

KARADENİZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

KOMŞU 6TİSCH AĞLARIN DENGELENMESİ

YÜKSEK LİSANS TEZİ

Erkan DEMİR

NİSAN 2024
TRABZON



KARADENİZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

KOMŞU 6TİSCH AĞLARIN DENGELENMESİ

Erkan DEMİR

Karadeniz Teknik Üniversitesi Fen Bilimleri Enstitüsünde
"BİLGİSAYAR YÜKSEK MÜHENDİSİ"
Unvanı Verilmesi İçin Kabul Edilen Tezdir.

Tezin Enstitüye Verildiği Tarih : 04 / 03 / 2024

Tezin Savunma Tarihi : 26 / 04 / 2024

Tez Danışmanı : Doç. Dr. Sedat
GÖRMÜŞ O

Trabzon 2024

ÖNSÖZ

“KOMŞU 6TİSCH AĞLARIN DENGELENMESİ” isimli bu tez Karadeniz Teknik Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı, Yüksek Lisans Programı’nda hazırlanmıştır.

Tez çalışmam boyunca bana zaman ayırıp bilgi ve birikimleriyle tezimde bana yol gösterip ışık tutan, saygıdeğer danışman hocam Sayın Doç. Dr. Sedat GÖRMÜŞ’e teşekkürlerimi sunarım. Ayrıca mesai kavramı olmaksızın bana yardımcı olup birikimlerini paylaştan değerli arkadaşım Arş. Gör. Hakan AYDIN ve danışman hocamın yüksek lisans ve doktora öğrencisi olan arkadaşlarıma da teşekkür ederim.

Tezimin hazırlanma sürecinde, bana sürekli destek olup motivasyon katan sevgili eşim Elif DEMİR’e, hayatım boyunca maddi ve manevi destekleriyle her zaman yanımda olan kıymetli annem, babam, ağabeyim ve kardeşime teşekkür ederim.

Bu tez çalışmasının bundan sonraki çalışmalara katkı sağlamasını temenni ederim.

Erkan DEMİR

Trabzon 2024

TEZ ETİK BEYANNAMESİ

Yüksek Lisans Tezi olarak sunduğum “KOMŞU 6TİSCH AĞLARIN DENGELENMESİ” başlıklı bu çalışmayı baştan sona kadar danışmanım Doç. Dr. Sedat GÖRMÜŞ’ün sorumluluğunda tamamladığımı, verileri/örnekleri kendim topladığımı, deneyleri/analizleri ilgili laboratuvarlarda yaptığımı/yaptırdığımı, başka kaynaklardan aldığım bilgileri metinde ve kaynakçada eksiksiz olarak gösterdiğimi, çalışma sürecinde bilimsel araştırma ve etik kurallara uygun olarak davrandığımı ve aksinin ortaya çıkması durumunda her türlü yasal sonucu kabul ettiğimi beyan ederim. 26/04/2024

Erkan DEMİR

İÇİNDEKİLER

	<u>Sayfa No</u>
ÖNSÖZ	III
TEZ ETİK BEYANNAMESİ.....	IV
İÇİNDEKİLER.....	V
ÖZET	VIII
SUMMARY	IX
ŞEKİLLER DİZİNİ.....	X
TABLolar DİZİNİ.....	XI
SEMBOLLER VE KISALTMALAR DİZİNİ	XII
1. GENEL BİLGİLER	1
1.1. Giriş.....	1
1.2. Motivasyon	2
1.3. Tezin Yapısı	3
2. NESNELERİN İNTERNETİ.....	4
2.1. Genel Bakış ve Kullanılan Teknolojiler.....	5
2.1.1. IEEE 802.15.4 PHY	6
2.1.2. IEEE 802.15.4e (TSCH)	8
2.1.3. TiSCH Çizelgesi	8
2.1.4. 6TiSCH	11
2.1.5. 6TiSCH Operation Sublayer (6top)	12
2.1.6. 6LoWPAN	13
2.1.7. CoAP.....	14
2.1.8. LTE (Long-Term Evolution)	16
2.1.9. SDN (Software Defined Networking).....	18
2.2. Sistem Mimarisi	19
2.2.1. Kullanılan İşletim Sistemleri	19
2.2.2. Contiki İşletim Sistemi.....	20
2.2.3. Contiki-NG.....	22
2.2.4. Cooja	23
3. YÖNLENDİRME PROTOKOLLERİ.....	26
3.1. Kablolu Ağ Yönlendirme Protokolleri:	26

3.1.1.	OSPF (Open Shortest Path First)	26
3.1.2.	EIGRP (Enhanced Interior Gateway Routing Protocol)	27
3.1.3.	RIP (Routing Information Protocol)	27
3.2.	Kablosuz Ağ Yönlendirme Protokolleri	28
3.2.1.	802.11 (Wi-Fi) Protokolleri	28
3.2.2.	Mobile IP	29
3.2.3.	BGP (Border Gateway Protocol)	29
3.3.	RPL (Routing Protocol for Low-Power and Lossy Networks)	30
3.3.1.	RPL Yapısı	31
3.3.2.	ICMPv6 Mesajları	33
3.3.3.	DIO (DODAG Information Object - Yönlendirilmiş Asiklik Grafik Bilgi Nesnesi)	34
3.3.4.	DAO (Destination Advertisement Object - Hedef İlan Nesnesi)	34
3.3.5.	Yönlendirme Metrikleri	35
3.3.6.	RPL'nin Amaç Fonksiyonları	36
3.3.7.	OF-0 (Objective Function 0)	38
3.3.8.	OF-1 (MRHOF- Minimum Rank with Hysteresis Objective Function)	39
3.3.9.	ETX (Expected Transmission Count)	40
4.	LİTERATÜR İNCELEMESİ	41
5.	YAPILAN ÇALIŞMA	47
5.1.	Önerilen Mekanizmalar	48
5.1.1.	Komşu 6TiSCH Ağının Kurulumu	49
5.1.2.	Kurulum Mekanizması	49
5.1.3.	Çakışma (İnterferans)	51
5.1.4.	Algoritma-1 Sonuçları	52
5.1.5.	Kanal Seçim Mekanizması	54
5.1.6.	Ağ Dengeleme Algoritması	55
5.1.7.	Gelişmiş İşaretçiler (Enhanced Beacons - EB)	60
5.2.	Test Ortamı Özellikleri	60
5.2.1.	Başarım Metrikleri	62
5.2.3.	Paket İletim Oranı	62
5.2.4.	Ortalama Paket Gecikmesi	63
5.2.5.	Paket Kayıpları	64
6.	SONUÇLAR	65

6.1. Ağ Kurulum Sonuçları	65
6.2. Ağ Dengeleme Algoritması Sonuçları	66
6.2.1. Ağ Dengeleme ile Paket İletim Oranı (PDR) Sonuçları	67
6.2.2. Ağ Dengeleme ile Paket Hızına Göre Paket İletim Oranı (PDR) Sonuçları	70
6.2.3. Ağ Dengeleme Paket Kayıpları Sonuçları	72
7. BULGULAR VE SONUÇLAR.....	74
8. ÖNERİLER.....	76
9. KAYNAKLAR	77
ÖZGEÇMİŞ	



Yüksek Lisans Tezi

ÖZET

KOMŞU 6TiSCH AĞLARIN DENGELENMESİ

Erkan DEMİR

Karadeniz Teknik Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı
Danışman: Doç.Dr. Sedat GÖRMÜŞ
2024, 82 Sayfa

Günümüzde IoT cihazların kullanım alanlarının hızla artmasına bağlı olarak, Kablosuz sensör ağlarının (WSN) kurulumu, birbiriyle iletişimi ve ağ optimizasyonu önem kazanmıştır. 6TiSCH ağ ortamında kullanılan sensörler, kısıtlı enerji ve kısıtlı iletim menziline sahip düğümlerden meydana gelen bir yapıdır. Bu ağların çeşitli uygulamalarda kararlı ve performanslı çalışabilmesi için IETF 6TiSCH bünyesindeki RPL protokolü ile yönlendirme işlemleri yapılmaktadır. Aynı coğrafi ortamda bulunan birden fazla IoT ağında, düğümler komşularını ve ağdaki düğüm yoğunluğunu bilmeden yerleştikleri için, ağda kendilerine uygun bir kök düğüm seçim işlemi yaparak ona katılmaları gerekmektedir. Komşu ağlara dahil olmak isteyen yeni düğümler, ağların düğüm yoğunluğunu bilmeden ağa katılınca ağın verimliliği düşer, enerji tüketimi artar. Bu doğrultuda çözüm yaklaşımı olarak; RPL altyapısı ile geliştirilen algoritma ile düğümlerin dengeli ve dinamik bir şekilde ağa katılabilmeleri ve optimizasyonu hedeflenmiştir. Bu kapsamda, programlanan merkezi bir denetleyici ile, komşu 6TiSCH ağlara bağlanmak isteyen düğümlerin, daha az düğüm sayısı olan ağa katılabilmeleri, ağların farklı kanallar üzerinden çalışabilmesi ve ağa senkronize olan düğümlerinde ilgili kanallar üzerinden çalışabilmesi hedeflenmektedir.

Yapılan testler sonrasında komşu 6TiSCH ağlarda düğümlerin daha az yoğunluğa sahip ağa katılabilmeleri, bu sayede düğümlerin yönetimi ve ağın dengelenmesi sağlanmış, başarımleri, mesaj iletim miktarlarında büyük ölçüde iyileştirme elde edilmiştir.

Anahtar Kelimeler: IoT, RPL, 6TiSCH, Ağ Dengeleme.

Master Thesis

SUMMARY

BALANCING OF NEIGHBORING 6TISCH NETWORKS

Erkan DEMİR

Karadeniz Technical University
The Graduate School of Natural and Applied Sciences
Computer Engineering Graduate Program
Supervisor: Assoc. Prof. Sedat GÖRMÜŞ
2024, 82 Pages

With the rapid increase in the use cases of IoT devices, the deployment, communication, and network optimization of Wireless Sensor Networks (WSNs) have become crucial. Sensors used in 6TiSCH networks are characterized by nodes with limited energy and transmission range. To ensure stable and efficient operation of these networks in various applications, routing operations are performed using the RPL protocol within the IETF 6TiSCH framework. In environments with multiple IoT networks, nodes must select a suitable root node and join it without knowing their neighbors or the node density in the network. New nodes wishing to join neighboring networks can decrease the efficiency and increase energy consumption by randomly joining a network without knowing the node density. In this context, a solution approach aims to enable nodes to join the network in a balanced and dynamic manner using an algorithm developed with the RPL infrastructure, thus optimizing network performance. With a centrally programmed controller, the goal is for nodes wishing to connect to neighboring 6TiSCH networks to join the network with fewer nodes, allowing networks to operate on different channels and synchronized nodes in the network to operate on the relevant channels.

