haberleşme için RabitMQ tercih edilmiştir. Bununla birlikte dahili servis veritabanı olarak MariaDB kullanılmıştır. Sistemde Trove, Heat ve Swift servisleri planlama aşamasında olmadığından ve gerekli donanım eksikliklerinden dolayı kurulmamıştır. Kurulum esnasında Hypervisor olarak varsayılan teknoloji olan KVM kullanılmıştır.

OpenStack işletim sisteminin fiziksel sistem üzerine kurulumu tamamlandıktan sonra IoT-WSN bulut sistemin mimarisi oluşturulmuştur. Bunun için OpenStack üzerinde 9 adet sanal makine oluşturulmuştur. Bu makinalar kendi aralarında bir internal ağ ve bir Router yardımıyla da external ağa bağlanmışlardır. Ubuntu 16.04 işletim sistemi üzerinde çalışan bu sanal makineler, Hadoop ekosisteminin yürütümünü, FVWSN framework ‘unun entegrasyonunu ve Pub/Sub arakatman yazılımının işletilmesini sağlamak için kullanılmıştır. Bunun için öncelikle master sanal makinede “NameNode” oluşturulmuştur. Bu node, Hadoop dosya sistemi olan HDFS ‘in ana parçasını oluşturmaktadır. Bütün istemci işlemleri ve metadata ‘ların yürütümünden bu düğüm sorumludur. NameNode ‘un hata vermesi veya olası olumsuzluklar durumunda sistemin işleyişinin bozulmaması için “ikincil (secondry)

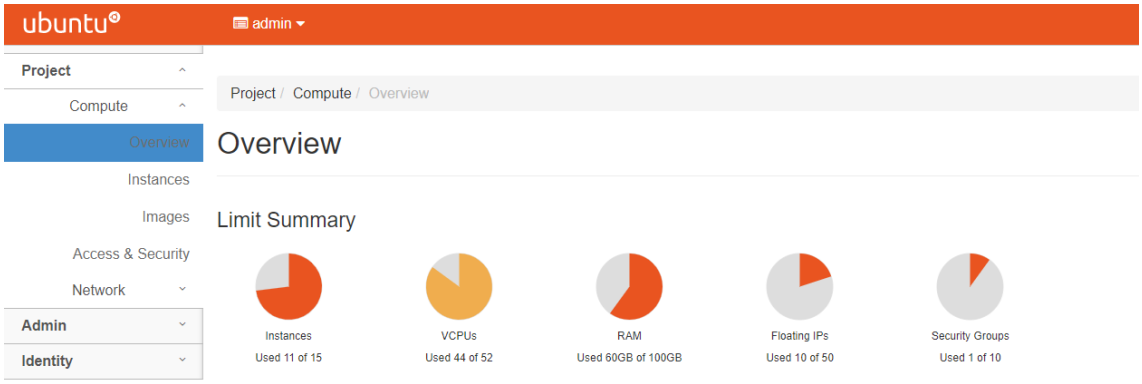
NameNode" oluşturulmuştur. Bu düğüm ise sürekli birincil NameNode ile etkileşim halinde olup senkronizasyonu tüm işlem yürütümlerinde yeniden sağlamaktadır. Hadoop sisteminin bir diğer düğümü ise master sanal makine üzerine kurulan "ResourceManager" dır. Bu node, diğer slave NodeManager deamon 'larının I/O görevlerinin yürütümünü düzenlemekle görevlidir. "DataNode" düğümler "slave" olarak tanımlanırlar ve HDFS üzerinde veri depolayan düğümlerdir. Bu makineler, "ResourceManager" ile iletişim kuran "NodeManager" proseslerini çalıştırmaktadır.

Şekil 6.3, önerilen sistemin genel çalışma yapısını göstermektedir. Bu şekilden de görüleceği üzere önerilen sistem yapısında FVWSN, Pub/Sub ara katman yazılımı, büyük veri teknolojisi Hadoop ve diğer yardımcı servisleri OpenStack üzerinde yürütmektedir. Sisteme kayıtlı olan istemciler hizmetleri üç şekilde alabilmektedir. Bunlardan ilki VNd yönetimi için SoAP/Restful protokolü ile erişimdir. İkincisi Pub/Sub ile veri alımı üçüncüsü ise genellikle kayıt yönetimi, VNd kontrolü ve büyük veri analizi için web arayüzünün kullanılmasıdır. Alt WSN sistemler gateway 'ler üzerinden sisteme veri akışı sağlamaktadır. Alınan veriler Kafka veri üretim fonksiyonu ile ilgili istemcilere doğrudan gönderilmektedir. Bu veriler aynı zamanda özel yazılmış alarm VNd 'ler tarfından kullanılabilirler. Alarm durumlarının tespitinde ise VNd 'ler kendi Kafka veri üretim fonksiyonları yardımıyla ilgili istemcileri bilgilendirmektedir. Alt WSN kaynaklara ulaşım ise VEngine tarafından sağlanmaktadır. VEngine 'deki VNd yürütüm sonuçları web servis veya yine kafka üzerinden istemcilere ulaştırılmaktadır. Veri akışları Hadoop üzerinde tutulmaktadır. Tutulan veri tipleri hem düz text dosya formatı veya istenilmesi durumunda HBase üzerinde depolanabilmektedir. HBase üzerinde tutulması ile Hive fonksiyonlarıda kullanılabilmektedir.

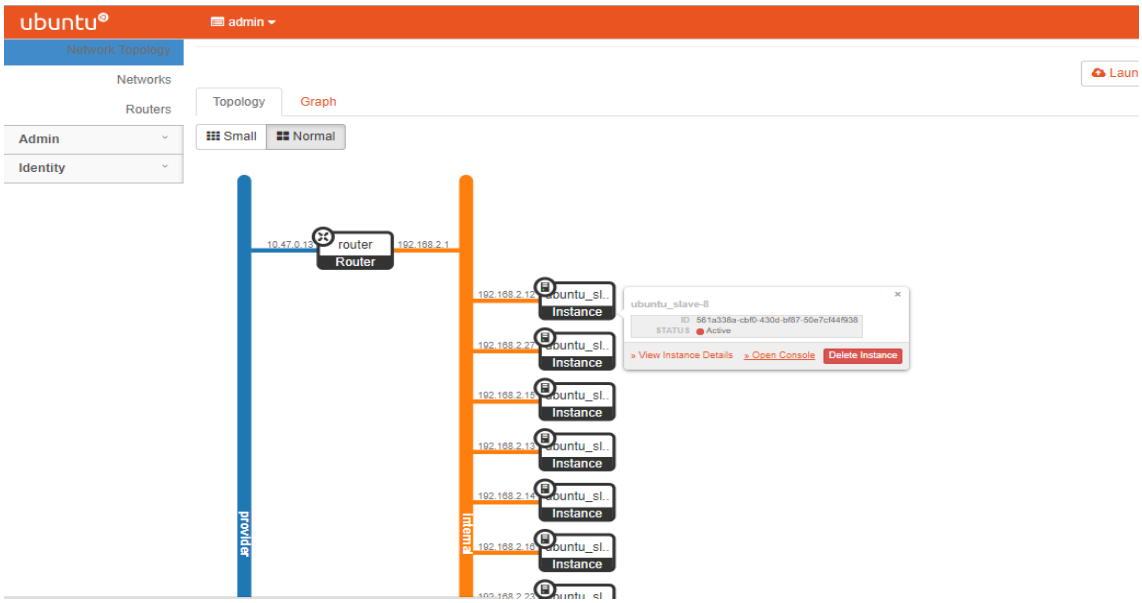
Şekil 6.4 ise belli bir fikir vermesi bakımından genel kurulum adımlarından sonra OpenStack 'e ve Hadoop ekosistemine ait sistem bilgilerini göstermektedir. Önerilen sistemin çalışma yapısından da görüleceği üzere FVWSN sistem hem Pub/Sub ara katman yazılımı ile hem de MapReduce ile etkileşim halindedir. Bu FVWSN kullanmanın bir diğer avantajını göstermektedir. Bunun anlamı VEngine sonucunda elde edilen veriler bir "topic" olarak Pub/Sub 'a sağlayıcılık yapabilirken aynı zamanda tanımlı bir MapReduce uygulaması ile de etkileşebilmektedir.



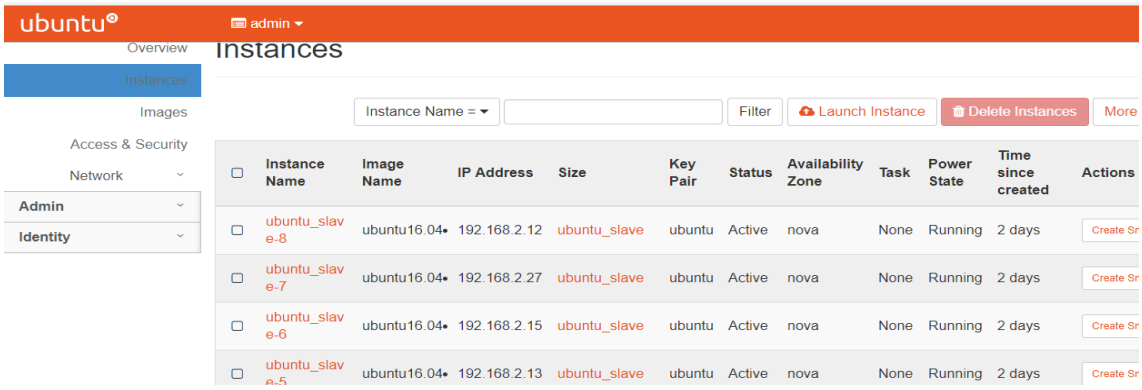
Şekil 6.3. Önerilen IoT-WSN sistemin modüler çalışma yapısı



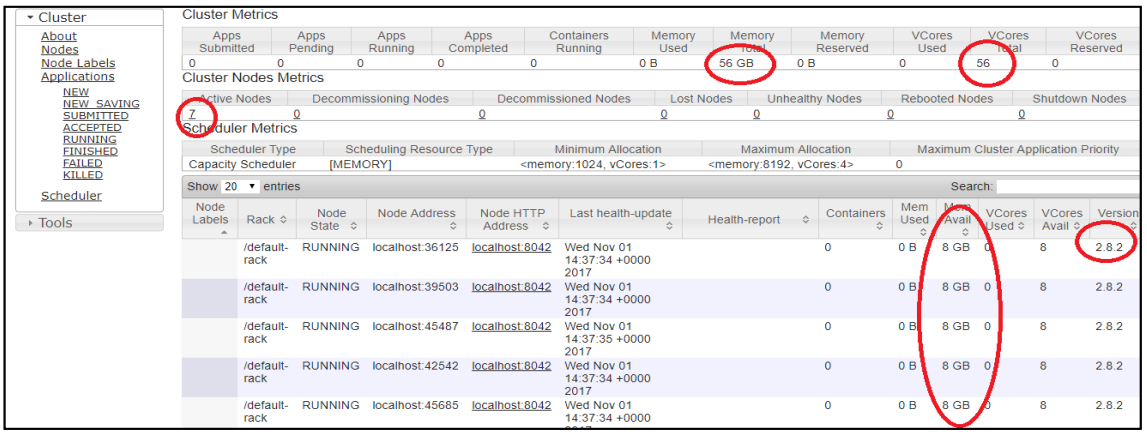
a) OpenStack kurulumundan sonra genel kaynak durumu



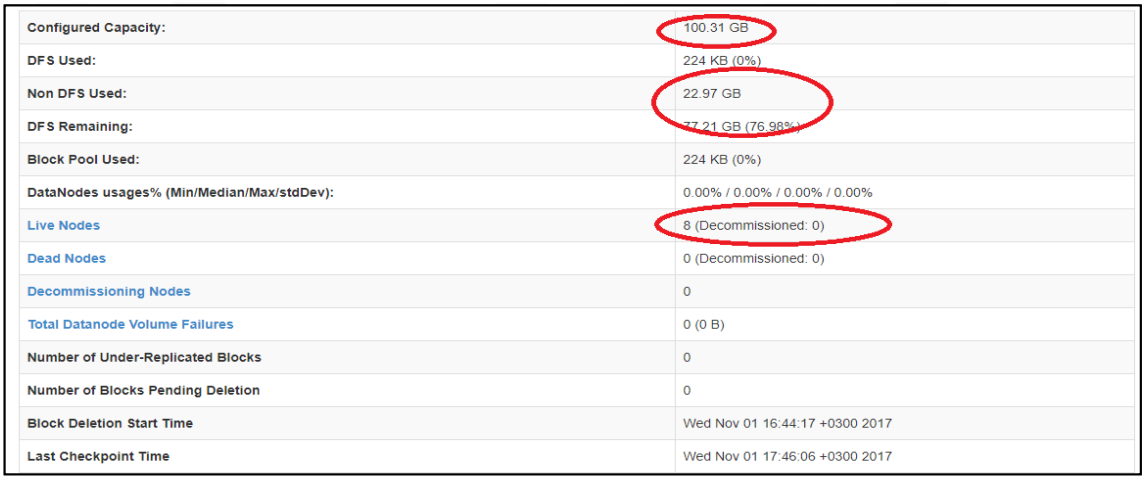
b) Sanal Makinalar ve oluşturulan ağ yapısı



c) Sanal makine imajları



d) Hadoop Cluster



e) Hadoop node bilgileri

f) Hadoop Datanode kullanım histogramı

Şekil 6.4. OpenStack ve Hadoop kurulumu sonrası genel durum verileri

Önerilen sistem veri dağıtma/toplama yöntemlerinden biri olarak "topic-based publish-subscribe" teknolojileri kullanmıştır. Basit yapıları, düşük başlık verileri ve stabil olmalarından dolayı sıklıkla tercih edilen bu teknolojiler, konu (topic) ve bu konuya abone olan istemciler arasında veri aktarımı sağlarlar. Bugün, RabbitMQ [127] ve Kafka [128] gibi iyi bilinen pub/sub Midleware 'ler bulut sistemlerde sıklıkla tercih edilmektedir. Bu çalışmada sıfır hata toleransı ile öne çıkan Kafka broker tercih edilmiştir. Kafka TCP üzerinden bir binary protokol kullanmaktadır. İstemciler bir soket bağlantısı kurarak tüm istek/cevap operasyonlarını bu bağlantı üzerinden yürütmektedirler. Kafka sadece byte mesajları desteklediğinden, bu çalışmada alt WSN kaynak verileri ilgili metadata verileri de içeren bir JSON mesaj formatı kullanmaktadır. Başlangıç olarak her bir alt WSN için bir topic mevcuttur. Ancak bu zorunlu olmayıp istemci taleplerine veya çalışma senaryolarına göre farklı topic 'ler oluşturulabilmektedir. Önerilen sistemdeki Pub/Sub mesaj dağıtımını diğerlerinden ayıran özellik istemci VNd 'lerinin belirli topic 'lere abone olarak erken uyarı/alarm takiplerini yapabilmeleridir. Anlık veri akışının VNd 'lere aktarılarak istemci tanımlı analizlerine imkan vermeleri ve bu erken uyarı VNd 'lerinin sonuçlarının alarm topic 'lerine gönderilmesi ile daha esnek bir erken uyarı sistemi modeli elde edilmiştir.