After testing, the goal was to enhance node management and balance in neighboring 6TiSCH networks by enabling nodes to join less dense networks, leading to significant improvements in performance metrics, especially in message transmission rates.

Key Words: IoT, RPL, 6TiSCH, Network Balancing.

ŞEKİLLER DİZİNİ

	<u>Sayfa No</u>
Şekil 2. Nesnelerin İnterneti	4
Şekil 2.1. IoT 6TiSCH Protokol Yığını	6
Şekil 2.1.2. Örnek TiSCH Çizelgesi	10
Şekil 2.1.6. CoAP Yapısı	14
Şekil 2.2.3. Contiki İşletim Sistemi Genel Mimarisi	23
Şekil 2.2.4. Cooja Simülatörü Arayüzüne Düğümlerin Eklenmesi	24
Şekil 2.2.5. Cooja Simülatörü Arayüzünde Kurulan Ağın İzlenmesi	24
Şekil 3.3.1. RPL Örneği	31
Şekil 3.3.4. RPL depolanmayan modu	35
Şekil 5.1.1. Komşu IoT Kurulumu	50
Şekil 5.1.4. Cooja Simülatörü ile Kurulan Komşu Ağ Yapısı ve Düğümler	53
Şekil 5.1.5. Cooja Simülatörü ile Kurulan Komşu Ağ Yapısı ve Atanan Kanallar	55
Şekil 5.1.6. Denetleyici ile 6TiSCH Ağının Dengelenmesi	56
Şekil 5.1.6.1. Ağ Dengeleme Algoritması	59
Şekil 5.2. Algoritma 1 Simülasyon Ortamı	61
Şekil 5.2.1. RPL Sınır Yönlendiricilere Kanal Ataması	62
Şekil 6.1. Düğümlerin Kanal Atamaları	65
Şekil 6.2. Denetleyici ile Kümelere Düğümlerin Tahsis Edilmesi	67
Şekil 6.2.1.1. Paket İletim Oranları (2 Sınır Yönlendirici & 4 Düğüm)	68
Şekil 6.2.1.2. Paket İletim Oranları (2 Sınır Yönlendirici & 10 Düğüm)	69
Şekil 6.2.1.3. Paket İletim Oranları (2 Sınır Yönlendirici & 20 Düğüm)	69
Şekil 6.2.2.1. Paket İletim Oranları (2 Sınır Yönlendirici & 20 Düğüm & 1 saniye)	70
Şekil 6.2.2.2. Paket İletim Oranları (2 Sınır Yönlendirici & 20 Düğüm & 5 saniye)	71
Şekil 6.2.2.3. Paket İletim Oranları (2 Sınır Yönlendirici & 20 Düğüm & 10saniye)	71
Şekil 6.2.3.1. Ortalama Paket Kayıpları (2 Sınır Yönlendirici & 4 Düğüm)	72
Şekil 6.2.3.2. Ortalama Paket Kayıpları (2 Sınır Yönlendirici & 10 Düğüm)	73
Şekil 6.2.3.3. Ortalama Paket Kayıpları (2 Sınır Yönlendirici & 20 Düğüm)	73
Şekil 7. Komşu IoT Ağı ve Düğümler	75

TABLÖLAR DİZİNİ

	<u>Sayfa No</u>
Tablo 1. Test Ortamı Özellikleri	60
Tablo 2. Ağ Kurulumu-Kanal Atama Sonuçları	66
Tablo 3. Farklı Düğüm Sayılarında Ağ Dengeleme Sonuçları	66



SEMBOLLER VE KISALTMALAR DİZİNİ

BGP	Border Gateway Protocol
CoAP	Constrained Application Protocol
DAG	Directed Acyclic Graph
DAO	Destination Advertisement Object
DIO	DODAG Information Object
DIS	DODAG Information Solicitation
DODAG	Destination Oriented Directed Acyclic Graph
EGP	Exterior Gateway Protocol
ETX	Expected Transmission Count
HTTP	Hyper-Text Transfer Protocol
ICMPv6	The Internet Control Message Protocol version 6
IEEE	Institute of Electrical & Electronic Engineers
IETF	Internet Engineering Task Force
IGP	Interior Gateway Protocol
IoT	Internet of Things
IP	Internet Protocol
IPv6	Internet Protocol Version 6
LLN	Low-Power and Lossy Networks
MAC	Media Access Control
MRHOF	Minimum Rank with Hysteresis Objective Function
OF	Objective Function
OF0	Objective Function Zero
OLSR	Optimized link state routing protocol
OSI	The Open Systems Interconnection
OSPF	Open Shortest Path First
PDR	Packet Delivery Ratio
PHY	Physical Layer
RAM	Random Access Memory

RIP	Routing Information Protocol
ROM	Read Only Memory
RPL	IPv6 Routing Protocol for Low-Power and Lossy Networks
RX	Receive
SF	Scheduling Function
TCP	Transmission Control Protocol
TSCH	Time Slotted Channel Hopping
TX	Transmit
UDP	User Datagram Protocol
URI	Uniform Resource Identifier
WSN	Wireless Sensor Networks
6LowPAN	IPv6 over Low power Wireless Personal Area Networks
6P	6TiSCH Operation Sublayer Protocol
6TiSCH	IPv6 over the TSCH mode of IEEE 802.15.4e
6top	6TiSCH Operation Sublayer

1. GENEL BİLGİLER

1.1. Giriş

Nesnelerin İnterneti (IoT), günümüzde giderek daha fazla önem kazanan ve teknolojinin evrimi ile birlikte hayatımıza yoğun bir şekilde giren bir kavramdır. Temel olarak, IoT, fiziksel nesnelerin internet aracılığıyla birbirleriyle iletişim kurabildiği ve veri alışverişinde bulunabildiği bir sistemdir. Bu nesneler, sensörler, cihazlar, araçlar, ev aletleri, endüstriyel ekipmanlar gibi geniş bir yelpazede bulunabilirler. Bu nesneler, entegre edilmiş sensörler ve bağlantı yetenekleri sayesinde çevrelerindeki verileri toplayabilir, işleyebilir ve paylaşabilirler.

IoT teknolojisi, nesnelerin internete bağlanmasını ve bir ağ üzerinden iletişim kurmasını sağlayarak, gerçek dünyadaki nesnelerin dijital dünyayla etkileşimini mümkün kılar. Sensörler ve akıllı cihazlar sayesinde, çevrelerindeki değişiklikleri algılayabilir ve bu verileri merkezi bir sistem veya bulut tabanlı platform üzerinden analiz edebilirler. Bu da daha akıllı, verimli ve etkili sistemlerin oluşturulmasını sağlar.

IoT'nin potansiyeli oldukça geniştir. Endüstriyel sektörde üretim süreçlerini optimize etmek, tarım alanında verimliliği artırmak, şehirlerde akıllı ulaşım sistemleri oluşturmak, sağlık sektöründe uzaktan izleme ve tedavi imkanları sunmak gibi birçok farklı alanda kullanılabilir. Ancak, bu büyük faydaların yanı sıra, veri güvenliği, gizlilik ve ağ uyumluluğu gibi zorluklar da beraberinde gelmektedir.

Sonuç olarak, Nesnelerin İnterneti kavramı, fiziksel dünyayla dijital dünyayı daha da entegre eden ve farklı sektörlerde büyük dönüşümlere yol açabilecek bir teknoloji alanını ifade eder. Bu teknoloji, nesneler arasındaki iletişimi artırarak, veri toplama ve analiz yeteneklerini optimize ederek, günlük yaşamımızı daha akıllı, verimli ve bağlantılı hale getirme potansiyeline sahiptir.

İnternet of Things (IoT) ve Kablosuz Sensör Ağı (WSN - Wireless Sensor Network), kablosuz iletişim teknolojilerini kullanarak nesnelerin (örneğin, cihazlar, sensörler, ve ekipmanlar) birbirleriyle veya internet üzerinden iletişim kurmalarını sağlayan teknolojik kavramlardır. IoT, genellikle bu nesnelerin veri toplaması, analiz etmesi ve bu verileri uzaktan izlemesi veya kontrol etmesi amacıyla kullanılırken, WSN, bu nesnelerin çevresel verileri toplamak için düşük maliyetli ve enerji verimli bir ağ oluşturmasına yardımcı olur.

Yani, IoT genellikle verilerin merkezi bir konumda toplandığı ve işlendiği bir yapıyı ifade ederken, WSN ise bu verilerin yerinde toplandığı ve iletim sağladığı kablosuz ağıın bileşenlerini temsil eder. Birlikte kullanıldıklarında, IoT cihazları WSN ağları üzerinden veri iletebilir ve bu sayede uzaktan izleme, otomasyon ve çeşitli uygulamalarda daha geniş kapsamlı çözümler sağlayabilir.

IoT (Internet of Things) cihazları genellikle IPv6 (Internet Protocol version 6) kullanarak internet üzerinden iletişim kurarlar. RPL (Routing Protocol for Low-Power and Lossy Networks), IPv6 tabanlı kablosuz sensör ağlarının (WSN - Wireless Sensor Networks) yönlendirme işlemlerini optimize etmek için kullanılan bir yönlendirme protokolüdür. RPL, düşük enerji tüketimi ve ağdaki kaynakların verimli kullanımı gibi IoT cihazları için önemli olan özellikleri destekler. Bu nedenle, IoT cihazları, IPv6 ile RPL'yi kullanarak kablosuz sensör ağları içinde veri iletebilir ve enerji tasarrufu sağlayarak uzun süreli çalışabilirler. Bu protokoller, IoT cihazlarının ve WSN'lerin daha etkili bir şekilde çalışmasına katkı sağlar.

IoT (Internet of Things) cihazları, genellikle sınırlı kaynaklara sahip kablosuz sensör ağlarında çalışırken, 6LoWPAN (IPv6 over Low-Power Wireless Personal Area Networks), bu tür cihazların IPv6 ile uyumlu hale gelmelerini sağlayan bir iletişim protokolüdür. 6LoWPAN, IPv6'nın geniş adresleme kapasitesini ve ağ katmanı özelliklerini, düşük güç tüketimi ve düşük bant genişliği ile sınırlı kablosuz ağlara entegre eder. Bu sayede, IoT cihazları IPv6 adreslemesi ve internet üzerinden veri iletimi gibi standart protokoller kullanarak iletişim kurabilirler. 6LoWPAN, IoT'nin küresel çapta iletişim sağlamasına yardımcı olurken enerji verimliliği ve düşük maliyet avantajları sunar.

1.2. Motivasyon

Günümüzde Nesnelerin İnterneti (IoT) teknolojisinin hızla yaygınlaşmasıyla birlikte, kablosuz sensör ağlarının (WSN) kurulumu ve yönetimi giderek daha kritik hale gelmiştir. Bu ağlar, kısıtlı enerji kaynaklarına ve iletim menziline sahip düğümlerden oluşurken, özellikle 6TiSCH (IPv6 over the TSCH mode of IEEE 802.15.4e) ağ ortamında kullanılan sensörler, daha düşük güç tüketimi ve daha iyi ağ verimliliği hedefleriyle tasarlanmıştır. Ancak, aynı coğrafi bölgede bulunan birden fazla IoT ağı söz konusu olduğunda, düğümler komşu düğümleri ve kök düğümleri bilmeden yerleşirler. Bu durum, ağın verimliliğini ve kararlılığını olumsuz etkileyebilir. Ayrıca, paket kaybı gibi sorunlar düğümlerin kök düğüm seçimlerini değiştirmelerine neden olabilir, bu da ağın performansını düşürebilir.

Bu bağlamda, mevcut RPL (Routing Protocol for Low-Power and Lossy Networks) protokolü IETF 6TiSCH bünyesinde yer alır ve ağdaki yönlendirme işlemlerini düzenler. Ancak, düğümlerin dinamik bir şekilde yönetimi ve optimizasyonu için yeni bir çözüme ihtiyaç vardır. Bu çalışmanın temel motivasyonu, düğümlerin daha etkili bir şekilde yönetilmesini ve ağın verimliliğinin artırılmasını sağlamak için RPL protokolünün temel altyapısını kullanarak yeni bir algoritma geliştirmektir. Bu algoritma, düğümlere farklı ağ önekleri ve kanal atamaları sağlayarak ağ içinde daha iyi bir dengeleme ve optimizasyon sağlamayı amaçlamaktadır.

Bu çalışma, aynı coğrafi ortamda bulunan komşu ağlarda düğümler arasındaki kök düğüm seçimi, düğüm dengeleme ve mesaj iletim miktarlarında elde edilen iyileştirmeleri göstermektedir. Bu iyileştirmeler, ağın daha kararlı, performanslı ve verimli bir şekilde çalışmasını sağlamak adına önemli bir adım olarak öne çıkmaktadır. Bu nedenle, IoT ağlarının daha etkili yönetimi ve optimizasyonu konusundaki bu çalışma, gelecekteki kablosuz sensör ağı uygulamaları için önemli bir katkı sağlamaktadır.

1.3. Tezin Yapısı

Bu çalışmada; komşu 6TiSCH ağların dengelenmesi ve ağı yeni katılan düğümlerin düğümlerin dengeli bir şekilde komşu ağlara dahil olabilmesi için yeni bir takım algoritmalar ve yöntemler önerilmiştir. Bu bildiride başlangıçta I.Bölüm’de tezin yapısına ilişkin genel bilgiler verilmiştir. II.Bölüm’de tezin genel konusu olan Nesnelerin İnterneti kavramına ilişkin temel kavramlar üzerinde durulmuştur. 6TiSCH ve tezin uygulama ve deneylerinin yapıldığı Contiki İşletim Sistemi ve Cooja Simülatöründen bahsedilmiştir. III.Bölüm’de ise Kablolu ve Kablosuz Ağ Yönlendirme Protokolleri ve RPL yapısı, yönlendirme ve amaç fonksiyonları irdelenmiştir. IV.Bölüm’de daha önce tez konusu ile ilgili yapılan halihazırdaki çalışmalar araştırılarak Literatür İncelenmesi yapılmış, mevcut tez ile karşılaştırılarak analizler paylaşılmıştır. V.Bölüm’de yapılan çalışma anlatılarak, geliştirilen algoritmalar ve sonuçları kapsam ve içerikleri detaylandırılmıştır. Algoritmik çalışmanın sonuçları, VI.Bölümde Contiki-NG (URL-3)’nin Cooja simülasyon ortamı (URL-4) ile derlenmiş ve analiz yapılmıştır. VII.Bölüm ve VIII.Bölüm’de ise uygulama ve test sonuçları sunulurken, gelecek çalışmalara ışık tutulmuştur.

2. NESNELERİN İNTERNETİ

İnternetin geleceği olarak tanımlanan IoT (internet of things) “Nesnelerin İnterneti”, günlük hayatta kullanılan nesnelerin, internet sayesinde diğer nesneler ile veri alışverişi yapabilmesini ve nesnelerin birbiriyle tam olarak senkronizasyon halinde olmalarını sağlayan bir teknolojidir.

Bu teknoloji ortamlardan verileri toplamayı, göndermeyi ve üzerinde işlem yapmayı sağlayan web özellikli akıllı cihazlardan oluşur. Nesnelerin interneti cihazları, toplanan verileri bir ağ geçidi ile aktarır, verilerin analiz edilebileceği bir bulut sistemine veya farklı bir uç cihaza bağlayarak paylaşır.

IoT fiziksel nesnelerin (cihazlar, sensörler, ev aletleri vb.) internet bağlantısıyla birbirleriyle ve/veya insanlarla iletişim kurabilme yeteneği sağlayan bir teknoloji ve kavramdır. Bu sayede, bu nesneler veri toplar, paylaşır ve uzaktan izlenebilir veya kontrol edilebilir. IoT, otomasyon, veri analizi, enerji verimliliği ve daha pek çok uygulama alanında kullanılarak yaşamımızı daha akıllı ve bağlantılı hale getirir.



Şekil 2. Nesnelerin İnterneti

2.1. Genel Bakış ve Kullanılan Teknolojiler

Günümüzde, birçok akıllı cihaz ve nesne, içerdikleri sensörler sayesinde çevrelerindeki gerçek zamanlı bilgileri algılamaktadır. Bu eş zamanlı veriler, dizüstü bilgisayarlar, akıllı otomobiller, akıllı ev cihazları gibi farklı nesnelerin birbirine bağlandığı ve bu verilerin analiz edilmesine imkan tanıyan bir ağ üzerinden iletişim kurduğu Nesnelerin İnterneti (IoT) kavramının temelini oluşturmaktadır. (Sharma vd., 2018).

Open Systems Interconnection (OSI) referans modeli, bilgisayar ağlarının tasarımını ve işleyişini anlamak için temel bir çerçeve sağlayan kapsamlı bir konsepttir. Bu model, ağ iletişimini yedi farklı katmana böler, her biri belirli bir işlevi yerine getiren bileşenleri içerir. OSI modeli, ağ protokollerinin ve iletişim teknolojilerinin geliştirilmesi ve uygulanmasını standartlaştırmayı amaçlar.

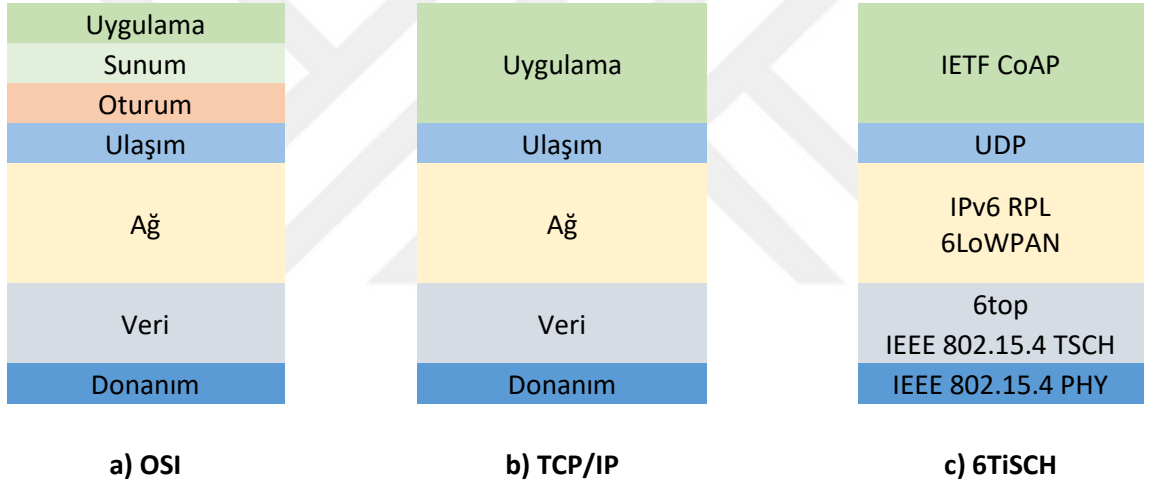
Modelin alt katmanları olan Fiziksel (Physical) ve Veri Bağlantısı (Data Link) katmanları, sırasıyla verilerin elektrik sinyallerine dönüştüğü fiziksel ortamı ve verilerin güvenilir ve hata toleranslı bir şekilde iletimini sağlar. Ardından Ağ (Network) katmanı, veri paketlerinin hedefe yönlendirilmesini ve ağ topolojisinin yönetimini gerçekleştirir. Bu katman, IP adreslemesi ve yönlendirme protokolleri ile öne çıkar.

Taşıma (Transport) katmanı, kaynak ve hedef cihazlar arasında güvenilir bir veri akışı sağlayarak veri bütünlüğünü ve akış kontrolünü yönetir. Uygulama (Application) katmanı, kullanıcıların ağdaki hizmetlere erişimini temsil eder; e-posta, web tarayıcıları, dosya paylaşımı gibi uygulamalar bu katmanda yer alır.

Bu katmanlar, ağ iletişiminin farklı yönlerini ve işlevlerini ayrı ayrı ele alarak ağ protokollerinin geliştirilmesini ve uyumluluğunu kolaylaştırır. OSI modeli, ağ mühendislerinin ve tasarımcılarının daha iyi bir ağ yapısı oluşturmalarına yardımcı olurken, farklı cihazlar ve ağ bileşenleri arasındaki etkileşimleri daha iyi anlamalarını sağlar.

OSI modeli, bilgisayar ağları ve iletişim sistemlerinin çalışma prensiplerini ve katmanlarını açıklayan bir standart referans modelidir. Bu model, ağ iletişiminin her aşamasını yedi katmana böler: fiziksel, veri bağlantısı, ağ, taşıma, oturum, sunum ve uygulama. Her katman, özel bir işlevi yerine getirir ve iletişim sürecini modüler hale getirir. OSI modeli, farklı cihazların ve protokollerin uyumlu bir şekilde birbirleriyle iletişim kurmasını kolaylaştıran bir temel referans noktası sağlar.

OSI modeli ve IoT (Internet of Things) kavramı, bilgisayar ağları ve iletişim sistemleri alanlarında farklı ama birbirini tamamlayan yönleri olan iki önemli konsepttir. OSI modeli, bilgisayar ağlarının çalışma prensiplerini ve katmanlarını tanımlar, bu sayede farklı cihazlar ve iletişim protokolleri arasındaki uyumu sağlar. IoT ise fiziksel nesnelerin internete bağlanmasını ifade eder ve bu nesnelerin veri toplaması, analiz etmesi ve uzaktan yönetilebilmesini mümkün kılar. IoT, OSI modelindeki katmanların özellikle fiziksel, veri bağlantısı, ağ ve uygulama katmanları ile etkileşim içinde bulunur. IoT cihazları, bu katmanlarda iletişim kurarak veri toplar ve işler, ardından bu verileri uygulama katmanı aracılığıyla kullanıcı veya diğer sistemlerle paylaşır. Bu sayede, OSI modeli ve IoT, modern iletişim altyapısının temelini oluşturarak cihazların birbirleriyle ve insanlarla bağlantılı hale gelmesine katkı sağlar.