Bu çalışma kapsamında alt WSN sistem verilerinin mesage broker 'a gönderilmesini sağlayacak ve yine istemcilerin abone oldukları topic 'lerden verileri alabilmelerini sağlayacak "FKaf_Producer" ve "FKaf_Consumer" yazılımı geliştirilmiştir. Bu iki basit yazılımdan ilki gerekli Kafka API 'lerinin sistem içerisinde kullanılabilir olmasını sağlarken, diğeri istemcilere görsel bir arayüz sağlayarak veri analizine imkan sağlar. Şekil 6.5 'de ebir ekran görüntüsü verilen FKaf_Consumer, aynı zamanda verilerin bir dosya yazılmasına da olanak sağlamaktadır. Sensör veriler Kafka içerisinde birden fazla "partition" a yazılmaktadır. Bu özellikle her hangi bir veri kaybını önlemek ve istemci bazında ölçekleme yapabilmek için yapılmıştır. Alınan sensör verilerinin istemcilere haber verilmesi, Kafka içerisinde oluşabilecek bir broker hatasının bildirilmesi gibi koordinasyon işlemleri ZooKeeper tarafından yapılmaktadır. Tüm Kafka servisleri bilgi paylaşımlarını ZooKeeper cluster 'ları üzerinden yapmaktadır. Üretilen sensör veriler ile ilgili metadatalar, broker ve kuyruk okumaları gibi temel veriler ZooKeeper üzerinde olacaktır. Her ne kadar dağıtık kurulumda zorunlu olmasa da ZooKeeper aynı zamanda HBase yürütmelerinde de koordinasyon görevi sağlayabilmektedir. Mesage Broker kullanımı sadece belli bir sorgu üzerine çalışmayan ve klasik ATBS çalışma için kullanılmaktadır.

Buradaki tek fark FVWSN temelli etkileşimli yürütümlerde istemci VNd ‘lerinin üretmiş oldukları verileri bir topic olarak yayımlayabilmeleridir. Bu bir istemcinin ön tanımlı VNd ‘lerinin yürütümlerinin bildirilmesi için kullanılabileceği gibi aynı zamanda farklı istemcilerin taleplerini karşılamak içinde kullanılabilir. Örneğin bir bölgedeki sensör ölçüm sonuçlarını belli bir bilimsel yöntemle inceleyen bir VNd bunu bir topic olarak mesagge broker üzerinden paylaşabilir.

Şekil 6.5. FKaf_Consumer yazılımı

Üretilen her verinin sistem üzerinde kaydedilmesi için iki seçenek mevcuttur. Bunlardan birincisi doğrudan text dosyalarının HDFS üzerinde tutulması ikincisi ise HBase üzerinde depolanmasıdır. HBase üzerindeki SQL benzeri sorgulamalar Hive yardımı ile gerçekleştirilebilmektedir. Bu seçenek uygulama tipine ve bulut yürütümüne bağlıdır. MapReduce servisleri statik verilerin analizinde kullanılacaktır. Her ne kadar Spark gibi daha hızlı çözümler bu sisteme eklenebilse de bu modelde analiz işlemlerinin statik olarak yapıldığı dikkate alınmıştır.

Bir diğer önemli konu ise üretilen verilerin benzer veya tekrarlı olmasından kaynaklanan verimsiz veri depolamadır. Bilindiği üzere klasik WSN sistemlerde gereksiz (redundant) veriler iki aşamalı olarak elenmektedirler. Bunlardan ilki düğüm üzerinde aynı veri setindeki benzer verilerin silinmesi, ikincisi ise aggregator düğümlerde farklı veri

setleri arasındaki benzerliklerin elimine edilmesidir. Ancak tahmin edileceği üzere, önerilen IoT/CPS WSN sağlayıcı sistemdeki bütün alt WSN sistemlerin hepsinde bu tip bir "data aggregation" işevi bulunması beklenemez. Dahası aynı bölge için aynı verileri gönderen istemciler birbirlerinden haberdar olmayabilirler. Örnek olarak Elazığ ili merkezi için sıcaklık, nem gibi verileri için farklı istemciler sisteme gönderebilir. Bu şekilde belli bir değişim aralığı içinde gelen aynı tip verilerin birleştirilerek sistemde tutulması avantaj sağlayacaktır. Aynı durum farklı VNd ler için üretilen ve paylaşılabilen veriler içinde geçerli olacaktır. Bu sebeple önerilen sistemde bir veri benzerlik denetleyicisinin olması yararlı olacaktır. Burada dikkat edilmesi gereken üretilen verilerin özgünlüğüdür. Üretilen verilerin özgünlüğü iki alana bağlıdır. Bunlardan birincisi VNd sahibinin ayırt edici kimlik verisi, ikincisi ise aynı veri kaynağı için veri üretim zaman bilgisidir. Bu ayırt edici veriler dışında aynı konum ve olay için belli bir benzerlik eşik değerinin üzerinde olan veri setleri depolanmayacaktır. Buradaki ayrım tamamen sisteme özel olup benzer uygulama istemci tanımlı olarak bir VNd içerisinde de gerçekleştirilebilir. Önemli olan parametre benzerlik eşik değeri olup, senaryo tipine göre değişiklik gösterebilmektedir. Her ne kadar literatürde bulunan benzerlik fonksiyonları uygulama alanı ve veri tiplerine göre farklılık gösterse de sadece basit vektör verilerin analizinde yaklaşık sonuç vermektedirler [128]. Bu temelde önerilen sistemde Tablo 6.2 'de verilen sık kullanılan benzerlik fonksiyonlarından "Jaccard Similarity Function –JSF" tercih edilmiştir.

Tablo 6.2. Sık kullanılan benzerlik fonksiyonları

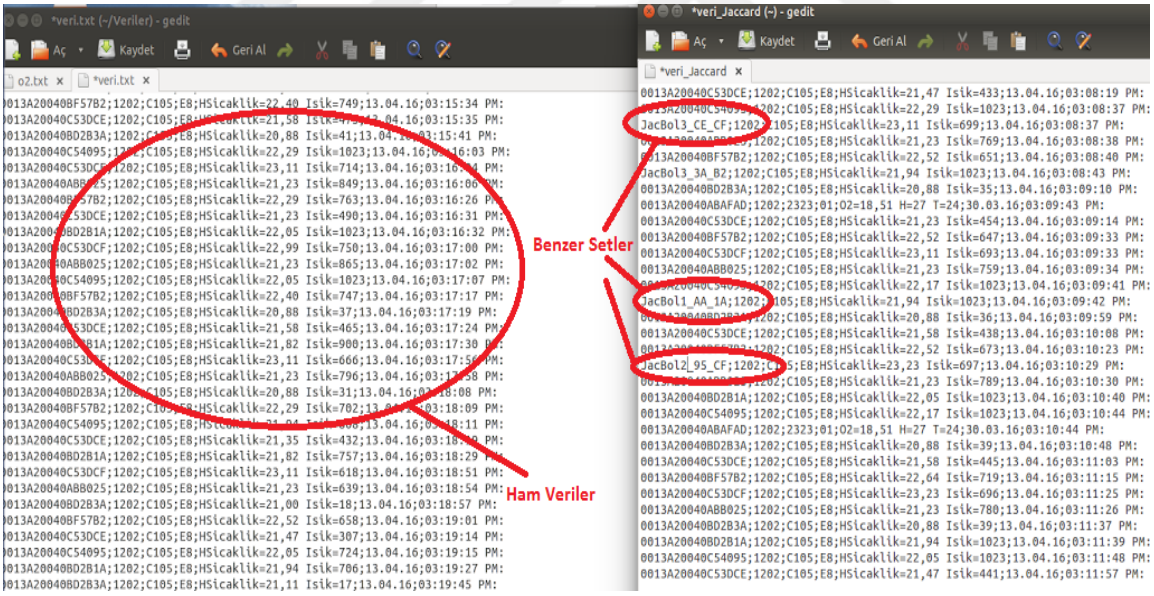
Fonksiyon Adı	**Fonksiyon İfadesi**
Cosine	$C(S_1, S_2) = \frac{\lvert S_1 \cap S_2 \rvert}{\sqrt{\lvert S_1 \rvert + \lvert S_2 \rvert}}$
Dice	$D(S_1, S_2) = \frac{2x\lvert S_1 \cap S_2 \rvert}{\lvert S_1 \rvert + \lvert S_2 \rvert}$
Overlap	$O(S_1, S_2) = \lvert S_1 \cap S_2 \rvert$
Jaccard	$J(S_1, S_2) = \frac{\lvert S_1 \cap S_2 \rvert}{\lvert S_1 \cup S_2 \rvert}$

Bölüm WSN laboratuvarında daha önceden toplanmış olan yaklaşık 800.000 ışık, sıcaklık,nem ve O_2 ölçümleri üzerinde oluşturulan 4 ortak bölge için bir dk içerisindeki farklı düğümlerden gelen aynı tip fiziksel ölçüm verileri üzerinde 0.50, 0.75 ve 1 benzerlik oranları için veri küçültme işlemi yapılmıştır. Ortak bölgelerdeki aynı tip sensörler için benzer okuma durumunda kullanacakları bir ortak ID atanmıştır. Bu durumda bir ortak

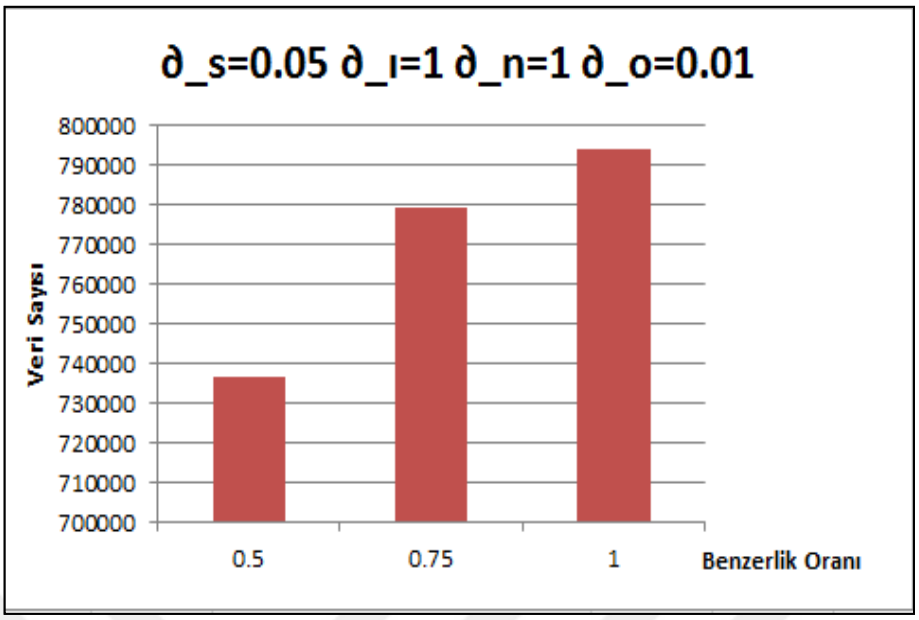
bölge için yapılan sorgulamada ortak ID verisi, aynı bölgedeki sensör kaynaklar için paylaşılabilir olmuştur. Bir veri vektöründe her bir alan öncelikle kendi özdeş alanı ile eşlik karşılaştırılmasına tabi tutulmuştur. Bu karşılaştırma da iki veri arsındaki farkın belirlenen eşik değerden (∂) küçük olup olmadığı kontrol edilmektedir. Bu eşik değer ölçüm tipine göre değişiklik gösteren ön tanımlı bir değerdir. Örneğin ∂_s sıcaklık, $\partial_ı$ ışık, ∂_n nem, ∂_o oksijen eşik değer tiplerini göstermektedir. Genel ifadesi Denklem 6.1 deki şekildedir.

$$EslikVar(v_i, v_j) = \begin{cases} 1 & if\ ||v_i - v_j|| < \partial \\ 0 & Diğer\ durumlar \end{cases} \tag{6.1}$$

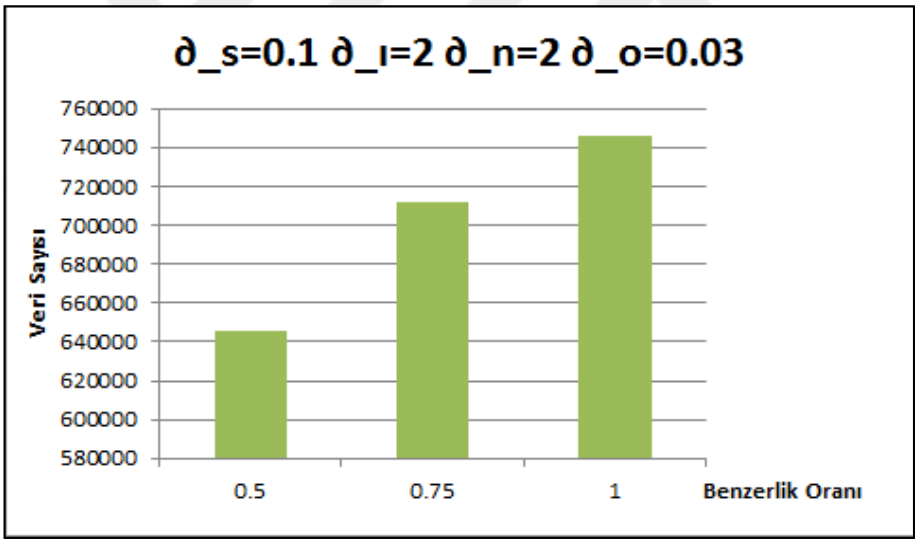
Şekil 6.6 bu işlemden öncesi ve sonrası için elde edilen bir karşılaştırma ekran görüntüsünü vermektedir. Şekil 6.7 a-c ise farklı eşik değerler için veri küçültme işlemine ait genel sonuçları göstermektedir.



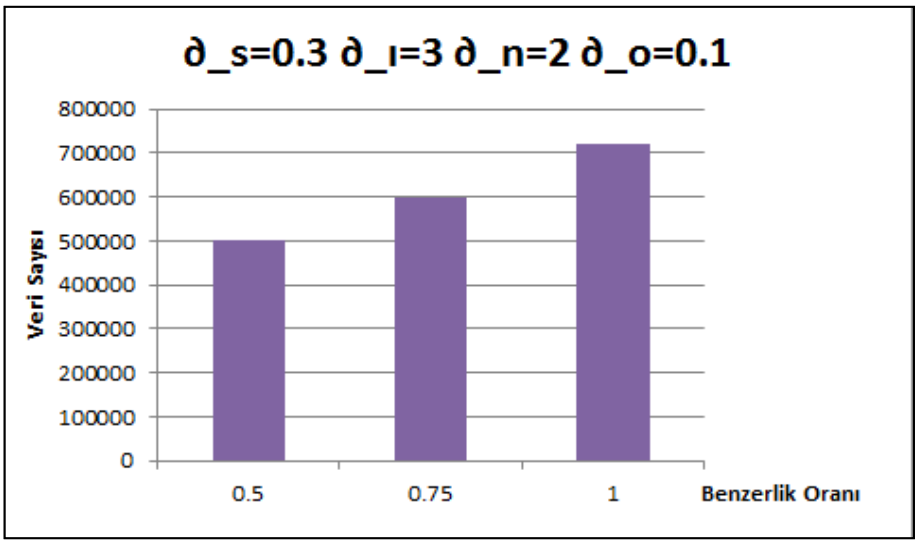
Şekil 6.6. Ham veri setlerinin JSF ile işlenmesi ve küçültülmesi



a) Sıcaklık eşik=0.05, Işık eşik=1, Nem eşik=1, Oksijen eşik=0.01



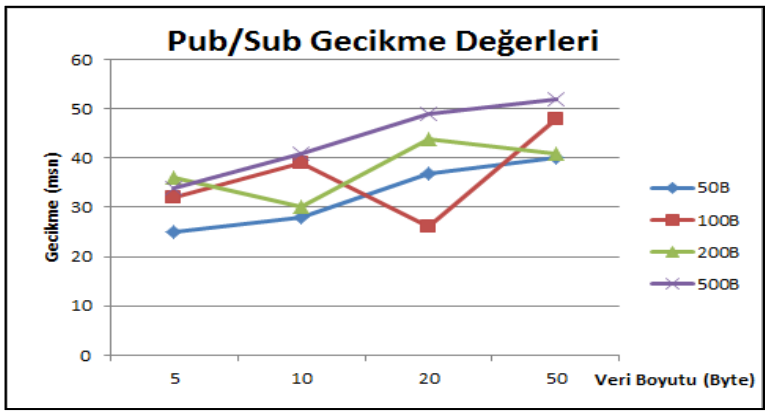
b) Sıcaklık eşik=0.1, Işık eşik=2, Nem eşik=2, Oksijen eşik=0.03



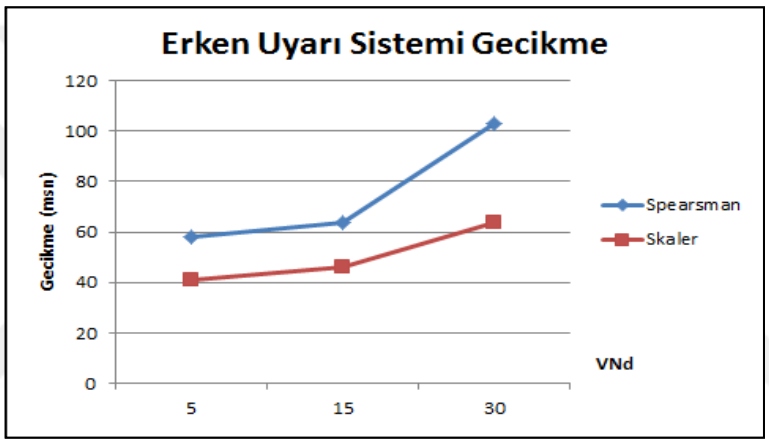
c) Sıcaklık eşik=0.3, Işık eşik=3, Nem eşik=2, Oksijen eşik=0.1

Şekil 6.7. Farklı eşik değerler için JSF ile veri küçültme deneyleri

Bunun dışında gözlemlenen bir diğer sistemin performans kriteri de Pub/Sub ara katman yazılımının gecikmesidir. Pub/Sub arakatman yazılımı gerek doğrudan veri gönderimi gerek de zamanlanmış VNd düğümlerin çalışma sonuçlarının bildirimlerinde kullanıldığından sistemin önemli bir parçasıdır. Pub/Sub arakatman olarak kullanılan Kafka teknolojisi, sağladığı replikasyon özelliği sayesinde gelen verileri belli sayıda broker üzerine dağıtmaktadır. Oluşturulan topic'lerin oluşturulmasında bu replikasyon değeri belirlenmekte ve ilgili veri kanalların ayarlanması Kafka tarafından bu broker 'lara otamatik olarak dağıtılmaktadır. Veri tüketiciler (Consumer) her ne kadar belirli bir socket üzerinden bağlantı kursalar da Kafka, ZooKeeper ile haberleşerek uygun broker 'dan verinin çekilmesini sağlayabilmektedir. Bir broker kanalından ayrı gruplar için veri dağılımı yüksek hızda sağlanabilmektedir. Kafka testleri farklı mesaj boyutları ve farklı mesaj üretim oranları için denenmiştir. Bunun için 50 üretici kaynak , 50-100-200-500 byte 'lık skaler verileri saniyede 5-10-20-50 veri üretim oranı ile göndermiştir. Bu testler sonucunda, he ne kadar bazı değerler için bir sonraki değere göre farklı sayılabilecek sonuçlar elde edilse de genel olarak gecikme değerinin çok yüksek olmadığı görülmüştür. Daha sonra erken uyarı sisteminin gecikme testi gerçekleştirilmiştir. Bu testte ise her biri sırasıyla "Spearsman Rank Corraletion" ve basit bir skaler kontrol algoritması çalıştıran 5, 15 ve 30 VNd için elde edilen gecikme değerleri gözlemlenmiştir. Her iki teste ait sonuçlar Şekil 6.8 a ve b 'de verilmiştir.



a) Pub/Sub gecikmesi

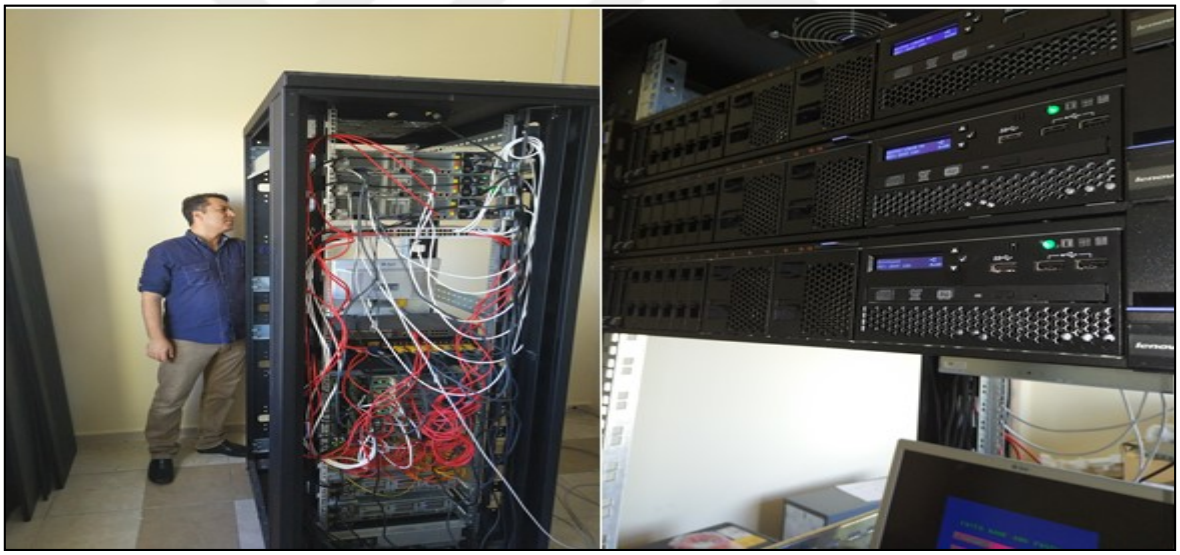


b) Erken uyarı modülü gecikmesi

Şekil 6.8. Farklı veri boyutlu skaler veriler için Pub/Sub gecikme değerleri

Bulut sistemin ulaşımı Şekil 6.9 daki bir web arayüzü aracılığı ile sağlanmaktadır. Bu web arayüzü sayesinde istemciler ve sağlayıcılar gerekli yönetimsel işlevleri gerçekleştirebilmektedir. FVWSN istemciler bu web sayfası üzerinden ayrı bir yönetim paneline geçmektedirler. Burada FVWSN framework 'ünü indirebilir, oluşturdukları VNd 'yi sisteme yükleyebilir veya kendi VNd 'lerinin durumlarını gözlemleyebilir. Bunun dışında kısıtlı yetkilere sahip, FVWSN istemcileri ise sadece kendi komut/sorgularını tanıtabilecekleri ve basit algoritmalarını yazabilecekleri ayrı bir arayüzü de bu sayfada bulunmaktadır. Son olarak önerilen IoT-WSN sistemin donanımlarının kurulu olduğu fiziksel ortama ait bir görüntü ise Şekil 6.10 'da verilmiştir.

Şekil 6.9. Sistem Web Arayüzleri



Şekil 6.10. Sistem odasından bir görüntü

7. SONUÇ VE ÖNERİLER

7.1. Sonuçlar

Günümüz teknolojileri artık IoT ve CPS ‘ler üzerinden birbirleri ile entegre olmakta ve daha karmaşık problemlerin çözümlerinde önemli görevler alabilmektedir. Ağ ve internet teknolojilerinin gelişmesi ile daha kolay bir hale gelen bu entegrasyon, heterojen otonom yapıların aynı çatı altında toplanmasına yol açmaktadır. Bugün özellikle akıllı şehir (smart city), akıllı şebekeler (smart grids) gibi büyük ölçekli IoT sistemlerde bu heterojen yapılarla sıklıkla karşılaşılmaktadır. Ölçme ve algılama pek çok IoT/CPS için olmazsa olmazlar arasında olduğundan, hiç kuşkusuz, bu heterojen yapıların en önemli bileşenlerinden birini WSN sistemler teşkil etmektedir. Mazisi birkaç on yıl geriye dayanan WSN sistemler, başlangıçta genellikle amaca yönelik kapalı sistemler olarak çalışmaktaydılar. WSN ‘lerin gelişme ve olgunlaşma döneminde yaşadığı en önemli problemlerden birisi standartlaşma olmuştur. Bu süre içerisinde birçok üniversite, araştırma merkezi ve ticari firma kendi amaçları doğrultusunda farklı WSN teknolojileri geliştirdi. Bu ise zaman içerisinde doğal olarak beraber çalışabilirlik problemlerinin ortaya çıkmasına yol açtı. Daha sonraları 802.15.4 protokolü ve bu protokol üzerine kurulu ZigBee, ZWave gibi bazı standartlaşma projeleri önemli yol kat ettilerse de tam bir çözüm sağlanamadı. Aktuatör kullanan, farklı güvenlik gereksinimleri olan yapılar ortaya çıktıkça sunulan çözümler belli sınırlar içerisinde kaldı. Daha önceleri gateway ve API yardımları ile sayılı WSN sistem arasında etkileşim sağlanabilmekte iken dinamik olarak değişen ölçeklenebilir sistemlerin ortaya çıkması ile bu durum yetersiz kalmaya başlamıştır. Daha önceleri tek bir WSN ‘deki heterojen yapılar için önerilen çözümler, doğal olarak, IoT/CPS içerisinde gözle görülür bir başarı elde edememiştir. IoT/CPS sistemlerin temel özelliklerinden olan kaynak paylaşımının önündeki en önemli problemlerden olan heterojenliğin çözümünde en çok kullanılan yöntem sanallaştırma olmuştur. Ancak konu kaynak kısıtlarına sahip WSN sistemler olunca sanallaştırma kavramı daha karmaşık bir hal almaktadır. Literatürde önerilen WSN sanallaştırma tekniklerinin, amaca yönelik ve genellikle homojen tip ağlara yönelik olduğu görülmektedir. Özellikle zaman-kritik etkileşimli heterojen uygulamaları içeren IoT/CPS projelerde bu yöntemler yetersiz

kalabilmektedir. Öte yandan çok sayıda sensör düğüm içeren büyük IoT/CPS sistemlerden elde edilen veriler devasa boyutlara ulaşabilmektedir. Bu verilerin depolanması, işlenmesi ve diğer istemcilerle paylaşılması dikkate alınması gereken bir başka önemli problemdir. Özetle IoT/CPS ölçekli projelerde en önemli problemlerden ikisi, kısıtlı kaynaklı heterojen WSN 'ler arasında kaynak paylaşımının nasıl yapılabileceği ve oluşabilecek büyük verilerin nasıl işlenebileceğidir.