Şekil 2.1. IoT 6TiSCH Protokol Yığını

2.1.1. IEEE 802.15.4 PHY

IEEE 802.15.4, kablosuz iletişim sistemlerinde düşük güç tüketimi ve düşük veri hızları gerektiren uygulamalar için tasarlanmış bir fiziksel (PHY) katman standardıdır. Bu standardın ana amacı, sensör ağları, endüstriyel otomasyon sistemleri, tıbbi cihazlar ve akıllı evler gibi düşük güç tüketimi gereksinimleri olan uygulamalar için düşük maliyetli ve enerji verimli bir kablosuz iletişim çözümü sunmaktır.

IEEE 802.15.4 PHY, düşük veri hızlarında çalışmak üzere optimize edilmiştir ve genellikle 250 kbps veya daha düşük veri iletim hızlarını destekler. Bu düşük veri hızları, sensör verileri gibi düşük bant genişliği gereksinimlerine sahip uygulamalar için uygundur.

Ayrıca, düşük güç tüketimi ve enerji verimliliği, cihazların uzun pil ömrüne sahip olmasını sağlar.

IEEE 802.15.4 PHY, düşük maliyetli donanım gereksinimlerine ve düşük karmaşıklığa sahip olma avantajına sahiptir. Bu nedenle, büyük ölçüde kablosuz sensör ağları ve IoT (Nesnelerin İnterneti) uygulamalarında kullanılır. PHY katmanı, IEEE 802.15.4 MAC (Medium Access Control) katmanı ile bir araya gelerek cihazların kablosuz ağlarda iletişim kurmasını sağlar.

Sonuç olarak, IEEE 802.15.4 PHY, düşük güç tüketimi, düşük maliyet ve düşük veri hızlarına ihtiyaç duyan kablosuz iletişim uygulamaları için önemli bir standarttır ve genellikle sensör ağları gibi çeşitli alanlarda kullanılır.

IEEE 802.15.4, düşük güç tüketimi gereksinimleri olan kablosuz cihazların iletişiminde desteklemek için tasarlanmış bir IEEE 802 kablosuz iletişim standardıdır. Bu standardın ana amacı, sensör ağları, endüstriyel otomasyon sistemleri, tıbbi cihazlar, akıllı ev uygulamaları ve diğer düşük veri hızı ve düşük güç tüketimi gereksinimlerine sahip IoT (Nesnelerin İnterneti) uygulamaları için uygun bir iletişim çözümü sunmaktır.

IEEE 802.15.4 standardı, iki temel katmandan oluşur:

Fiziksel Katman (PHY): IEEE 802.15.4 PHY, düşük veri hızları için optimize edilmiş bir kablosuz iletişim standardıdır. Genellikle 250 kbps veya daha düşük veri iletim hızlarını destekler. Bu düşük veri hızları, sensör verileri gibi düşük bant genişliği gereksinimlerine sahip uygulamalar için uygundur. Ayrıca, düşük güç tüketimi ve enerji verimliliği sağlamak için tasarlanmıştır.

Orta Erişim Kontrol Katmanı (MAC): IEEE 802.15.4 MAC katmanı, kablosuz cihazların veri iletimini ve alımını yönetir. Bu katman, veri paketlerinin zamanlamasını, güvenliğini ve yönlendirmesini düzenler. Ayrıca, cihazların ağa katılma, ağdan ayrılma ve enerji tasarrufu gibi işlevleri de kontrol eder.

IEEE 802.15.4, düşük maliyetli donanım gereksinimlerine sahip olduğu için yaygın olarak kullanılır ve genellikle pil ömrü kritik olan kablosuz cihazlar için tercih edilir. Ayrıca, düşük güç tüketimi, iletişim mesafesini artırmak için kullanılan enerji tasarrufu modları ile birleştirilerek, uzun pil ömrüne sahip cihazlar için ideal bir seçenek sunar.

Bu standardın yaygın kullanım alanları, akıllı ev otomasyonu, akıllı şebekeler, tıbbi cihazlar, endüstriyel otomasyon, tarım izleme ve sensör ağları gibi IoT uygulamalarını içerir. IEEE 802.15.4, çeşitli frekans bantları ve iletişim modları için çeşitli sürümleri destekler, bu da farklı uygulamalara uyum sağlama esnekliği sunar.

2.1.2. IEEE 802.15.4e (TSCH)

IEEE 802.15.4e, IEEE 802.15.4 standardının bir uzantısıdır ve Zaman Eşzamanlı Çoklu Erişim (Time-Slotted Channel Hopping - TSCH) adı verilen iletişim yöntemini tanıtarak kablosuz ağların daha güvenilir ve zaman açısından hassas hale gelmesini amaçlar. TSCH, belirli bir zaman çizelgesine göre düzenlenmiş veri iletim zaman yuvaları kullanarak iletişimi koordine eder. IEEE 802.15.4e (TSCH) temel olarak:

Zaman Eşzamanlı Çoklu Erişim (TSCH) Kavramı: TSCH, kablosuz ağlarda güvenilirlik, düşük gecikme ve enerji verimliliği sağlamak için bir dizi iletişim zaman yuvası (slot) kullanır. Bu yuvalar, cihazların iletişimlerini belli bir zaman dilimine yaymalarına olanak tanır ve çakışmaları önler. TSCH ayrıca, ağ cihazlarının düşük güç tüketimini korurken enerji tasarrufu sağlar. Zaman yuvaları, cihazların uyku modlarına girmelerine ve uyandıklarında iletişim yapmalarına olanak tanır.

Güvenilir ve Zaman Hassas İletişim: IEEE 802.15.4e (TSCH), özellikle endüstriyel otomasyon, akıllı şebeke ve sağlık izleme gibi uygulamalarda güvenilir ve zaman açısından hassas iletişim gereksinimlerini karşılar. İletişim yuvaları ve zaman çizelgesi, veri iletimini tahmin edilebilir ve güvenilir hale getirir, böylece cihazlar zamanında ve düşük gecikme ile veri gönderebilirler. Bu, endüstriyel otomasyon sistemlerinin ve gerçek zamanlı kontrol uygulamalarının kablosuz ağlarını desteklemek için önemlidir.

Uygulama Alanları: IEEE 802.15.4e (TSCH), enerji verimliliği ve güvenilirlik gerektiren birçok uygulama alanında kullanılır. Özellikle endüstriyel otomasyon, akıllı şebeke yönetimi, akıllı şehirler, lojistik izleme, akıllı tarım ve sağlık izleme sistemleri gibi alanlarda tercih edilir. TSCH, bu tür uygulamalar için iletişim güvenilirliğini artırırken enerji tüketimini minimize etmeye yardımcı olur ve böylece cihazların pil ömrünü uzatır.

IEEE 802.15.4e (TSCH), endüstriyel IoT ve benzeri karmaşık kablosuz ağ uygulamaları için önemli bir iletişim standardıdır, çünkü zaman açısından hassas ve güvenilir veri iletimi sağlama yeteneğiyle fark yaratır.

2.1.3. TiSCH Çizelgesi

TiSCH, bir zaman çizelgesine dayalı bir iletişim yöntemidir.

- ❖ Yatay eksenle zaman temsil edilir. Zamanın belirli bir süresi boyunca bir iletişim çerçevesi oluşturulur.

- ❖ Dikey ekseninde cihazlar veya düğümler temsil edilir. Bu düğümler, ağa katılan cihazları veya iletişim yapan cihazları ifade eder.
- ❖ Her cihaz veya düğümün bir dizi zaman yuvası (slot) ile ilişkilendirilmektedir. Bu zaman yuvaları, düğümlerin veri iletimlerini ve alımlarını zamanlayan bölümlerdir.
- ❖ TiSCH, bu zaman yuvalarını belirli bir düzen içinde kullanır. Örneğin, bir cihaz belirli bir zaman yuvasında veri gönderirken, diğerleri sessiz kalabilir veya başka bir yuvada iletişim yapabilir.
- ❖ Zaman yuvaları arasındaki geçişler, zaman açısından hassas bir şekilde düzenlenir. Bu, çakışmaları ve gecikmeleri minimize eder.

Time Slotted Channel Hopping (TiSCH) protokolü zaman çizelgesini zaman dilimlerine böler. Ağdaki iletişim bu şekilde düzenlenir.

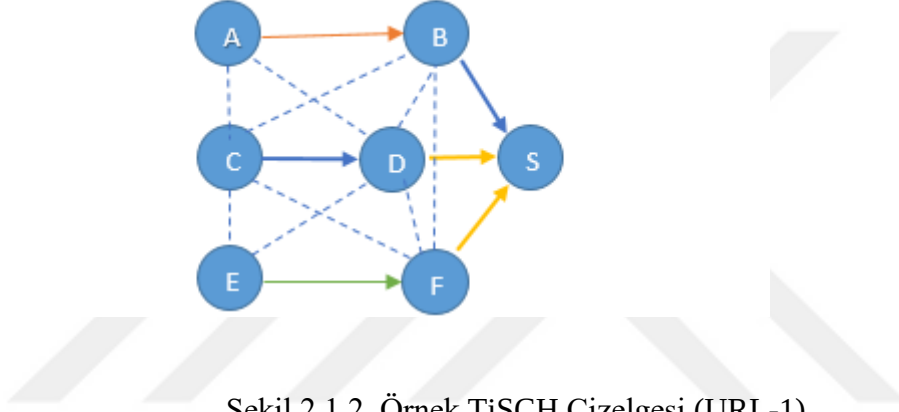
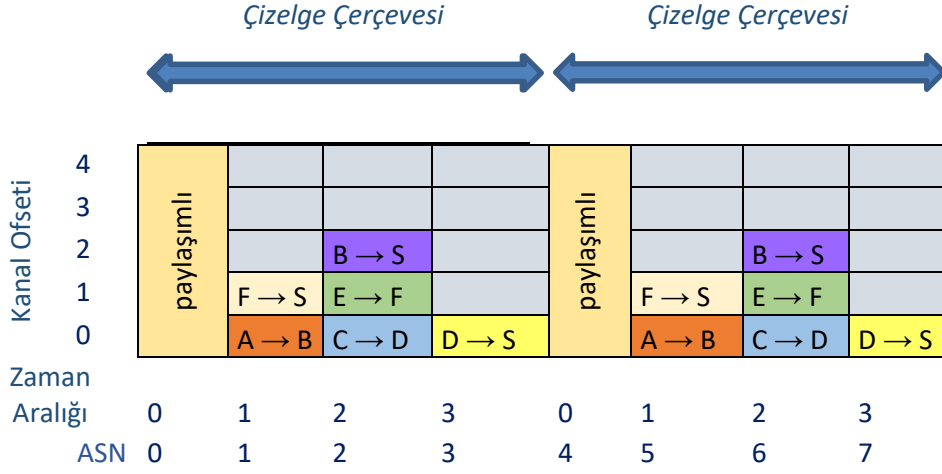
IEEE 802.15.4e standardında, dört farklı hücre tipi tanımlanmıştır: İletim (TX), Alım (RX), Paylaşımlı (Shared) ve Zaman Tutma (Timekeeping) (Liu vd.,2019). Bu hücre tipleri, düğümlerin farklı iletişim ve işlevsel gereksinimlerini karşılamak için kullanılır.

Bir düğüm, veri ileteceği bir paketi varsa "TX" moduna geçer ve aksi takdirde gelen veriyi dinler ("RX" moduna geçer). Bir düğüm, IEEE 802.15.4'e uygun bir bağlantı katmanı onayı gerektiren bir iletim yaparsa ve bu onayı alamazsa, muhtemel çakışmaları çözmek için bir çarpışma çözümü mekanizması kullanır ("Shared" biti ayarlanır).

Ağa katılan bir düğüm, ilk komşusuyla zaman senkronizasyonunu korumak için bir hücreyi kullanabilir ("Timekeeping" biti ayarlanır). Bu, ağdaki tüm düğümlerin zaman içinde senkronize olmasını ve verilerin daha verimli bir şekilde iletilmesini sağlar.

Bu şekilde, IEEE 802.15.4e standardı, farklı iletişim senaryolarına ve düğüm gereksinimlerine uygun hücre tiplerini tanımlar, böylece kablosuz ağlar daha iyi performans ve güvenilirlik sağlayabilir.(Vilajosana vd., 2022).

TiSCH, Şekil 2.1.2'de görüldüğü zaman dilimi ve frekans bantlarının matris şeklinde gösterildiği bir çizelge içinde resmedilebilir. Zaman Paylaşımlı / Kanal Atlamalı (Time Slotted / Channel Hopping - TSCH) ortam erişimi kullanıldığında, aynı zaman diliminde farklı kanal ofsetleri kullanılarak iletim gerçekleştirilebilir. Bu sayede paylaşılan ve özel olarak tahsis edilen hücrelerin efektif bir şekilde kullanılması mümkün olur.



Şekil 2.1.2. Örnek TiSCH Çizelgesi (URL-1)

11-26 arasında 16 kanal ile 2.4 GHz frekansında çalışan bu sistemde zaman aralıkları ASN denilen mutlak sıra numarası ile belirlenir. ASN ifadesi, ağ kurulumundan itibaren zaman aralıklarının sayısını hesaplayan bir değişken olarak karşımıza çıkar.

$$ASN = k \cdot N_s + t_s \quad (2)$$

ASN değeri, bir düğümün çerçevesini ne zaman göndereceğine veya alacağına karar vermek için kullanılan bir zaman damgasını temsil eder. ASN, çizelge çerçevesi döngüsünün bir katı olan k ile çizelge çerçevesi boyutu N_s ve zamanlayıcı t_s arasındaki ilişkiyi ifade eder.

Frekans (f), ASN, kanal ofseti ($chof$) ve mod operatörünün bir kombinasyonudur. Fonksiyon f , $chof$ 'u bir frekansa dönüştürür. Çok kanallı iletişim mekanizması, farklı kanal ofsetleri kullanarak aynı anda birden fazla iletişimi mümkün kılar. Bu mekanizma, aynı ağdaki iki düğüm arasındaki bağlantılar için farklı zaman aralıklarında farklı frekanslarla kanal atlama yapabilir.

Kanal ofset sayısı N_c ile ifade edilir ve her zaman diliminde iletim için $N_c \leq 16$ 'ya kadar farklı fiziksel frekanslar mevcuttur. TSCH, iletişimleri programlamak için bir bağlantı matrisi (hücreler) sağlar, burada her bağlantı bir çift $[ts, chof]$ ile tanımlanabilir. Bu, mevcut kanalların tümünün iletişim için kullanılmasını ve girişimin olumsuz etkisinin azaltılmasını sağlar.

Bu kapsamda TiSCH çizelgesi, belirli bir zaman dilimindeki düğümlerin iletişimlerini düzenlemek ve koordine etmek için kullanılır. Her düğümün belirli zaman yuvalarını kullanarak iletişim yapması, güvenilir ve zaman açısından hassas bir iletişim sağlar. Bu sayede, karmaşık endüstriyel IoT uygulamaları gibi gereksinimleri olan uygulamalarda kullanılabilir.

2.1.4. 6TiSCH

IETF (Internet Engineering Task Force), TSCH (Time-Slotted Channel Hopping) ve IPv6 protokollerini bir araya getirerek kablosuz ağlar için yüksek verimlilik ve minimum gecikme süresi sağlamayı amaçlayan 6TiSCH (IPv6 over the TSCH mode of IEEE 802.15.4e) adlı bir çalışma grubu oluşturmuştur.

6TiSCH'in ana hedefi, endüstriyel kablosuz sensör ağları gibi uygulamalarda kullanılan WirelessHART ve ISA100 gibi teknolojiler için uygun bir çözüm sunmaktır. Bu çalışma grubu, standartların ve cihazların çeşitliliği göz önüne alındığında, farklı cihazların sorunsuz bir şekilde bir arada çalışabilmesini sağlamak için temel bir 6TiSCH konfigürasyonunu sunarak uyumluluk seviyesini en aza indirmeye amacını taşır.

6TiSCH, IEEE 802.15.4e standardının TSCH (Time-Slotted Channel Hopping) modunu temel alan bir kablosuz ağ protokolüdür. Bu protokol, özellikle endüstriyel IoT (Nesnelerin İnterneti) ve akıllı şebeke uygulamaları gibi güvenilir ve zaman açısından hassas kablosuz iletişim gereksinimlerini karşılamak için tasarlanmıştır. (URL-1).

TSCH Temeli: 6TiSCH, IEEE 802.15.4e standardının TSCH modunu kullanır. TSCH, iletişimi zaman yuvalarına bölen ve çakışmaları önlemek, güvenilir iletişim sağlamak ve enerji tasarrufu yapmak için düzenlenmiş bir zaman çizelgesi kullanır. Bu sayede cihazlar zaman açısından hassas ve düşük güç tüketimi ile iletişim yapabilir.

IPv6 Desteği: 6TiSCH, IPv6 protokolünü temel alır. Bu, cihazların internet protokolü üzerinden iletişim kurabilmesini sağlar. IPv6, geniş çaplı kablosuz ağlar için

ölçeklenebilirlik sağlayan bir protokoldür ve bu nedenle 6TiSCH, büyük ağlar için uygun bir seçenek sunar.

Endüstriyel ve Akıllı Şebeke Uygulamaları: 6TiSCH, endüstriyel otomasyon, akıllı şebeke yönetimi ve benzeri uygulamalar için idealdir. Bu tür uygulamalarda iletişim güvenilirliği, düşük gecikme süreleri ve enerji tasarrufu kritik öneme sahiptir. 6TiSCH, bu gereksinimleri karşılamak için zaman senkronizasyonu ve düzenli zaman yuvası tahsisi gibi özellikler sunar.

Açık Standart: 6TiSCH, IEEE 802.15.4e ve IPv6 gibi açık standartlara dayanır. Bu, farklı üreticilerin ve geliştiricilerin uyumlu cihazlar ve yazılım oluşturmalarını kolaylaştırır ve pazarda çeşitliliği artırır. Açık standartlar, endüstriyel IoT ve akıllı şebeke uygulamalarının büyümesine katkıda bulunur.

Sonuç olarak, 6TiSCH, endüstriyel IoT ve akıllı şebeke gibi karmaşık ve zaman açısından hassas kablosuz ağ uygulamaları için tasarlanmış güçlü bir protokoldür. TSCH ve IPv6 ile uyumlu olması, güvenilirlik, ölçeklenebilirlik ve açık standartlara dayanma özellikleri ile bu tür uygulamalar için uygun bir iletişim çözümü sunar.

2.1.5. 6TiSCH Operation Sublayer (6top)

6TiSCH Operation Sublayer (6top), kablosuz ağlarda IEEE 802.15.4e TSCH (Time-Slotted Channel Hopping) modunu temel alan, düşük güç tüketimi ve yüksek güvenilirlik gereksinimleri olan endüstriyel IoT (Nesnelerin İnterneti) ve benzeri uygulamalar için tasarlanmış bir protokol katmanıdır. 6top, 6TiSCH çalışma grubunun bir parçası olarak geliştirilmiştir ve ağdaki düğümlerin veri iletimini düzenlemek ve koordine etmek için kullanılır.

Zaman Yuvaları ve Rütbeler: 6top, ağdaki zaman yuvalarını ve düğümleri yönetmek için kullanılır. Bu, iletişimin zaman içinde nasıl organize edileceğini ve düğümlerin hangi zaman yuvalarında iletişim yapacaklarını düzenler. Ayrıca, düğümlerin ağdaki "rütbesini" belirler, yani iletişimdeki önceliklerini ve düğümlerin ne zaman iletişim yapabileceğini belirler.

Veri İletimi ve Enerji Tasarrufu: 6top, düşük güç tüketimi gereksinimlerine sahip kablosuz cihazlar için enerji tasarruflu veri iletimini destekler. Zaman yuvaları sayesinde cihazlar zamanında iletişim yapabilir ve ardından enerji tasarruflu uyku modlarına geçebilirler. Bu, cihazların pil ömrünü uzatırken güvenilir iletişimi sağlar.

Uyumluluk ve Esneklik: 6top, farklı üreticilerden gelen cihazların ve farklı ağ yapılarının bir arada çalışabilmesini sağlamak için esnek bir yapı sunar. Bu, endüstriyel IoT uygulamaları gibi heterojen ağ ortamlarında uyumluluğun korunmasına yardımcı olur. Ayrıca, 6top, ağın değişen ihtiyaçlarına uyum sağlamak için konfigürasyon ve yönetim esnekliği sunar.

Sonuç olarak, 6TiSCH Operation Sublayer (6top), endüstriyel IoT ve benzeri uygulamalar için düşük güç tüketimi, yüksek güvenilirlik ve esneklik gereksinimlerini karşılamak üzere tasarlanmış bir protokol katmanıdır. Zaman yuvaları ve rütbeleme gibi özellikler, ağın verimli bir şekilde çalışmasını ve düşük güç tüketimini sağlar. Bu, endüstriyel kablosuz sensör ağları ve akıllı şebeke uygulamaları gibi kritik gereksinimlere sahip uygulamalarda önemli bir rol oynar.