Bu tez çalışmasında, farklı heterojen WSN ağların dinamik olarak birbirleri ile kolay bir şekilde entegre olup, kaynak paylaşımları yapılabilmesine olanak sağlayan, oluşabilecek büyük verilerin depolanmasına ve işletilebilmesine imkan verebilen bir IoT/CPS WSN sistemin mimarisi ve bu mimarinin gerçekleştirilmesi yapılmıştır. Bunun için öncelikle Bölüm 2 'de açıklandığı gibi IoT/CPS 'lerde WSN heterojenliği incelenmiş ve bir sınıflandırma modeli sunulmuştur. Bu sınıflandırmaya göre IoT/CPS ölçekli projelerde temel olarak iki tip heterojenlik olabileceği ifade edilmiştir. Bunlar düğüm bazlı heterojenlik ve ağ bazlı heterojenliktir. Düğüm bazlı heterojenlikte katman bazlı, donanım bazlı, sensör/aktuatör bazlı ve yönetim bazlı farklılıklar ayrı ayrı açıklanmıştır. Ağ bazlı heterojenlikte ise oluşabilecek topoloji, lokasyon ve çalışma modeli farklıkları izah edilmiştir. Oluşabilecek bu heterojen yapılar arasında kaynak paylaşımı için kullanılacak WSN sanallaştırma tekniklerinin teknik detayları yine bu bölümde verilmiştir. Bu sanallaştırma teknikleri, WSN düğümler üzerindeki yazılım teknolojileri, geliştirilen P2P protokoller veya bulut teknolojileri vasıtasıyla gerçekleştirilmiştir. Bir WSN düğümün hafıza ve işlem kapasitesindeki kısıtlar dikkate alındığında çok sayıda istemci uygulaması gerektirebilen IoT/CPS projelerde düğüm üzerindeki sanallaştırma tekniklerinin yetersiz kalacağı ve bu tekniklerin diğer olumsuz yönleri bu bölümde detaylandırılmıştır. Ancak bununla beraber zaman-kritik uygulamalarda en etkin çözümü bu tekniklerin sağladığı bir gerçektir. Özel P2P protokollerin kullanıldığı ağ bazlı sanallaştırma tekniklerinde ise en önemli problem, tüm WSN teknolojilerinin bu protokolleri destekleyememeleridir. Bu nedenlerden dolayı IoT/CPS ölçeğinde kullanılabilecek en iyi kaynak paylaşım tekniğinin bulut teknolojili WSN sanallaştırma sistemleridir. Ancak bulut sistem üzerinden yapılan kaynak paylaşımı ise genellikle veri yönelimli ve tek yönlü etkileşimi sağlamaktadır. Farklı ağlardaki WSN düğümler arasındaki etkileşim ve kod anlaşılabilirliği bu sanallaştırma yönteminde göz ardı edilmiştir. Öte yandan zaman-kritik uygulamalar için çok verimli değildirler. Kısaca, IoT/CPS projelerde dinamik olarak sisteme katılabilen/ayrılabilen WSN sistemler arasında kod/algoritma taşınabilirliğinin sağlanabilmesi ve özellikle

zaman-kritik uygulamalarda hızlı etkileşime olanak veren bir IoT/CPS WSN sağlayıcı sistem dizaynına ihtiyaç olduğu görülmektedir. Bu ihtiyaç temelinde bu tez çalışmasında istemci ve sağlayıcı arasında çalışabilen, istemci tanımlı kod/algoritmaları ön ayarsız bir şekilde yürütücü (sağlayıcı) sisteme taşıyabilen ve arka plan karmaşıklığını en iyi şekilde gizleyen bir yazılımsal framework geliştirilmiştir. FVWSN olarak adlandırılan bu yazılımsal framework 'ün teknik detayları ve bunun yürütüldüğü sistem dizaynı Bölüm 3 'de verilmiştir. Bu framework sayesinde istemciler tarafından oluşturulan mantıksal sanal düğüm bileşenleri (VNd) ile istemciye ait komut/sorgu ve algoritmalar yine istemci tarafından kolaylıkla tanımlanabilmektedir. İstemciler bunu yaparken, sağlayıcı sisteme ait teknik detay bilgisine ihtiyaç duymadan sistem fonksiyonlarını çağırabilmekte ve bunları sağlayıcı sistemde ön ayarsız bir şekilde aktif hale getirebilmektedir. Bu framework ve yürütücü sistem hem gateway 'ler üzerinde hem de bulut sistemler üzerinde çalıştırılabilmektedir. Ancak önerilen modelin IoT/CPS projelere yönelik olmasından dolayı bu yazılım, bir bulut sisteme entegre edilmiştir. Bulut sistem olarak OpenStack teknolojisi kullanılmıştır. Öte yandan önerilen sistem mimarisi büyük ölçekli IoT/CPS sistemlerde oluşabilecek büyük veri problemini de dikkate almaktadır. Bu problemin çözümü için ise günümüz büyük veri teknolojilerinden olan Hadoop ekosistemden faydalanılmıştır. Aynı zamanda veri yönelimli taleplere hizmet verilebilmesine olanak sağlayan Pub/Sub arakatman teknolojisi de sistem mimarisinde yer almaktadır. FVWSN framework 'ünün sağladığı bir diğer avantaj ise, oluşturulan VNd 'ler ile bu Pub/Sub mimarinin beraber çalışabiliyor olmasıdır. Bu özellikle erken uyarı/alarm uygulamaları için oldukça elverişlidir. Bu erken uyarı algoritmalarının istemci tanımlı olması önerilen sistemin bir başka özgün yanıdır. Bir istemci VNd 'si belirlediği topic 'lere abone olabilir ve topic 'lerden gelen veri akışını inceleyen erken uyarı algoritmalarını yürüterek acil durumlarda alarm topic 'lerine uyarı verilerini gönderebilir.

Sonuç olarak bu tez çalışmasının literatüre katkıları aşağıdaki şekilde özetlenebilir;

- Önerilen sistem, istemcilere, sunucu tarafında seçtikleri kaynakları, düğüm bazlı sanallaştırmaya ihtiyaç kalmadan, kendi komut, sorgu veya algoritmaları ile yönetebilmesine imkan verebilmektedir. Bu özellikle zaman-kritik uygulamalar için daha düşük bir gecikme sağlayabilmektedir.
- Smart city, smart grid gibi IoT/CPS projelerde, sisteme dinamik olarak eklenen WSN 'lere ait düğümler, daha önce hiç bilmedikleri başka WSN sistemlerdeki düğümlerle kolaylıkla M2M olarak haberleşebilmektedir.

- Değerlendirme ve kontrol süreçlerinin sağlayıcı tarafta işletilmesinden dolayı sağlayıcı-istemci arası trafik minimize edilebilmektedir. Bu ise istemciye daha ekonomik bir çözüm sunarken, sağlayıcı taraf için ise daha az trafik yoğunluğu sağlamaktadır.
- FVWSN framework ‘ü sayesinde sağlayıcı taraftaki tüm teknik karmaşıklık istemciden gizlenebilmekte ve istemci için maksimum kodlama basitliği sağlanabilmektedir. FVWSN yardımı ile oluşturulan VNd ‘ler sağlayıcı sistemde herhangi bir ön ayara ihtiyaç duymadan aktif hale getirilebilmektedir.
- Önerilen IoT/CPS WSN bulut sistem içerdiği pub/sub ara katman yazılımı ile aynı zamanda veri yönelimli bir hizmet de sunabilmektedir. Buna ek olarak istemci VNd ‘leri belirli topic ‘lere abone olup erken uyarı/alarm algoritmalarını online olarak işleyebilir ve bunları alarm topic ‘lerinden yayımlayabilir.
- Önerilen sistem büyük veri işleyebilen yapısı ile büyük ölçekli IoT/CPS projeler için esnek ve ekonomik bir alt yapı desteği sağlayabilmektedir. Sistemin burada sağladığı ek hizmet ise toplanan verilerin benzerlik fonksiyonlarından geçirilerek gereksiz veri kayıtlarını önleyebilmesidir.
- Önerilen sistem ölçeklenebilir ve adaptif bir yapıya sahip olduğundan günümüz internet teknolojileri ile güvenli bir şekilde entegre olabilmektedir.

Tezde önerilen sistemin, hem temel gatewayler üzerine hem de farklı bulut teknolojileri üzerine kurulabilir olması, özellikle akıllı şehir gibi büyük ölçekli IoT/CPS projelerinde tercih edilebilirliğini arttıracaktır. Yukarda özellikleri sıralanan tez çalışmasında önerilen sistemin, IoT/CPS projelerinde önemli bir boşluğu dolduracağı öngörülmektedir.

7.2. Sistemin Geliştirilmeye Açık Yönleri Gelecek Çalışmalar

İstemci VNd ‘lerinin sunucu sisteme yüklenerek çalıştırılması doğal olarak güvenlik problemlerini de beraberinde getirmektedir. Bu nedenle önerilen sistem, güvenlik açısından iyileştirmelere ihtiyaç duymaktadır. Her ne kadar bu sistemde tüm istemciler VNd oluşturma yetkisine sahip olmasalar da ve oluşturulan VNd ‘ler Java Security/Policy kontrolünden geçirilseler de güvenlik açısından farklı geliştirilmelere gerek duyulmaktadır. Öte yandan istemci taleplerine göre yük dağılımı, VNd replikasyonlarının ikinci bir kontrolörde tutulması ve sistemin sürekli gözlemlenmesi sistem yürütümü açısından önem arz etmektedir.

Bundan sonraki çalışmalarda bu IoT/CPS WSN sağlayıcı mimarisi altyapısını kullanan akıllı şehir, akıllı kampus gibi projelerin gerçekleştirilmeleri amaçlanmaktadır. Özellikle toplanan verilerin bilimsel yönetimlerle izlenebilmesini, diğer disiplinlerle paylaşılabilmesini sağlayacak bölgesel ve ülke çapında bir bulut hizmetinin sağlanabilmesini sağlayacak projeler önerilecektir. Bunlara ek olarak gereksiz verilerin elenerek optimize büyük verilerin oluşturulması ve bu oluşturulan verilerin anlamsallığını artırarak daha paylaşılabilir veri setleri elde edilmesi için gerekli çalışmalar bu temelde yapılacaktır.

KAYNAKLAR

[1] **Akyildiz, I.F. and Vuran, M.C.,** 2011. Wireless Sensor Networks, Willey & Sons, West Sussex.

[2] **Emary, M.M.I., and Ramakrishnan, S.,** 2016. Wireless Sensor Networks: From Theory to Applications, CRC Press, New York.

[3] **Papavassiliou, S., Kato, N., Liu, Y., Xu, C.Z., and Wang, X.**, 2012. Cyber-Physical Systems (CPS), IEEE Transaction on Parallel and Distributed Systems, Vol:23, No: 9.

[4] **Fuqaha, A.A., Guizani, M., Mohammadi, M., Aledhari, M., and Ayyash, M.**, 2015. Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications, IEEE Communications Surveys & Tutorials, Vol:17, No: 4, doi: 10.1109/COMST.2015.2444095.

[5] **He, T., Krishnamurthy, S., Stankovic, J. A., Abdelzaher, T., Luo, L., Stoleru, R., Yan, T., Gu, L., Zhou, G., Hui, J., and Krogh, B.,** 2006. VigilNet: an integrated sensor network system for energy-efficient surveillance. ACM Transactions on Sensor Networks, 2(1):1–38.

[6] **Basha, E. A., Ravela, S., and Rus, D.,** 2008. Model-based monitoring for early warning flood detection. In Proceedings of ACM SenSys'08, Raleigh, NC, USA, pp. 295–308.

[7] **Gao, T., Pesto, C., Selavo, L., Chen, Y., Ko, J. G., Lim, J.H., Terzis, A., Watt, A., Jeng, J., Chen, B.R., Lorincz, K., and Welsh, M.,** 2008. Wireless medical sensor networks in emergency response: implementation and pilot results. In Proceedings of IEEE International Conference on Technologies for Homeland Security, Waltham, MA, USA, pp. 187–192.

[8] **Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., and Cayirci, E.,** 2002. Wireless sensor networks: a survey. Computer Networks, 38(4):393–422.

[9] **Gharaibeh, A., Salahuddin, M.A., Hussini, S.J., Khreishah A., and Khalil, I.**, 2017. Smart Cities: A Survey on Data Management, Security and Enabling Technologies, IEEE Communications Surveys & Tutorials, Vol:PP, No: 99, doi: 10.1109/COMST.2017.2736886.

[10] **Sidra, I., Munam, A. S., Abid, K., and Mansoor, A.,** 2016. Smart Cities: A Survey on Security Concerns, International Journal of Advanced Computer Science and Applications, Vol. 7, No. 2.

[11] **Arockia, C., and Sudarsanan, K**., 2016. A Survey on Smart Grid, Emerging Trends in Engineering, Technology and Science (ICETETS), 24-26 February, Pudukkottai, India, doi: 10.1109/ICETETS.2016.7603069.

[12] **Kelvin, L., and Yier, J.**, 2016. Security Challenges in CPS and IoT: From End-Node to the System, IEEE Computer Society Annual Symposium on VLSI (ISVLSI), Pittsburg, USA, 11-13 July, doi: 10.1109/ISVLSI.2016.109.

[13] **Samad, T.,** 2016. Control Systems and the Internet of Things, IEEE Control Systems, Vol.36, No:1, pp:13-16.

[14] **Yıldırım, G., Tatar, Y., ve Aydın, G.,** 2014. Hadoop ile Kaos Temelli FCW Optimizasyon Algoritmasının Analizi An Analysis of Chaos-Based the FCW Optimization Algorithm by Hadoop, Eleco14.

[15] **Khan, I., Belqasmi, F., Glitho, R., Crespi, N., Morrow, M., Polakos P**., 2015. Wireless sensor network virtualization: A survey, IEEE Communications Surveys & Tutorials Vol: 18, No: 1, doi: 10.1109/COMST.2015.2412971.

[16] **Levis, P., and Culler, D.,** 2002. Mate: A Tiny Virtual Machine for Sensor Networks. In Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems, San Jose, CA, USA, October 5–9.

[17] **Kabadayi, S., Pridgen, A., and Julien, C.,** 2006. Virtual Sensors: Abstracting Data from Physical Sensors. In Proceedings of the International Symposium on a World of Wireless, Mobile and Multimedia Networks, Buffalo-Niagara Falls, NY, USA, June 26–29.

[18] **Shin, J.H., and Park, D.,** 2007. A virtual infrastructure for large-scale wireless sensor networks. J. Comput. Commun. 30, 2853-2866.

[19] **Polastre, J., Hui, J., Levis, P., Zhao, J., Culler, D., Shenker, S., and Stoica**, **I.**, 2005. A Unifying Link Abstraction for Wireless Sensor Networks. In Proceedings of the 3rd International Conference on Embedded Networked Sensor, San Diego, CA, USA, November 2–4.

[20] **Waharte, S., Xiao, J., and Boutaba, R.,** 2004. Overlay Wireless Sensor Networks for Application-Adaptive Scheduling in WLAN. In Proceedings of the 7th

IEEE International Conference on High Speed Networks and Multimedia Communications (HSNMC), Toulouse, France, June 30– July 2, LNCS 3079, pp. 676-684.

[21] **Yu, Y., Rittle, L.J., Bhandari, V., and LeBrun, J.B.,** 2006. Supporting Concurrent Applications in Wireless Sensor Networks. In Proceedings of the 4th International Conference on Embedded Networked Sensor Systems, Boulder, CO, USA, November 1–3.

[22] **Houben, H.,** 2006. Virtual Sensor Technology for Mars Exploration. In Proceedings of 2006 Second Workshop on Mars Atmosphere Modelling and Observations, Granada, Spain, February 27– March 3.

[23] **Efstratiou, C., Leontiadis, I., Mascolo, C., and Crowcroft, J.,** 2010. Demo Abstract: A Shared Sensor Network Infrastructure. In Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems (Sensys'10), Zurich, Switzerland, November 3–5.

[24] **Bhattacharya, A., Fernando, M.S., and Dasgupta, P.,** 2008. Community Sensor Grids: Virtualization for Sharing across Domains. In Proceedings of the First Workshop on Virtualization in Mobile Computing, Breckenridge, CO, USA, June 17–20.

[25] **Hassan, M.M., Song, B., and Huh, E.N.,** 2009. A Framework of Sensor—Cloud Integration Opportunities and Challenges. In Proceedings of the 3rd International Conference on Ubiquitous Information Management and Communication, Suwon, Korea, January 15–16.

[26] **Nuno C., Antonio P., and Carlos S.,** 2007. Virtual Machines Applied to WSN's: The state-of-the-art and classification, Second International Conference on Systems and Networks Communications (ICSNC 2007),25-31 Aug., Esterel, France, doi: 10.1109/ICSNC.2007.83.

[27] **Rifat, J., Imran, K., Jagruti, and S.,Roch G.,** 2016. A framework for ontology provisioning in Virtualized Wireless Sensor Networks, IEEE Symposium on Computers and Communication (ISCC), June 27-30, Messina, Italy, doi: 10.1109/ISCC.2016.7543825.

[28] **Ali, M., and Koen L.,** 2007. A case for peer-to-peer network overlays in sensor networks, International Workshop on Wireless Sensor Network Architecture (WWSNA'07), 2007, pp.56-61.

[29] **Fersi G., Louati W., and Jemaa M. B.,** 2013. Distributed Hash table-based routing and data management in wireless sensor networks: a survey, Wireless Netw, vol. 19, no. 2, pp. 219–236.

[30] **Luu, H. V., and Xueyan T.,** 2013. Constructing rings overlay for robust data collection in wireless sensor networks, Journal of Network and Computer Applications Vol:36, No:5 : pp:1372-1386.

[31] **Antonio J. J., Dominique G., and Yann B.,** 2014. Big Data for Cyber Physical Systems: An Analysis of Challenges, Solutions and Opportunities, Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), July 2-4, Birmingham, doi: 10.1109/IMIS.2014.139.

[32] **Yıldırım G., ve Tatar Y.,** 2017. On WSN Heterogeneity in IoT and CPSs, Uluslararası Bilgisayar Bilimleri ve Mühendisliği Konferansı, 5-8 Ekim, Antalya.

[33] www.riverbed.com/, En son ziyaret 14.04.2017

[34] **Jaroodi J., and Aziz J., Mohamed N.,** 2009. Middleware for RFID systems: An overview, 33. IEEE International Conference (COMPSAC'09), vol. 2, pp. 154–159.

[35] **Karpischek, S., Michahelles, F., Resatsch, F., and Fleisch, E**., 2009. Mobile sales assistant an NFC-based product information System for retailers, Proceedings of the First International Workshop on Near Field Communications, Hagenberg, Austria.

[36] **Issarny, V., Georgantas, N., Hachem, S., Zarras, A., Vassiliadist, P., Autili, M., Gerosa, M.A., and Hamida, A.B.,** 2011. Service-oriented middleware for the future internet: state of the art and research directions, J. Internet Service Applications Vol:2 No:1, pp: 23–45.

[37] **Botterman, M.,** 2009. Internet of Things: An Early Reality of the Future Internet, Report of the Internet of Things Workshop, Prague, Czech Republic.

[38] http://www.ipso-alliance.org/, En son ziyaret 10.10.2017

[39] **Duquennoy, S., Grimaud, G., and Vandewalle, J. J.,** 2009. The web of things: interconnecting devices with high usability and performance, In: Proceedings of ICESS '09, Zhejiang, China, s.551-556.

[40] **Mohammad A.R., Marija M., Andrei P., and Siobhán C.,** 2016. Middleware for Internet of Things: A Survey, IEEE Internet Of Things Journal, Vol. 3, No. 1.

[41] **Gazis, V**., 2016. A Survey of Standards for Machine-to-Machine and the Internet of Things, IEEE Communications Surveys & Tutorials, Vol: 19, No: 1, doi: 10.1109/COMST.2016.2592948.

[42] **Hui J., and Thubert, P.,** 2011. Compression format for IPv6 datagrams over IEEE 802.15.4-based networks, RFC 6282, Internet Eng. Task Force, Fremont, CA, USA.

[43] **Thubert P.,** 2012. Objective function zero for the routing protocol for low-power and lossy networks (RPL), RFC 6552, Internet Eng. Task Force, Fremont, CA, USA,

[44] **Shelby Z., Hartke K., and Bormann C.,** 2014. The constrained application protocol (CoAP), RFC 7252, Internet Eng. Task Force, Fremont, CA, USA.

[45] https://xmpp.org/extensions/xep-0322.html, En son ziyaret, 10.10.2017

[46] https://www.w3.org/TR/soap/ , En son ziyaret, 10.10.2017

[47] http://www.restapitutorial.com/, En son ziyaret, 10.10.2017

[48] **Yang, G., Li, X., Ma¨ntysalo, M., Zhou, X., Pang, Z., Xu, L.D., Walter, S.K., Chen, Q., and Zheng, L.,** 2014. A health-IoT platform based on the integration of intelligent packaging, unobtrusive biosensor and intelligent medicine box, IEEE Trans. Ind. Inf., Vol:10, No:4, s.2180–2191.

[49] **Zhang, J., and Anwen, Q.,** 2010. The application of internet of things (IOT) in emergency management system in China, In: Proceedings of IEEE International Conference on Technologies for Homeland Security (HST), Los Angeles, pp. 139–142.

[50] **Wang, S., Zhang, Z., Ye, Z., Wang, X., Lin, X., and Chen, S.,** 2013. Application of environmental Internet of Things on water quality management of urban scenic river. Int. J. Sustainable Dev. World Ecol., Vol: 20, No: 3, s.216–222.

[51] **Wu, C.H., Chang, Y.T., and Tseng, Y.C.**, 2010. Multi-screen cyber–physical video game: an integration with body-area inertial sensor networks, in: Int'l Conf. Pervasive Computing and Communications, s. 832–834.

[52] **Deng, X., and Yang, Y.,** 2013. Communication Synchronization in Cluster-Based Senser Networks for Cyber-Physical Systems, IEEE Emerging Topics in Computing, DOI: 10.1109/TETC.2013.2273219.

[53] **Lo, C.H., Peng, W.C., Chen, C.W., Lin, T.Y., and Lin, C.S.**, 2008. CarWeb: a traffic data collection platform, in: Int'l Conf. Mobile Data Management, s.221–222.

[54] **Gelenbe, E.,** 2013. Future Research on Cyber-Physical Emergency Management Systems, Future Internet '13, s.336-354; doi:10.3390/fi5030336.

[55] **Mark, Y., Nandakishore, K., Harkirat, S., Anand, R., York, L., and Suresh, S.,** 2005. Exploiting Heterogeneity in Sensor Networks, 24th Annual Joint Conference of the IEEE Computer and Communications Societies, Miami, USA, 13-17 March, doi: 10.1109/INFCOM.2005.1498318.

[56] tinyos.stanford.edu/tinyos-wiki/index.php/Imote2

[57] https://www.eol.ucar.edu/isf/facilities/isa/internal/CrossBow/DataSheets/stargate.pdf

[58] http://www.memsic.com/userfiles/files/Datasheets/WSN/telosb_datasheet.pdf

[59] **Levis, P.,** 2005. TinyOS: An Operating System for Sensor Networks, Ambient Intelligence, Berlin Heidelberg, pp. 115–148.

[60] **Dunkels, A., Gronvall, B., Voigt, T.,** 2004. Contiki - a lightweight and flexible operating system for tiny networked sensors, 29th Annual IEEE International Conference on Local Computer Networks, pp. 455–462.

[61] **Baccelli, E.,** 2013. RIOT OS: Towards an OS for the Internet of Things, 32nd IEEE INFOCOM. NJ, USA, pp. 354-362.

[62] **Islam, M., Mehedi, H.M., Lee, G., and Huh, E**., 2012. A Survey on Virtualization of Wireless Sensor Networks, Sensors, Vol: 12, s. 2175-2207, doi:10.3390/s120202175.

[63] **Sudip, M., Subarna, C., Obaidat, M.S**., 2014. On Theoretical Modeling of Sensor Cloud: A Paradigm Shift From Wireless Sensor Network, IEEE Systems Journal, Vol:22, doi: 10.1109/JSYST.2014.2362617.

[64] **Delgado, C., JoséRamón, G. , Canales, M. , Ortín, J., Sonda, B., and Matteo, C**., 2016. On optimal resource allocation in virtual sensor networks, Ad Hoc Networks, Vol: 50 p.p: 23–40, doi:10.1016/j.adhoc.2016.04.004.

[65] **Eugster, P. T., Felber, P. A., Guerraoui, R., and Kermarrec, A.-M.,** 2003. The many faces of publish/subscribe, ACM Comput. Surveys, vol. 35, no. 2, pp. 114–131.

[66] **Hassan, M. M., Song, B., and Huh, E.-N.,** 2009. A framework of sensor-cloud integration opportunities and challenges, in Proc. Int. Conf. Ubiquitous Inf. Manage. Commun., pp. 618–626.

[67] **Kumar, L. D.,** 2012. Data filtering in wireless sensor networks using neural networks for storage in cloud, Recent Trends In Information Technology (ICRTIT), Tamil, India, April 19-21, pp. 202–205.

[68] **Khan ,I., Belqasmi, F., Glitho, R., and Crespi, N.,** 2013. "A multi-layer architecture for wireless sensor network virtualization," in Wireless and Mobile Networking Conference (WMNC), 2013 6th Joint IFIP, Dubai, UAE, pp. 1–4.

[69] **Bandara, H. M. N. D., Jayasumana ,A. P., and Illangasekare, T. H.,** 2008.Cluster Tree Based Self Organization of Virtual Sensor Networks, in 2008 IEEE GLOBECOM Workshops, pp. 1–6.

[70] http://www.wisebed.eu, En son ziyaret 11.10.2017

[71] **Eggert, M.**, 2013. SensorCloud: Towards the interdisciplinary development of a trustworthy platform for globally interconnected sensors and actuators, RWTH Aachen Univ., Aachen, Germany.

[72] **Lai, C.-F., Lai Y.-X., Chao H.-C., and Wan J.,** 2013. Cloud-assisted realtime transrating for http live streaming, IEEE Wireless Commun. Mag., vol. 20, no. 3, pp. 62–70.

[73] **Lai, C.-F., Wang, H., Chao H.-C., and Nan, G.,** 2013. A network and device aware QoS approach for cloud-based mobile streaming, IEEE Trans. Multim., vol. 15, no. 4, pp. 747–757.

[74] **Yuriyama, M., and Kushida, T.**, 2010. Sensor-cloud infrastructure physical sensor management with virtualized sensors on cloud computing, in Proc. Int. Conf. Netw.-Based Inf. Syst., pp. 1–8.

[75] **Zhengguo, S., Hao, W., Changchuan, Y., , Xiping, H., Shusen, Y., and Leung, M.,** 2015. Lightweight Management of Resource-Constrained Sensor Devices in Internet of Things, IEEE Internet Of Things Journal, Vol. 2, No. 5.

[76] **Sarakis, L., Zahariadis, T., Leligou, H.-C., and Dohler, M.**, 2012. A framework for service provisioning in virtual sensor networks, J Wireless Com Network, vol. 2012, no. 1, pp. 1–19.

[77] **Hoebeke, J., et al.,** 2011. Managed Ecosystems of Networked Objects, Wireless Pers Commun, vol. 58, no. 1, pp. 125–143.

[78] **Ishaq, I., Hoebeke, J., Moerman, I., and Demeester, P.**, 2012. Internet of Things Virtual Networks: Bringing Network Virtualization to Resource-Constrained Devices, in 2012 IEEE International Conference on Green Computing and Communications (GreenCom), pp. 293–300.

[79] **De Poorter, E., Troubleyn, E., Moerman, I., and Demeester, P.,** 2011. IDRA: A Flexible System Architecture for Next Generation Wireless Sensor Networks, Wirel. Netw., vol. 17, no. 6, pp. 1423–1440.

[80] **Kohler, E., Morris, R., Chen, B., Jannotti, J., and Kaashoek, M. F.**, 2000. The Click Modular Router, ACM Trans. Comput. Syst., vol. 18, no. 3, pp. 263–297.

[81] **Evensen, P., and Meling, H.,** 2009. Sensewrap: A service oriented middleware with sensor virtualization and self-configuration, 5th International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 261–266.

[82] http://www.zeroconf.org/, En son ziyaret 10.10.2017

[83] **Luu, H. V., and Xueyan, T.,** 2013. Constructing rings overlay for robust data collection in wireless sensor networks, Journal of Network and Computer Applications 36.5 (2013): 1372-1386.

[84] http://sensors.cs.umass.edu/projects/senseye/, En son ziyaret 10.10.2017

[85] **Anthony, D., WoodLeo, S., Stankovic, A.,** 2008. SenQ: An Embedded Query System for Streaming Data in Heterogeneous Interactive Wireless Sensor Networks, Distributed Computing in Sensor Systems, Vol:37, s.531-543.

[86] **Saruwatari, S., Suzuki, M., and Morikawa, H.,** 2012. PAVENET OS: A Compact Hard Real-Time Operating System for Precise Sampling in Wireless Sensor Networks, SICE Journal of Control, Measurement, and System Integration, vol. 5, pp. 24–33.

[87] **Cao, Q., Abdelzaher, T., Stankovic, J., and He, T.,** 2008. "The LiteOS Operating System: Towards Unix-Like Abstractions for Wireless Sensor Networks," in

International Conference on Information Processing in Sensor Networks, IPSN '08, pp. 233–244.