2.1.6. 6LoWPAN

6LoWPAN (IPv6 over Low-Power Wireless Personal Area Network), düşük güç tüketimi, düşük veri hızları ve kısıtlı kaynaklara sahip kablosuz cihazlar için tasarlanmış bir ağ protokolüdür. Bu protokol, özellikle Nesnelerin İnterneti (IoT) uygulamalarında kullanılmak üzere geliştirilmiştir ve kablosuz sensör ağları gibi düşük güç tüketimi gerektiren senaryolarda yaygın olarak kullanılır.

IPv6 Desteği: 6LoWPAN, İnternet Protokolü sürüm 6 (IPv6) ile uyumlu bir şekilde çalışır. Bu, kablosuz cihazların doğrudan internete bağlanabilmesini sağlar. IPv6, geniş adresleme kapasitesi ve daha iyi güvenlik özellikleri sunar ve bu nedenle IoT cihazları için ideal bir seçenektir.

Veri Sıkıştırma: 6LoWPAN, düşük güç tüketimi gereksinimlerine sahip cihazlar için veri sıkıştırma teknikleri sunar. Bu, veri iletimini daha verimli hale getirir ve enerji tasarrufu sağlar. Ayrıca, sınırlı bant genişliği olan kablosuz ağlarda veri sıkıştırma, iletişim verimliliğini artırır.

Adresleme ve Yönlendirme: 6LoWPAN, düşük güç tüketimi gereksinimlerini karşılayacak şekilde IPv6 adresleme ve yönlendirme yöntemlerini optimize eder. Bu, kablosuz sensör ağları gibi cihazların düşük güç tüketimi ile çalışmasını sağlar ve ağın verimli bir şekilde yönlendirilmesine olanak tanır.

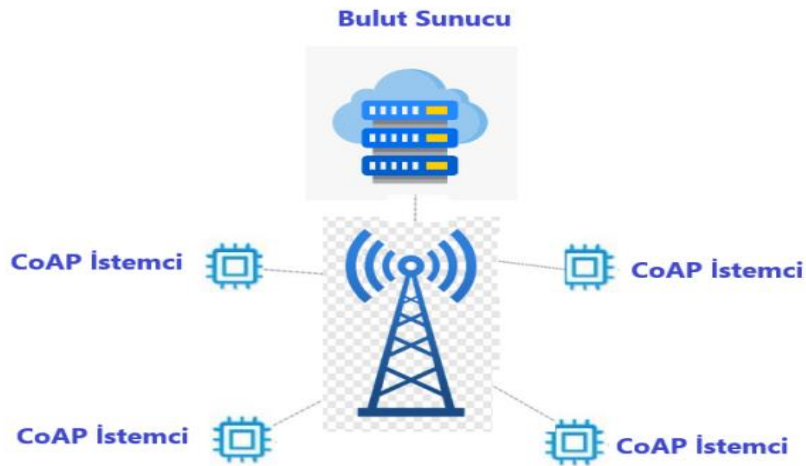
Uygulama Alanları: 6LoWPAN, enerji izleme, akıllı ev otomasyonu, akıllı şebeke yönetimi, tarım izleme ve daha birçok IoT uygulama alanında kullanılır. Bu protokol, düşük maliyetli ve enerji verimli kablosuz cihazların internetle bağlantısını kolaylaştırır ve IoT ekosistemini genişletir.

Sonuç olarak, 6LoWPAN, düşük güç tüketimi ve kısıtlı kaynaklara sahip kablosuz cihazlar için IPv6 tabanlı bir ağ protokolüdür. IoT uygulamaları için ideal bir çözüm sunar ve düşük güç tüketimi, veri sıkıştırma ve adresleme optimizasyonu gibi özellikleri ile enerji verimliliği sağlar. Bu, IoT dünyasında büyümeye devam eden kablosuz ağ teknolojilerinin önemli bir bileşenidir.

RPL, 6LoWPAN'ın yönlendirme işlevselliğini sağlayan bir üst katman protokolüdür ve bu protokolün detayları Bölüm 3.3'te kapsamlı bir şekilde açıklanmıştır.

2.1.7. CoAP

CoAP (Constrained Application Protocol), sınırlı kaynaklara sahip cihazlar arasında iletişim kurmayı kolaylaştıran hafif ve RESTful (Representational State Transfer) bir iletişim protokolüdür. CoAP, özellikle Internet of Things (IoT) ve M2M (Machine-to-Machine) uygulamaları gibi düşük güç tüketimi ve düşük bant genişliği gereksinimlerine sahip cihazlar için tasarlanmıştır.



Şekil 2.1.6. CoAP Yapısı

CoAP, HTTP'ye benzer bir yapıya sahiptir, ancak daha sade ve hafiftir. HTTP'nin yerine daha hızlı ve daha az kaynak tüketen bir iletişim mekanizması sunar. CoAP, REST mimarisini takip eder ve kaynakların benzersiz tanımlayıcılarını (URI'ler) kullanarak kaynaklara erişim sağlar. Bu, CoAP'ın web tabanlı uygulamalara benzer bir modeli desteklemesini sağlar.

CoAP, güvenli iletişim için DTLS (Datagram Transport Layer Security) ile birleştirilebilir ve böylece verilerin güvenli bir şekilde iletilmesini sağlar. Ayrıca, CoAP, mesajlarının hızlı ve güvenilir bir şekilde iletilmesini sağlamak için çeşitli iletişim profilleri ve yöntemleri sunar.

Sonuç olarak, CoAP, sınırlı kaynaklara sahip IoT cihazları için tasarlanmış hafif bir iletişim protokolüdür. Bu protokol, düşük güç tüketimi, düşük bant genişliği ve güvenlik gereksinimleri gözönüne alınarak geliştirilmiştir. CoAP, IoT dünyasında cihazlar arasında veri alışverişi için önemli bir iletişim standardıdır ve daha fazla cihazın internetle bağlantılı olmasını kolaylaştırır.

CoAP, beş temel katmandan oluşan bir iletişim protokolüdür. İşte CoAP'ın katmanlarını iki paragraf üzerinden açıklamak:

Uygulama Katmanı: CoAP'ın en üst katmanıdır ve kullanıcı tarafından erişilen katmandır. Bu katman, RESTful (Representational State Transfer) mimarisini takip eder ve kaynakların URI (Uniform Resource Identifier) ile temsil edildiği ve kaynaklara erişim için HTTP benzeri komutlar ve yöntemler kullandığı bir yapı sunar. Uygulama katmanı, istemci ve sunucu arasında kaynakların paylaşılması ve iletilmesi için gereken işlevselliği sağlar.

Yönlendirme Katmanı: CoAP'ın yönlendirme katmanı, CoAP mesajlarının ağ üzerinde iletilmesini ve yönlendirilmesini sağlar. Bu katman, kaynakların bulunduğu düğümler arasında iletişimi koordine eder ve hedef kaynağa mesajların güvenli bir şekilde iletilmesini sağlar. Yönlendirme katmanı, ağ topolojisinin oluşturulması, düğümlerin keşfedilmesi ve yönlendirme tablolarının güncellenmesi gibi görevleri yerine getirir. Bu sayede CoAP, düşük güç tüketimine sahip cihazlar arasında bile güvenilir iletişim sağlar.

CoAP'ın diğer katmanları arasında taşıma katmanı (UDP veya DTLS ile uygulanabilir), güvenlik katmanı (DTLS kullanarak güvenlik sağlar) ve ağ katmanı (IP adreslemesi ve yönlendirme) yer alır. Bu katmanlar, CoAP'ın düşük güç tüketimi, düşük bant genişliği ve güvenlik gereksinimlerini karşılamak için bir araya gelerek IoT cihazları arasında etkili iletişim sağlar.

2.1.8. LTE (Long-Term Evolution)

LTE kablosuz iletişimde bir standart olan 4G (4. Nesil) teknolojisinin bir parçasıdır. Geliştirilmiş veri hızları, daha iyi performans ve daha düşük gecikme süreleri sunarak önceki 3G teknolojisinin ötesinde bir deneyim sağlar. LTE, geniş bant veri hızlarını destekleyerek yüksek hızlı internet erişimini mobil cihazlar üzerinden mümkün kılar.

LTE'nin temel amacı, mobil geniş bant iletişimini iyileştirerek daha hızlı veri aktarımı, daha iyi ses kalitesi ve daha güçlü bağlantı stabilitesi sunmaktır. Bu standardın bir diğer önemli özelliği, IP tabanlı bir ağ mimarisi kullanmasıdır. Bu, daha esnek ve verimli bir ağ yapısı sağlar ve farklı cihaz türleri arasında sorunsuz bir iletişim sağlar.

LTE'nin benimsenmesi, mobil cihazlarda yüksek çözünürlüklü video akışları, hızlı internet gezintisi ve karmaşık uygulamalara daha iyi bir tepki süresi gibi avantajlarla sonuçlanmıştır. Ayrıca, LTE Advanced (LTE-A) gibi evrim geçirmiş versiyonları, daha yüksek veri hızları ve gelişmiş ağ kapasitesi gibi özelliklerle iletişim endüstrisinin sürekli gelişimini destekler.

LTE Arayüzleri ise şunlardır:

S1 Arayüzü: E-UTRAN (Evolved UMTS Terrestrial Radio Access Network) ve Evolved Packet Core (EPC) arasındaki ana iletişim arayüzüdür. S1, mobil cihazlar ve baz istasyonları ile ağın çekirdek kısmı arasındaki trafiği yönetir.

X1 Arayüzü: Evolved NodeB (eNB) cihazları arasındaki iletişimi sağlamak ve yönetmek için kullanılan bir arayüzdür. Mobil cihazların hareketini takip eder, hücreler arası trafiği optimize eder ve ağın genel kontrolünü sağlar.

Hücre sınırları içinde veya arasında elde değişiklikleri yönetir. Elde değişiklikleri, mobil cihazların bir hücreden diğerine geçişi veya farklı bir eNB'ye bağlanması durumlarını içerir.

Radyo kaynak yönetimini destekler, bu da hücreler arasında kaynak tahsisi, taşıma yönetimi ve radyo kaynak optimizasyonunu içerir. LTE ağının performansını izler ve iyileştirir, ağın genel verimliliğini artırır.

X1 arayüzü, mobil iletişim ağlarında önemli bir role sahiptir ve mobil cihazların hareketliliği ile ilgili işlevlerin düzenlenmesinde kilit bir rol oynar.

X2 Arayüzü: E-UTRAN'daki farklı baz istasyonları arasındaki iletişimi yönetir. Bu arayüz, hücreler arasında handover (eldeğiştirme) işlemlerini gerçekleştirir ve trafiği optimize eder.

S5/S8 Arayüzleri: Bu arayüzler, Evolved Packet Core (EPC) içindeki Mobility Management Entity (MME) ve Serving Gateway (SGW) arasındaki iletişimi sağlar. Mobil cihazların konum yönetimi ve taşıma işlemleri bu arayüzler üzerinden gerçekleşir.

S6a Arayüzü: E-UTRAN ve EPC arasında abonelik yönetimi ve yetkilendirme işlevlerini sağlayan bir arayüzdür.

S10 Arayüzü: MME ve SGW arasındaki iletişimi yönetir. Bu arayüz, mobil cihazların konumunu izler ve taşıma işlemlerini yönlendirir.

X1 arayüzü, LTE (Long-Term Evolution) ağlarında Evolved NodeB (eNB) cihazları arasındaki iletişimi sağlamak için kullanılan bir arabirimdir. Bu arayüz, mobil iletişim ağlarının çeşitli işlevlerini gerçekleştirmek üzere tasarlanmıştır.

X1 arayüzü incelendiğinde sağladığı özellikler aşağıdaki gibidir:

İletişim ve Kontrol: X1 arayüzü, eNB cihazları arasında trafiği yönetmek, mobil cihazların hareketini izlemek ve ağın genel kontrolünü sağlamak amacıyla kullanılır. Bu arayüz, cihazlar arasında iletişim kurarak, ağın güvenilir ve etkili bir şekilde çalışmasını sağlar.

Güncellemeler ve Elde Değişiklik: Mobil cihazların hareketi nedeniyle eNB'ler arasında elde değişikliklerin yönetimi için kullanılır. Bu, bir mobil cihazın bir hücre sınırlarını geçtiği veya farklı bir eNB'ye bağlandığı durumları içerir. X1 arayüzü, elde değişikliklerini düzenler ve cihazın hareketini izler.

Radyo Kaynak Yönetimi: Arayüz, radyo kaynaklarının etkili bir şekilde yönetilmesini sağlar. Bu, hücreler arasında kaynak tahsisi, hücreler arasında taşıma yönetimi ve başka türden radyo kaynak optimizasyonu gibi işlevleri içerir.

Performans İyileştirmesi: X1 arayüzü, ağın performansını izlemek ve iyileştirmek için kullanılır. Bu, hücreler arasındaki trafiği optimize etmek, hizmet kalitesini artırmak ve ağın genel verimliliğini sağlamak anlamına gelir.

Protokoller ve Standartlar: X1 arayüzü, 3GPP'nin belirlediği standartlara ve protokollere uyar. Bu standartlar, ağın uyumlu bir şekilde çalışmasını ve farklı cihazlar arasında etkileşimin düzenlenmesini sağlamak amacıyla belirlenir.

X1 arayüzü, mobil iletişim ağlarının karmaşıklığını yönetmek ve optimize etmek için önemlidir. Bu, kullanıcı deneyimini artırmak, ağın kapasitesini artırmak ve hücresel iletişim sistemlerini daha verimli hale getirmek için tasarlanmış bir arabirimdir.

2.1.9. SDN (Software Defined Networking)

SDN, geleneksel ağ yönetimi ve kontrol modellerini değiştiren bir ağ yönetim yaklaşımıdır. Geleneksel ağlarda, ağ cihazları (anahtarlar ve yönlendiriciler gibi) genellikle kendilerine özgü yönetim ve kontrol sistemlerine sahiptir ve ağ yönetimi karmaşık ve statik olabilir. SDN, ağ kontrolünü merkezi bir yazılım tabanlı kontrol düzlemine taşıyarak, ağın daha esnek, programlanabilir ve dinamik olmasını sağlar. (Marcio vd., 2018).

SDN'nin temel öğelerinden biri, ağ cihazlarının kontrol düzlemini ayıran bir mimari olan "kontrol ayırma"dır. Bu, ağdaki anahtar ve yönlendirici gibi cihazların, ağ trafiğini yönlendirmek yerine, merkezi bir kontrolör tarafından yönetildiği anlamına gelir. Kontrol düzlemi, ağın durumunu izler ve uygulanan politikalara dayanarak cihazlara yönlendirme kararları gönderir. Bu, ağ yöneticilerine daha büyük bir esneklik ve hız kazandırarak, ağ kaynaklarını daha etkili bir şekilde yönetmelerini sağlar.

SDN'nin bir diğer önemli özelliği, programlama araçları ve API'lar aracılığıyla ağ kaynaklarının dinamik olarak yönetilebilir olmasıdır. Bu, ağın hızlı bir şekilde uyarlanması ve değiştirilmesini sağlar. SDN, özellikle bulut bilişim ortamları, büyük veri uygulamaları ve akıllı ağlarda kullanılarak, ağ yönetimini daha verimli, ölçeklenebilir ve esnek hale getirir.

IoT ve SDN arasındaki ilişki, genellikle ağların daha etkili bir şekilde yönetilebilmesi ve IoT cihazlarıyla daha iyi entegre olabilmesi amacıyla kurulan bir bağlamda ortaya çıkmaktadır. IoT, birçok nesnenin internet üzerinden birbirine bağlandığı ve veri alışverişi yaptığı bir ekosistemdir. Milyonlarca sensör, cihaz ve nesnenin birbirine bağlanması, ağ altyapısından daha fazla esneklik ve zeki yönetim ihtiyacını beraberinde getirir. SDN, bu bağlamda ağ yönetimini daha etkili hale getirebilir.

SDN, ağ kontrolünü merkezi bir yazılım tabanlı kontrol düzlemine taşıyarak ağ yönetimini daha esnek ve programlanabilir hale getirir. Bu, IoT cihazlarından gelen büyük veri akışlarını daha etkili bir şekilde işlemeyi ve yönlendirmeyi sağlar. SDN, ağ kaynaklarını dinamik olarak yönetebilir, trafiği optimize edebilir ve IoT cihazları arasında daha akıllı ve hızlı bir iletişim sağlayabilir.

Örneğin, bir IoT uygulamasında, bir dizi sensör tarafından toplanan veriler SDN tarafından yönetilen ağ üzerinden hızlı bir şekilde iletilerek, bulut tabanlı analiz veya yerel işleme algoritmalarına yönlendirilebilir. SDN, ağ üzerindeki trafik yönetimini ve kaynak tahsisini dinamik olarak ayarlayarak, IoT cihazları arasında daha iyi bir performans ve daha

düşük gecikme süreleri sağlayabilir. Bu entegrasyon, büyük ölçekli IoT uygulamalarının yönetimini ve optimize edilmesini kolaylaştırabilir.

2.2. Sistem Mimarisi

2.2.1. Kullanılan İşletim Sistemleri

IoT (Nesnelerin İnterneti) uygulamaları için kullanılan işletim sistemleri, sınırlı kaynaklara sahip cihazlar ve düşük güç tüketimi gereksinimleri göz önüne alınarak tasarlanmıştır. Bu işletim sistemleri, IoT cihazlarının güvenilir, enerji verimli ve ölçeklenebilir bir şekilde çalışmasını sağlar. Bazı IoT için kullanılan işletim sistemleri aşağıdaki gibidir:

Contiki: Contiki, IoT uygulamaları için hafif bir işletim sistemidir. Bu işletim sistemi, sınırlı kaynaklara sahip cihazlar için tasarlanmıştır ve geniş bir cihaz yelpazesi için kullanılabilir. Contiki, düşük güç tüketimi ve IEEE 802.15.4 gibi kablosuz iletişim standartlarına uyum sağlar. Örneğin, akıllı ev sensörleri, Contiki işletim sistemi kullanarak enerji tasarruflu bir şekilde çalışabilir.

TinyOS: özellikle kablosuz sensör ağları ve düşük güç tüketimi gereksinimlerine sahip cihazlar için tasarlanmış bir açık kaynaklı gömülü işletim sistemidir. TinyOS, düşük güç tüketimi, hızlı tepki süreleri ve sınırlı kaynaklara sahip cihazlar için optimize edilmiştir. Özellikle IoT (Nesnelerin İnterneti) uygulamaları, kablosuz sensör ağları ve çeşitli gömülü sistem projeleri için kullanılır.

TinyOS, bileşen tabanlı bir mimariye sahiptir ve bu sayede farklı işlevselliğe sahip bileşenlerin kolayca bir araya getirilmesine olanak tanır. Bu, farklı cihazlar ve uygulamalar için özelleştirilebilirlik sağlar.

Bu işletim sistemi, küçük ve enerji verimli sensör ağı cihazları için özellikle uygundur ve bu nedenle IoT alanında yaygın olarak kullanılır. TinyOS topluluğu, sürekli olarak güncellenen ve geliştirilen bir dizi bileşen ve kütüphane sunar, bu da geliştiricilere hızlı bir şekilde projelerini başlatma ve özelleştirme olanağı sağlar.

RIOT: RIOT, düşük güç tüketimi ve düşük veri hızları gerektiren IoT uygulamaları için açık kaynaklı bir işletim sistemidir. RIOT, çeşitli sensör ağları, akıllı şehir uygulamaları ve akıllı tarım projeleri gibi birçok IoT alanında kullanılır. Aynı zamanda çoklu platformları destekler, bu nedenle farklı cihazlar ve mikrodenetleyiciler üzerinde çalışabilir.

FreeRTOS: FreeRTOS, gömülü sistemler ve IoT uygulamaları için popüler bir açık kaynaklı işletim sistemidir. FreeRTOS, düşük maliyetli ve enerji verimli IoT cihazları için özellikle uygundur. Örneğin, tıbbi cihazlar, güvenlik kameraları ve endüstriyel otomasyon sistemleri gibi çeşitli uygulamalarda kullanılır.

Zephyr: Zephyr, çok çeşitli IoT cihazlarını desteklemek için tasarlanmış açık kaynaklı bir işletim sistemidir. Zephyr, geniş bir cihaz yelpazesi için uyumluluk sağlar ve hafif, düşük güç tüketimli cihazlar için idealdir. Örneğin, akıllı giyilebilir teknolojiler, tıbbi cihazlar ve endüstriyel IoT uygulamaları Zephyr işletim sistemi kullanarak geliştirilebilir.

Bu işletim sistemleri, IoT cihazlarının özelliklerine ve gereksinimlerine göre seçilebilir ve çeşitli uygulamalarda kullanılabilir. Her biri, IoT dünyasında farklı gereksinimleri karşılamak üzere geliştirilen özel özelliklere sahiptir ve IoT projelerinin başarılı bir şekilde geliştirilmesini sağlar.

2.2.2. Contiki İşletim Sistemi

2002 yılında Adam Dunkels liderliğinde geliştirilen, C programlama diliyle yazılmış açık kaynaklı bir işletim sistemi olan Contiki, IoT (Nesnelerin İnterneti) dünyasında önemli bir rol oynamıştır. Contiki, özellikle sınırlı kaynaklara sahip cihazlar ve düşük güç tüketimi gereksinimlerini göz önünde bulundurarak tasarlanmıştır. Bu işletim sistemi, kaynak kısıtlı IoT cihazları için optimize edilmiş, hafif ve güvenilir bir çözümdür.