[88] **Fok, C.-L., Roman, G.-C., and Lu, C.,** 2009. Agilla: A Mobile Agent Middleware for Self-adaptive Wireless Sensor Networks, ACM Trans. Auton. Adapt. Syst., vol. 4, no. 3, pp. 16:1–16:26.

[89] **Levis, P., and Culler, D.,** 2009. Maté: A tiny virtual machine for sensor networks. In ASPLOSX: Proceedings of the 10th International Conference on Architectural Support for Programming Languages and Operating Systems, San Jose, CA, pp. 85-95.

[90] **Koshy, J., and Pandey, R.,** 2005. VMSTAR: Synthesizing Scalable Runtime Environments for Sensor Networks, in Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems, New York, NY, USA, pp. 243–254.

[91] **Simon, D.**, 2006. Java on the Bare Metal of Wireless Sensor Devices: The Squawk Java Virtual Machine, in Proceedings of the 2nd International Conference on Virtual Execution Environments, New York, NY, USA, pp. 78–88.

[92] http://www.sentilo.io, En son ziyaret 10.10. 2017

[93] http://www.libelium.com, En son ziyaret 10.10. 2017

[94] http://www.iotsens.com/en, En son ziyaret 10.10. 2017

[95] **Serrano, E., and Arsénio, A.,** 2012. Cloud Framework for Wireless Sensor Networks. Master Thessis, Technico Lisboa.

[96] https://www.amazon.com/clouddrive, En son ziyaret 10.10. 2017

[97] http://sensorrush.com, En son ziyaret 10.10. 2017

[98] http://www.digi.com/products/cloud/digi-device-cloud, En son ziyaret 10.10. 2017

[99] http://www.nimbits.com, En son ziyaret 10.10. 2017

[100] https://thingspeak.com, En son ziyaret 10.10. 2017

[101] http://www.pachube.com, En son ziyaret 10.10. 2017

[102] http://hadoop.apache.org/, En son ziyaret 10.10. 2017

[103] https://spark.apache.org, En son ziyaret 10.10. 2017

[104] www.pachyderm.io, En son ziyaret 10.10. 2017

[105] https://cloud.google.com/bigquery/, En son ziyaret 10.10. 2017

[106] **White, T.,** 2015. Hadoop: The Definitive Guide, O'Reilly, Cambridge,USA.

[107] **Medlej, M.,** 2014. Big data management for periodic wireless sensor networks, Doctoral Thesis, Universite de Franche-Comte, France.

[108] https://flume.apache.org, En son ziyaret 10.10. 2017

[109] https://zookeeper.apache.org, En son ziyaret 10.10. 2017

[110] https://hive.apache.org, En son ziyaret 10.10. 2017

[111] https://pig.apache.org, En son ziyaret 10.10. 2017

[112] https://hortonworks.com/apache/sqoop/, En son ziyaret 10.10. 2017

[113] https://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html, En son ziyaret 10.10. 2017

[114] https://hadoop.apache.org › Hadoop/hadoop-yarn, En son ziyaret 10.10. 2017

[115] https://hbase.apache.org, En son ziyaret 10.10. 2017

[116] www.opengeospatial.org/standards/sensorml, En son ziyaret 10.10. 2017

[117] https://memcached.org/, En son ziyaret 10.10. 2017

[118] **Hong, G., Xiaolin, F., Jianzhong, L., Yingshu, L.,** 2015. Data Collection in Multi-Application Sharing Wireless Sensor Networks, IEEE Transactions On Parallel And Distributed Systems, Vol. 26, No. 2.

[119] **Hou, T., Shi ,Y., and Sherali, H.**, 2008. Rate allocation and network lifetime problems for wireless sensor networks, IEEE/ACM Trans. Netw. 16 (2),pp. 321–334.

[120] **Shi, Y., Hou, T., Liu, J., and Kompella, S.**, 2013. Bridging the gap between protocol and phys- ical models for wireless networks, IEEE Trans. Mobile Comput. 12 (7) pp.1404–1416.

[121] **Suh, C., Joung, J.-E., and Ko, Y.-B.,** 2007. New RF models of the tinyos simulator for IEEE 802.15.4 standard, in: Proceedings IEEE WCNC 2007, Hong Kong, pp. 2238–2242

[122] **Yıldırım, G., and Tatar, Y.,** 2017. Alternative Resource Allocation Model for Dynamic Resource-Sharing WSN Systems, Fifth International Conference on Advances in Computing, Electronics and Communication - ACEC 2017, Rome, Italy, 24-27 May, doi: 10.15224/978-1-63248-121-4-04

[123] **Keshtgary, M., Mansouri, M.R., Mohammadi, R., Mahmoudi, M.,** 2012. Analytical Model of Energy Consumption in Cluster based Wireless Sensor Networks with Data Aggregation using M/M/1 Queuing Model,

International Journal of Computer Applications (0975 – 8887) Vol:53, No:13, s.1-6.

[124] **Lu, Z., and Yang, H.Y.,** 2012. Unlocking the Power of OPNET Modeler, Cambridge Press, Cambridge, UK.

[125] https://www.openstack.org/, En son ziyaret 10.10. 2017

[126] **Ionescu, V.M.,** 2015. The analysis of the performance of RabbitMQ and ActiveMQ, RoEduNet International Conference - Networking in Education and Research (RoEduNet NER), 24-26 September, Craiova, Romanya, doi: 10.1109/RoEduNet.2015.7311982.

[127] **Nguyen, C.N., Kim, J.S., Hwang, S.,** 2016. KOHA: Building a Kafka-Based Distributed Queue System on the Fly in a Hadoop Cluster, Foundations and Applications of Self Systems, IEEE International Workshops, 12-16 September, doi: 10.1109/FAS-W.2016.23

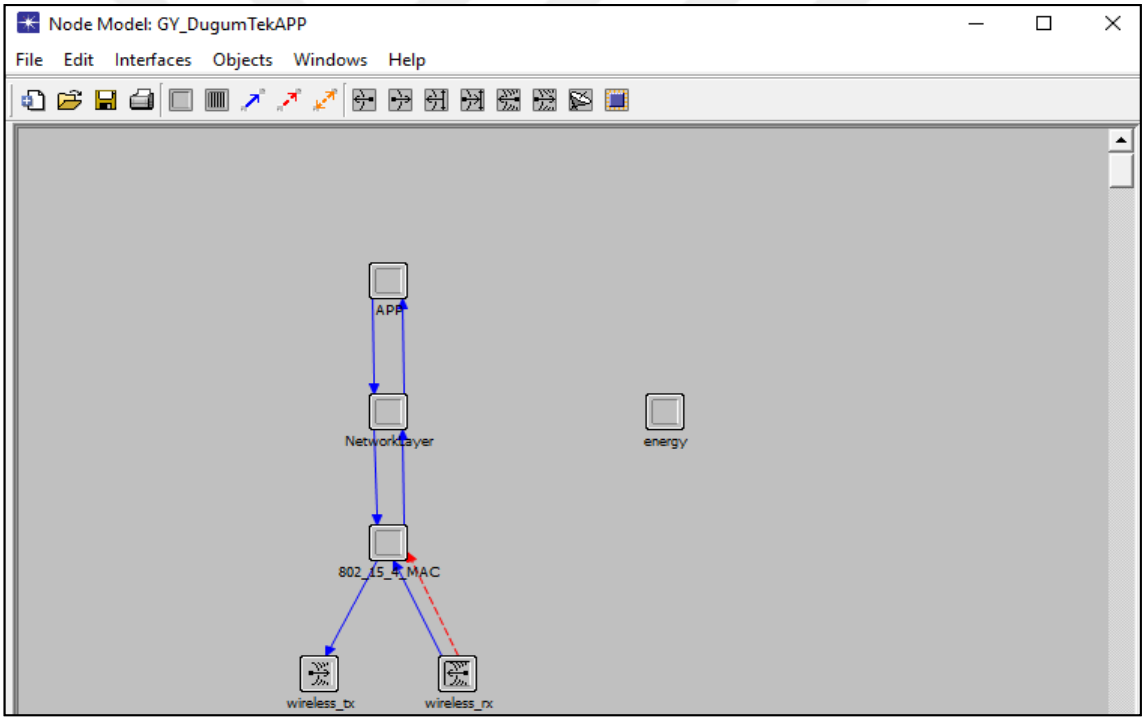
[128] **Bayardo, R.J., Ma, Y., and Srikant, R.,** 2007. Scaling up all pairs similarity search. 16th international conference on World Wide Web, WWW'07, 8-12 May, Banff, Canada, s.131–140.

[129] **Yıldırım, G., TATAR Y.,** 2017. On WSN heterogeneity in IoT and CPSs. Computer Science and Engineering (UBMK), 2017 International Conference, 5-8 Oct. Antalya, Turkey, doi: 10.1109/UBMK.2017.8093421

EKLER

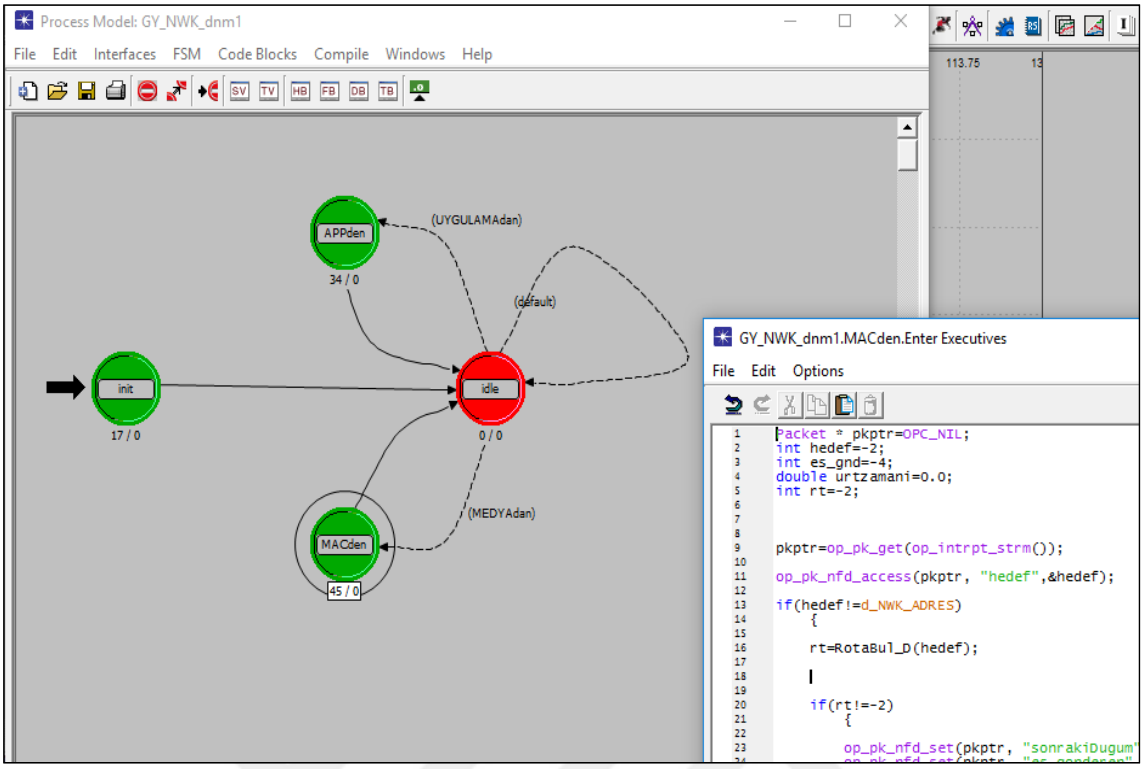
EK 1. OPNET WSN Düğüm Modeli

Tez çalışması kapsamında yapılan OPNET simülasyonlarında kullanılan WSN düğüm modelleri gerçek düğüm modellerine ve Bölüm 4 ‘deki matematiksel modellere göre yapılmıştır. OPNET WSN genel düğüm modeli Şekil 1 ‘de verilmiştir. Şekilden de görüleceği üzere düğüm fiziksel, MAC, ağ ve uygulama katmanlarından oluşmaktadır. Katmalar arası iletişim için erişim primitivleri geliştirilmiştir.

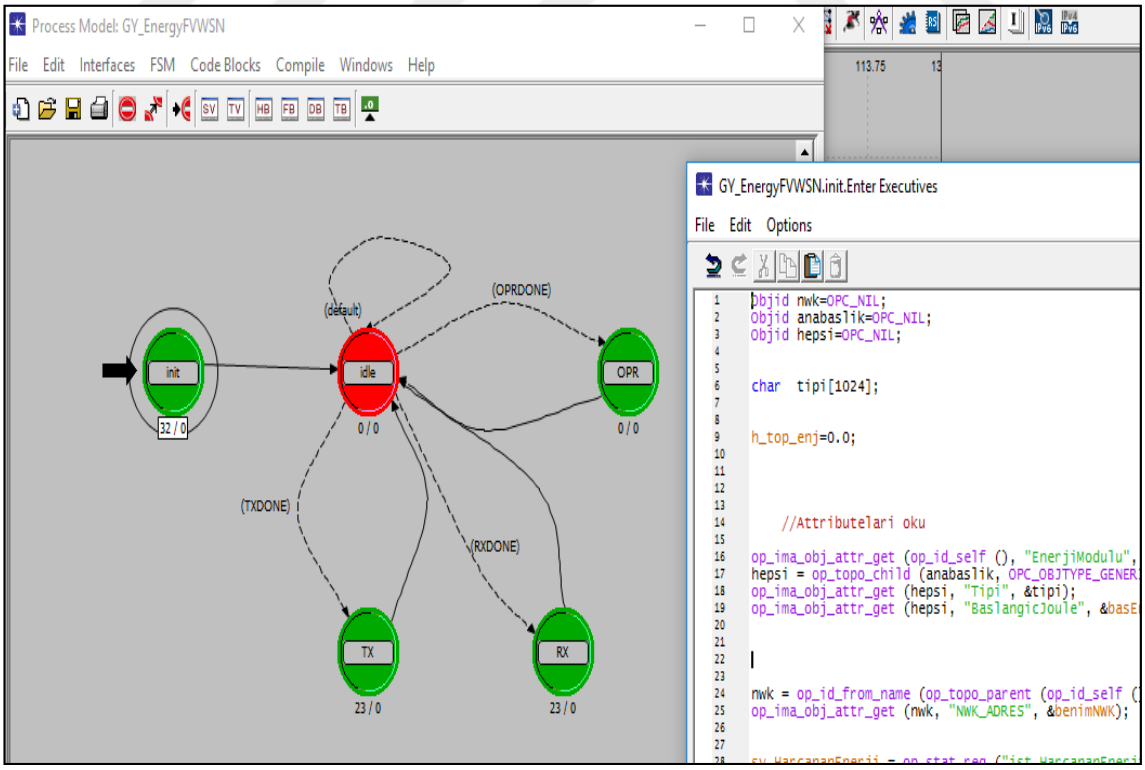


Ek Şekil 1. OPNET ‘de geliştirilen genel WSN düğüm modeli

Düğüm uygulama katmanı tezde kullanılan senaryolara göre tasarlanmış ve Bölüm 5 ‘de detay bilgileri verilmiştir. WSN düğüm modelinde kullanılan ağ katmanı FSM Şekil 2 ‘de verilmiştir. Diğer bir önemli modül ise Enerji modülüdür. Bu modüle ait genel FSM diyagramı ise Şekil 3 ‘de sunulmuştur. Enerji modülü veri alım ve veri gönderim için Bölüm 4 ‘de verilen matematiksel modellere göre çalışmaktadır.

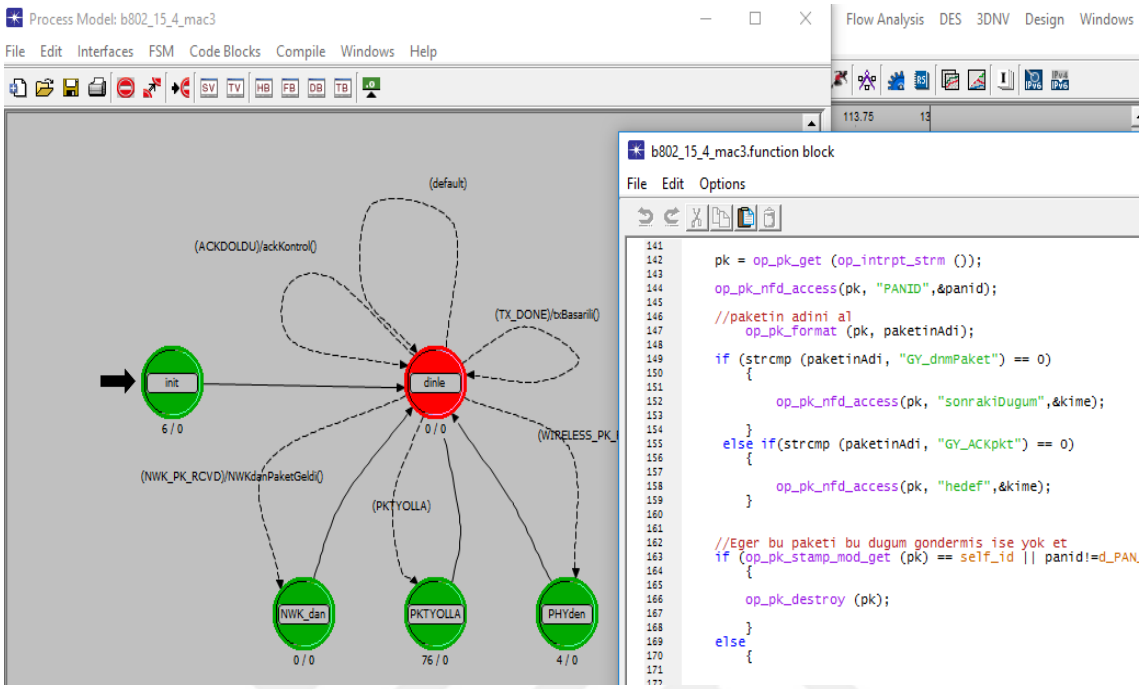


Ek Şekil 2. NWK katmanı FSM

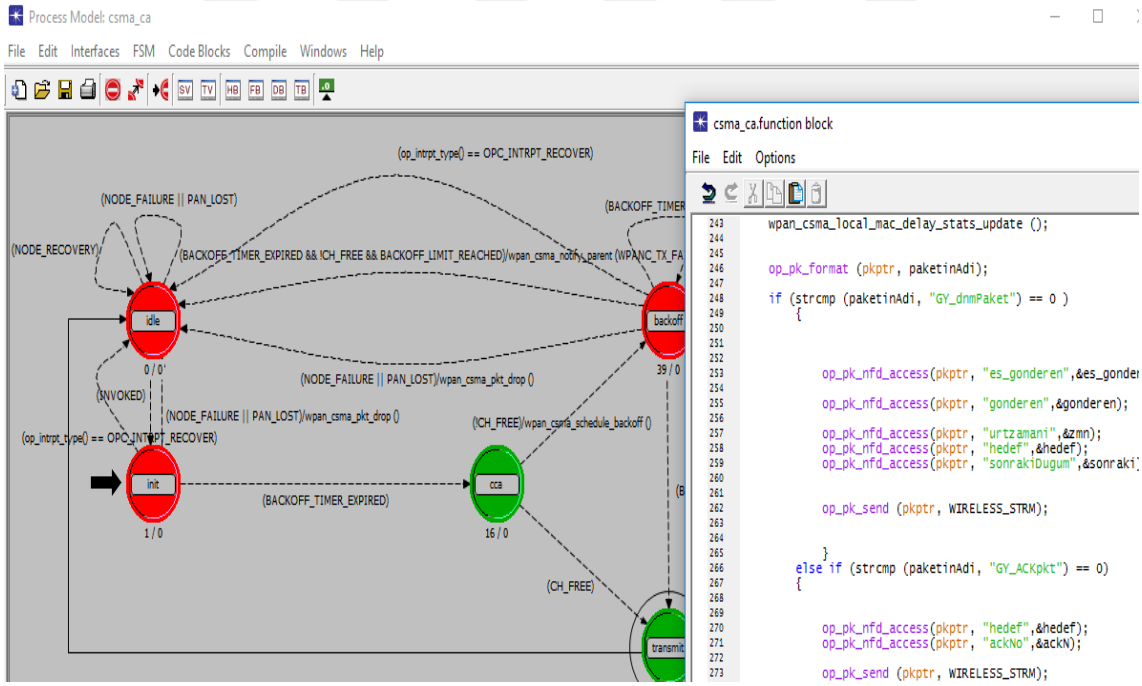


Ek Şekil 3. Enerji katmanı FSM

Düğüm modelinde kullanılan MAC modeli, pratik alanda sıklıkla kullanılan yarışma temelli 802.15.4 olup bu modeli gerçekleyen FSM ler Şekil 4 a ve b ‘de verilmiştir.



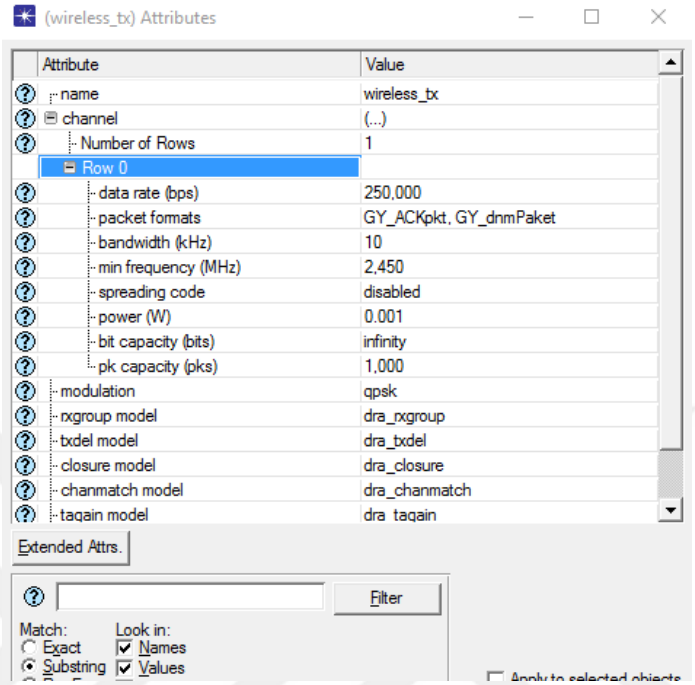
a) MAC FSM



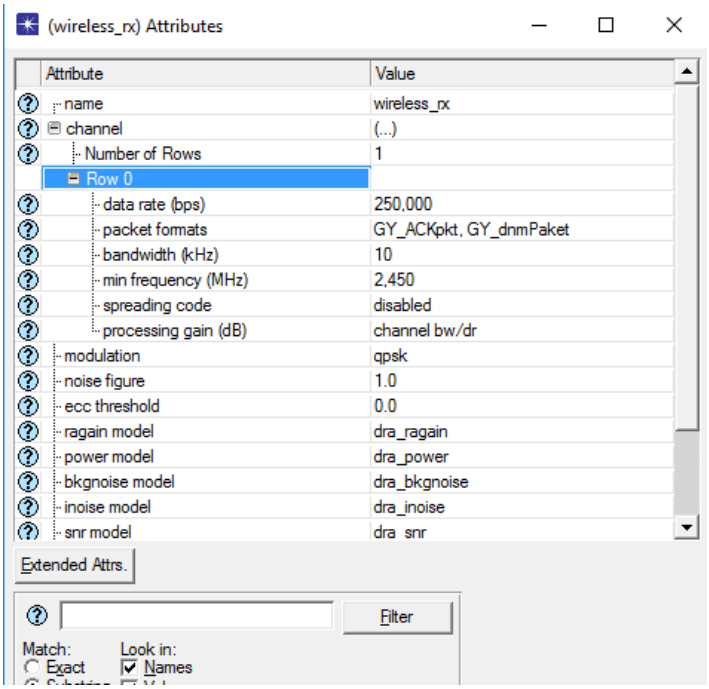
b)) CSMA FSM

Ek Şekil 4. Düğüm için FSM’ler

WSN düğüm modellerinde kullanılan fiziksel katman öznitelikleri 802.15.4 ‘de kullanılan değerlere uygun olarak atanmıştır. Bu öznitelik verileri ise Şekil 5 a ve b ‘de verilmiştir.



a) Tx öznitelikleri



b) Rx öznitelikleri

Ek Şekil 5. PHY katman öznitelikleri

EK 2. FVWSN Yazılım Çerçevesinin Kullanımı ve Ön Tanımlı MapReduce Uygulamalarının Çalıştırılmasına Ait Örnek Uygulama

FVWSN yazılım çerçevesi ile WSN sanallaştırma uygulamasında ilk olarak Tip A istemcilerin kullanacakları kaynakları web sayfasından seçmeleri, sisteme üye olmaları ve giriş yapmaları gerekmektedir. Daha sonra istemcinin FVWSN yazılım çerçevesini web sayfasından indirmesi gerekmektedir. Ek Şekil 5 a-c 'de bu adımlar göstermektedir.

a) FWSN IoT Web sitesi ana sayfası

b) Üye girişi

c) FVWSN yazılım çerçevesinin indirilmesi

Ek Şekil 6. FVWSN temelli WSN sanallaştırma için başlangıç adımları

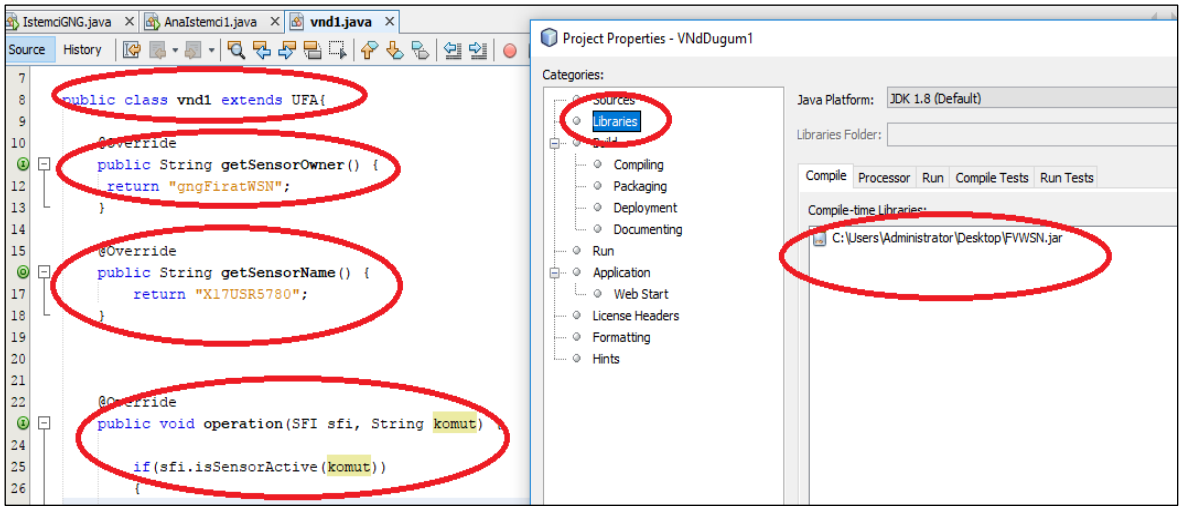
Daha sonra oluşturulacak VNd için bir ID, web sayfasındaki "Servisler" kısmından alınır ve bu işlemin başarılı olduğu sistemden kontrol edilir. Bu adımlar Ek Şekil 6 a-b 'de verilmiştir.

a) VNd için sistemden ID alınması

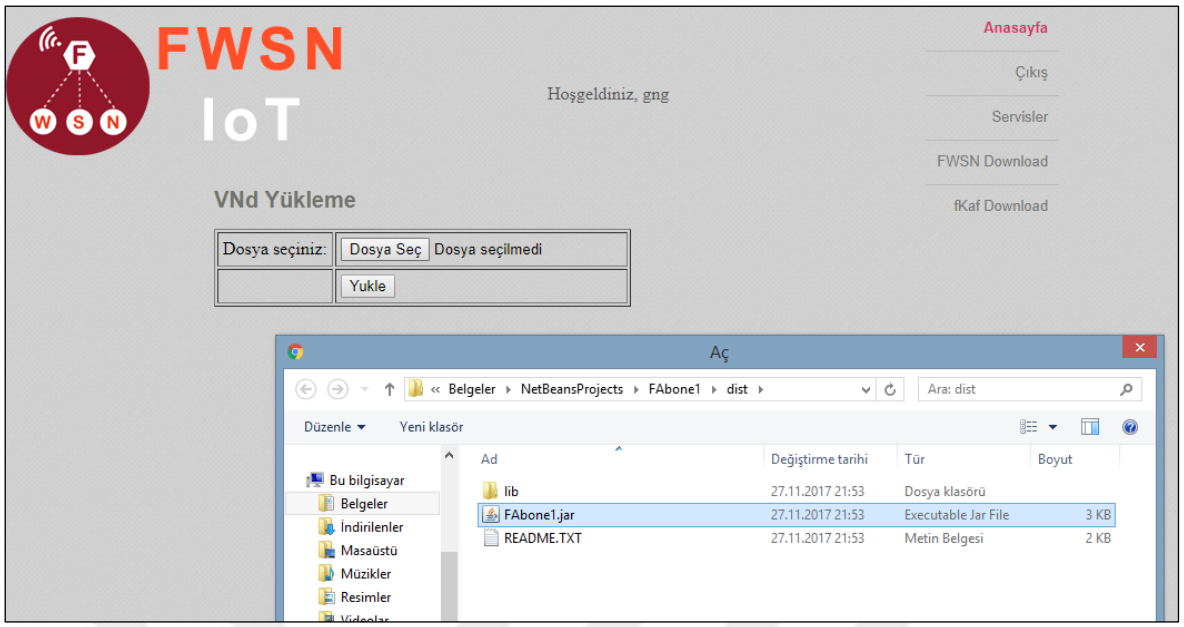
b) VNd için sistemden ID alınması

Ek Şekil 7. Alınan VNd ID 'nin listeden kontrol edilmesi

VNd 'nin oluşturlması için FVWSN yazılım çerçevesi ilgili IDE ortamına eklenmektedir. FVWSN yazılım çerçevesinin sağladığı UFA soyut sınıfından istemci VNd düğümü sınıfı türetilir.Bu sınıftaki operation ()" sınıfı ve diğer gerekli metotlar "implement" edilirek sisteme web üzerinden tekrar sisteme yüklenir. Bu adımlar Ek Şekil 7 a ve b 'de gösterilmiştir.



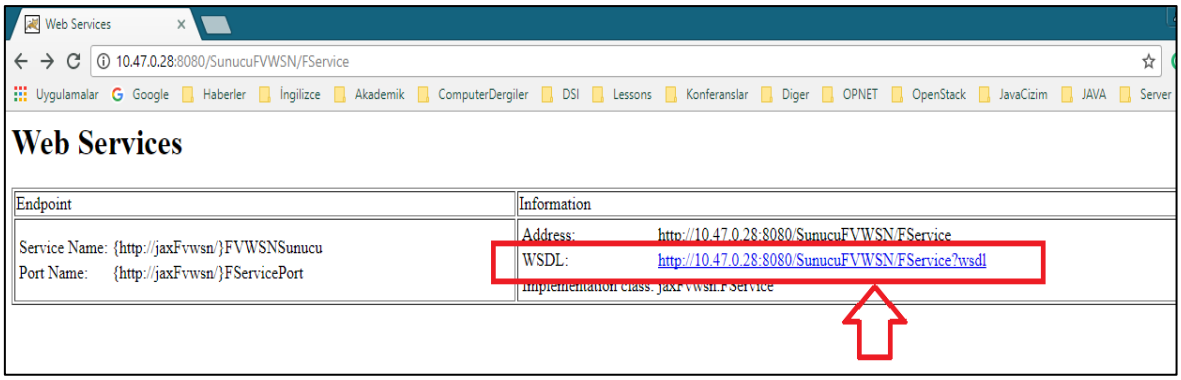
a) FVWSN 'nin IDE 'ye eklenmesi ve UFA soyut sınıfı ile örnek VNd kodlaması



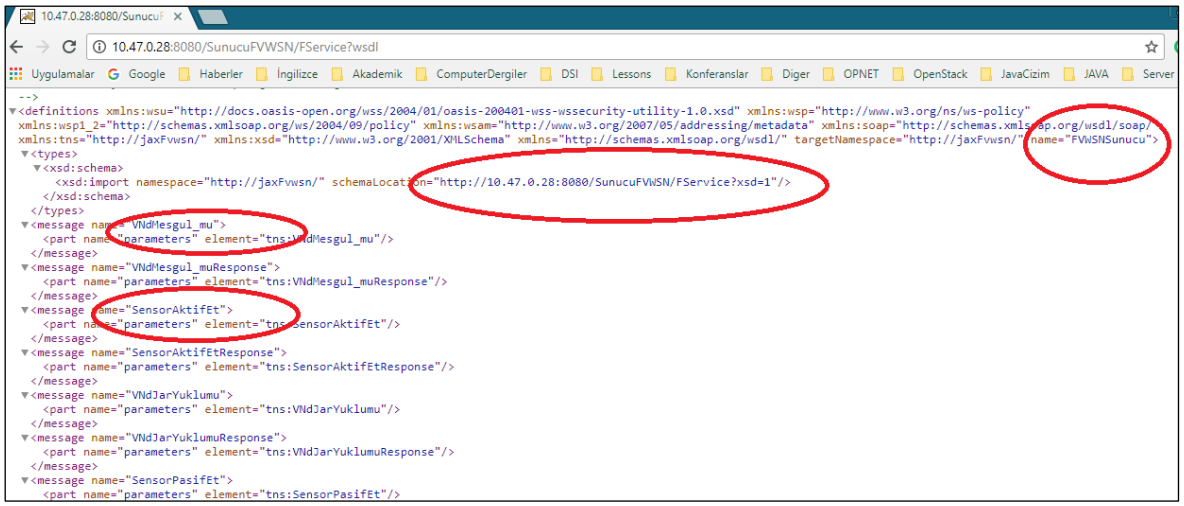
b) VNd 'yi içeren jar dosyasının sisteme yüklenmesi

Ek Şekil 8. VNd oluşturlması ve sisteme yüklenmesi

Son olarak bu VNd nin kontrol edilmesi için SOAP temelli web servis ile bağlantı gerçekleştirilerek ilgili web metotlarına ulaşılır. Öncelikle ilgili web servisin wsdl adresi elde edilir. Gerekirse bu wsdl adres üzerinden ilgili web metotlar görülebilir. Bu web metotlar ilgili wsdl adresi ile bir web servis istemcisi oluşturulması ile projeye otomatik dahil edilebilir. Bu adımlar Ek Şekil 8 'de verilmiştir.



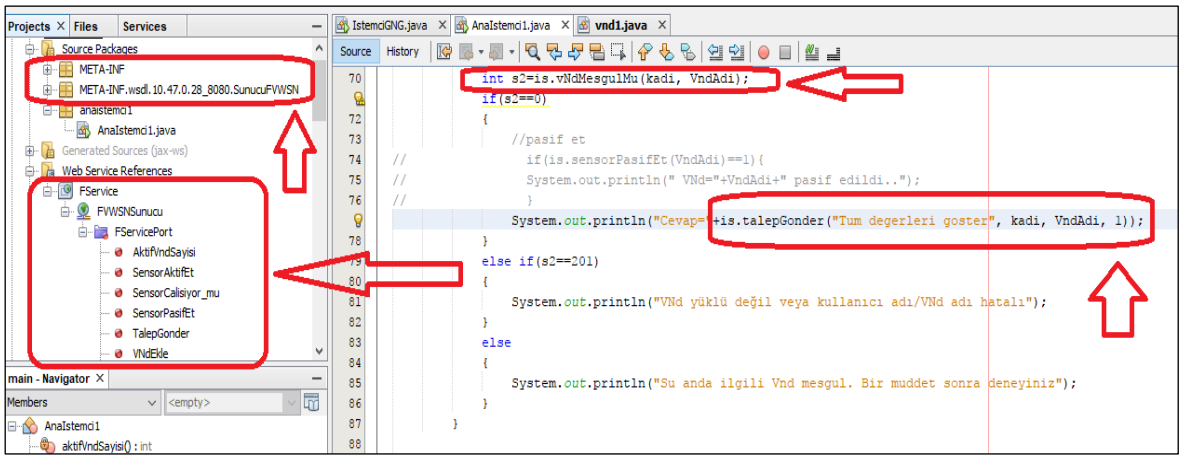
a) Web servis wsdl adresi



b) WSDL adrsinde tanımlı web metotlar

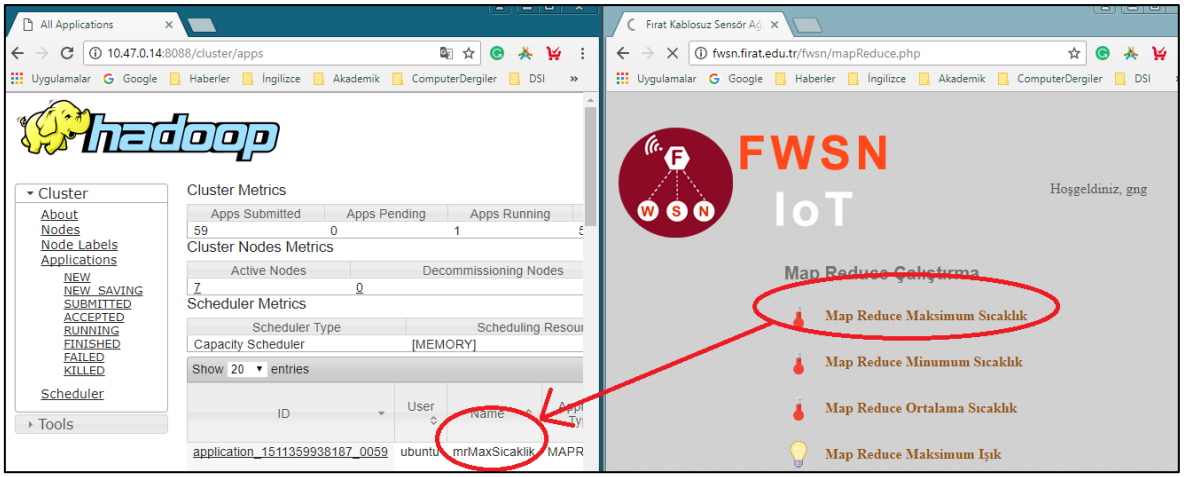
Ek Şekil 9. Web servis wsdl adresi ve tanımlı web metotlar

Bu wsdl adresi üzerinden projeye dahil edilen web metolar ilgili projeye dahil edilerek sisteme yüklenen VNd kontrol edilebilir. Bu örnekte "talepGonder()" web metodu VNd 'ye istemci tanımlı komut göndermek için kullanılmaktadır. Diğer metotlar ise VNd 'nin uzaktan yönetimi ve kontrolü için kullanılmaktadır. Genel bir VND kontrol uygulaması kod bloğu Ek Şekil 9 'da verilmiştir.

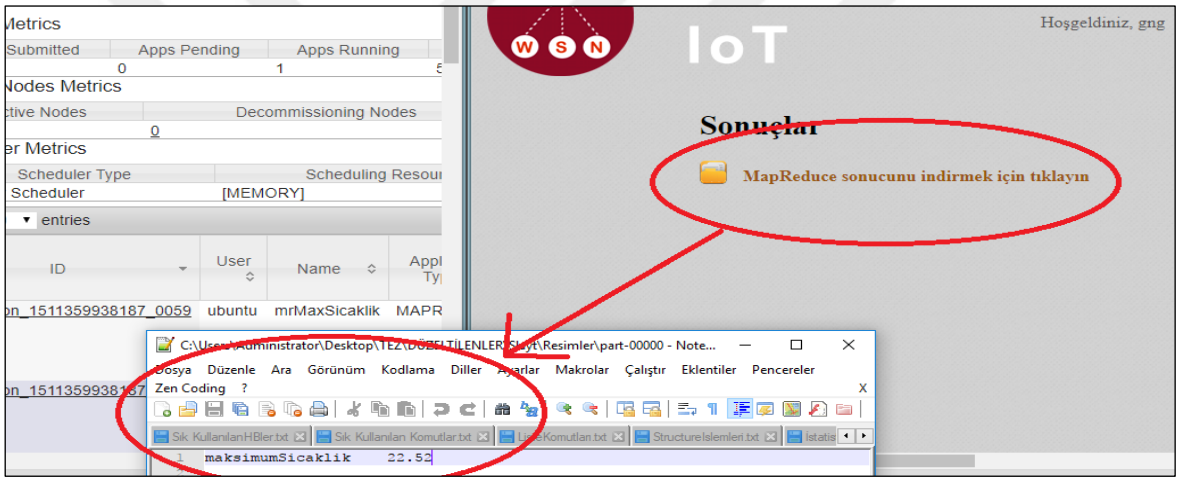


Ek Şekil 10. VNd kontrol programına ait örnek kodlama bloğu ve ilgili web servislerin projeye eklenmesi

Bu tez kapsamında oluşturulan FWSN IoT WSN bulut sistemi üzerinde aynı zamanda istemcilere ait ön tanımlı MapReduce uygulamaları da uzaktan çalıştırlabilmektedir. Bu MapReduce uygulamaları istemci VNd 'leri ile aktif edilebilirken aynı zamanda istemci web arayüzünden de aktif edilebilinir. Bu işlem Ek Şekil 10 'da gösterildiği şekilde yapılabilinir.



a) İstemci MapReduce 'lerinin manuel olarak istemci web sayfasından etkinleştirilmesi



b) İstemci MapReduce sonuçlarının elde edilmesi

Ek Şekil 11. Web servis wsdl adresi ve tanımlı web metotlar

ÖZGEÇMİŞ

Güngör YILDIRIM

DSİ 9. BÖLGE MÜDÜRLÜĞÜ

Barajlar Şubesi

Elazığ

0 424 238 6911/1713

E-posta: gyildirim@dsi.gov.tr

1978	:	Elazığ'da doğdu.
1992-1995	:	Mehmet Akif Ersoy Lisesi'ni bitirdi.
2000	:	Fırat Üniversitesi Mühendislik Fakültesi Elektrik–Elektronik Bölümünden mezun oldu.
2000-2005	:	Fırat Üniversitesi Tunceli MYO'da Öğretim Görevlisi olarak çalıştı.
2005-..	:	DSİ 9. Bölge Müdürlüğü Barajlar Şubesinde Elektrik-Elektronik Müh. olarak çalışmaktadır.
2012	:	Fırat Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Ana Bilim Dalı Donanım Bilim Dalında Yüksek Lisansını tamamladı
2013 -…	:	Fırat Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Ana Bilim Dalı Donanım Bilim Dalında Doktora çalışmalarına başladı