Contiki'nin gelişimi, öncelikle sınırlı kaynaklara sahip cihazlarda TCP/IP protokol yığınının başarıyla uygulanmasından kaynaklanmıştır. Ayrıca, Contiki'nin temelleri mikro IP (uIP) olarak bilinen bir protokol üzerine inşa edilmiştir. 2003 yılında ilk olarak tanıtılan Contiki, düşük güç tüketimi gereksinimleri göz önüne alındığında oldukça etkili bir işletim sistemidir. Tipik bir Contiki konfigürasyonu, sadece 2 KB RAM ve 40 KB ROM bellek gerektirir ve IPv4 ve IPv6 üzerinden iletişim sağlama yeteneğine sahiptir.

Contiki'nin başlıca özelliklerinden biri, farklı donanım platformlarına kolayca yerleştirilebilmesi için yapılandırılabilir olmasıdır. Bu işletim sistemi, geniş bir cihaz yelpazesi için uyumlu ve platformdan bağımsızdır. Ayrıca, belirli platformlara uygulanırken sadece az sayıda platforma bağlı bileşenin değiştirilmesi gerektiğinden, geliştirme sürecini kolaylaştırır. Contiki, dokümantasyon, uygulama kütüphaneleri ve örnek uygulamalar gibi kaynakları içeren açık kaynak bir ekosistem sunar. Bu nedenle Contiki, IoT dünyasında güvenilir ve hafif bir işletim sistemi olarak tercih edilir.

Contiki işletim sistemi içinde yer alan temel katmanları aşağıdaki gibidir:

Dökümantasyon (doc):

- ❖ Contiki işletim sisteminin dökümantasyonunu içerir.
- ❖ Temel bilgileri, belgeleri ve kullanım kılavuzlarını sağlar.
- ❖ Çeşitli platformlar ve bileşenler hakkında detaylı bilgileri içerir.
- ❖ İşletim sisteminin geliştirilme aşamasında olan kısımları hakkında bilgiler de içerebilir.

Uygulama Kütüphaneleri (apps):

- ❖ Contiki işletim sistemi için özel uygulama kütüphanelerini içerir.
- ❖ Bu kütüphaneler, özellikle örnekler dizininde yer alan uygulamalar tarafından kullanılır.
- ❖ Direkt olarak çalıştırılabilen Contiki uygulamalarını barındırır.

Örnekler (examples):

- ❖ Apps kütüphanelerini kullanan örnek uygulamaları içerir.
- ❖ Örnek uygulamalar genellikle apps kütüphanelerinden süreçleri yürüten işlemleri tanımlar.
- ❖ Her bir örneğe ait dosyalar farklı dizinlerde bulunabilir ve işletim sistemi özelliklerini kullanarak farklı senaryoları simüle edebilir.

Çekirdek (core):

- ❖ Contiki işletim sisteminin temel çekirdek (core) katmanını içerir.
- ❖ İşletim sistemi çekirdeği, farklı katmanlara ait alt dizinleri içerir ve sistemin temel işlevselliğini sağlar.

İşlemci (cpu):

- ❖ Bu bölüm, işletim sisteminin üzerinde çalıştığı donanıma ait mikro denetleyici dosyalarını içerir.
- ❖ Örneğin, cpu/msp430/ dizini Sky mote gibi belirli cihazlar için gereken mikro denetleyici sürücülerini içerebilir.

Platform (platform):

- ❖ Platforma özgü yapılandırmayı ve donanımlara ait sürücülerini tanımlar.
- ❖ İşletim sisteminin farklı donanım platformlarına kolayca taşınabilmesini sağlar

Araçlar (tools):

- ❖ Seri arabirim üzerinden platforma özgü donanımların yeniden programlanabilmesi için araçlar sağlar.

- ❖ Contiki işletim sistemi içindeki uygulamaların simülasyonunu gerçekleştirmek için COOJA simülatörünü içerebilir.

Bu katmanlar, Contiki işletim sisteminin temel yapısını oluşturur ve işletim sistemi belgelerini, uygulamalarını, çekirdek işlevselliğini ve donanım platformlarına özgü sürücülerini içerir. Bu bölümler, Contiki geliştiricilerine işletim sistemi ile ilgili kaynaklara kolayca erişme olanağı sağlar.(URL-4).

2.2.3. Contiki-NG

Contiki-NG (Contiki Next Generation), orijinal Contiki işletim sisteminin bir çatallanmasıdır ve Contiki'nin gelişmiş bir sürümü olarak düşünülebilir.(URL-3). Contiki-NG, IoT (Nesnelerin İnterneti) cihazları ve kablosuz sensör ağları için tasarlanmış bir açık kaynaklı işletim sistemidir. Contiki-NG'nin Contiki'den farklılık ve özellikleri aşağıdaki gibidir:

Modern ve Daha Güncel Bir Geliştirme: Contiki-NG, orijinal Contiki'den daha modern bir geliştirme sürecine sahiptir. Daha yeni C derleyicileri ve geliştirme araçları ile daha uyumlu ve güncel bir şekilde geliştirilmektedir.

IPv6 Desteği: Contiki-NG, IPv6 desteği konusunda daha fazla iyileştirme sunar. Bu, IoT cihazları için daha geniş bir adresleme alanı ve geleceğe yönelik daha iyi bir hazırlık sağlar.

Platform Çeşitliliği: Contiki-NG, farklı donanım platformlarına daha fazla destek sunar. Bu, daha geniş bir cihaz yelpazesi için kullanılabilirlik sağlar.

Güvenlik Geliştirmeleri: Contiki-NG, güvenlik konusunda geliştirmeler sunar ve özellikle düşük güç tüketimi gereksinimlerine sahip IoT cihazları için güvenli iletişimi teşvik eder.

Etkinlik Tabanlı Programlama (Event-Driven Programming): Contiki-NG, etkinlik tabanlı programlama modelini daha iyi destekler. Bu, olayların tetikleyicilerine dayalı olarak çalışan IoT uygulamalarının daha iyi tasarlanmasını sağlar.

Daha Geniş Topluluk: Contiki-NG'nin geliştirilmesi ve bakımı için daha geniş ve aktif bir topluluk vardır. Bu, sorunların daha hızlı çözülmesi ve işletim sisteminin sürekli olarak güncellenmesi anlamına gelir.

Performans ve Daha İyi Bellek Yönetimi: Contiki-NG, genellikle daha iyi performans ve daha iyi bellek yönetimi sağlar. Bu, IoT cihazlarının daha verimli çalışmasını sağlar.

Contiki ve Contiki-NG, her ikisi de IoT cihazları için güçlü işletim sistemleri olsa da, Contiki-NG, daha güncel ve genişletilebilir bir platform olarak öne çıkar. Seçim, projenin ihtiyaçlarına ve donanımınıza bağlı olarak yapılmalıdır.



Şekil 2.2.3. Contiki İşletim Sistemi Genel Mimarisi

2.2.4. Cooja

COOJA, Contiki işletim sistemi ve Contiki-NG için bir simülasyon ve geliştirme ortamıdır ve IoT (Nesnelerin İnterneti) uygulamalarının geliştirilmesi ve test edilmesi için kullanılır. COOJA, IoT cihazlarının davranışlarını simüle etmek ve farklı senaryolarda nasıl çalıştıklarını görmek için kullanışlı bir araçtır.

COOJA'nın çalışma prensibi şu şekildedir:

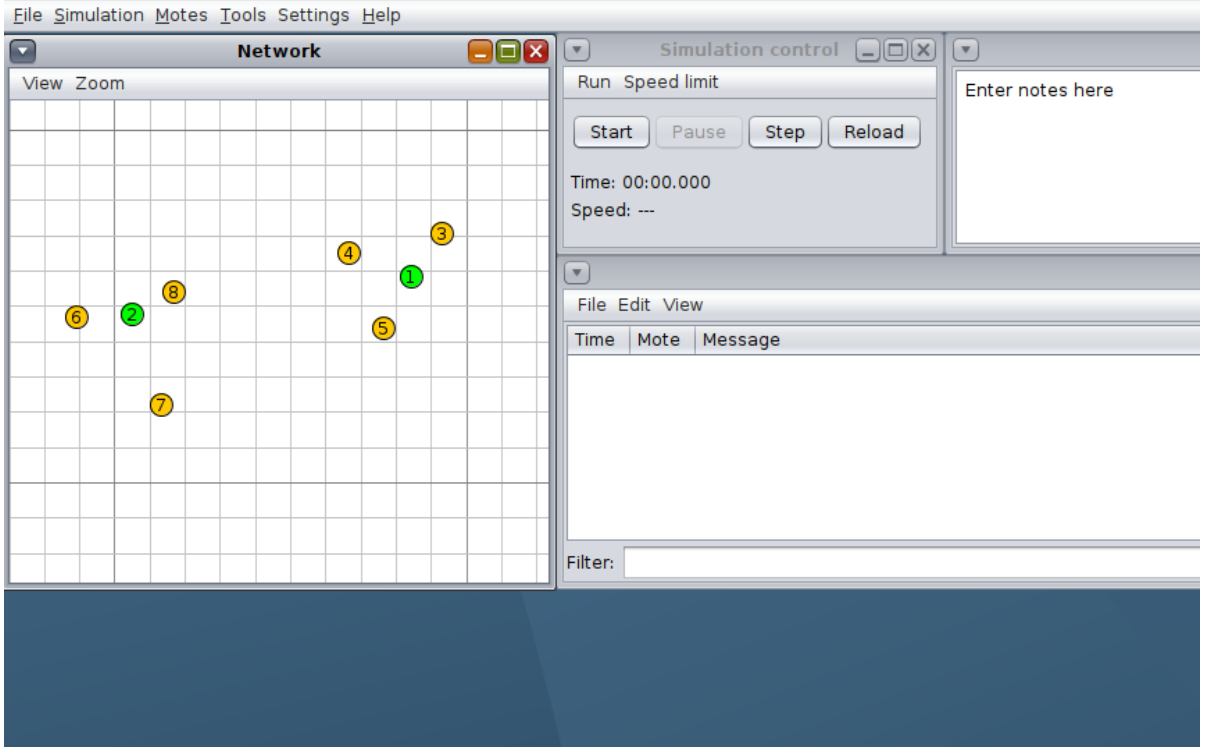
COOJA, Contiki işletim sistemi veya Contiki-NG üzerinde çalışan IoT cihazlarının sanal bir ortamda simüle edilmesini sağlayan bir simülasyon ve geliştirme ortamıdır. Bu, gerçek dünya cihazları kullanmadan IoT uygulamalarını test etmek ve hata ayıklamak için çok kullanışlıdır. COOJA, farklı ağ topolojileri, iletişim protokolleri ve senaryolarını oluşturmamıza olanak tanır.

Örnek olarak aşağıdaki senaryolar verilebilir:

Akıllı Ev Simülasyonu: COOJA ile akıllı ev sensörleri ve cihazları gibi IoT cihazları simüle edilebilir. Bu, ev otomasyonu uygulamalarının nasıl çalıştığını test etmek için kullanışlıdır.

Kablosuz Sensör Ağı Senaryoları: COOJA, kablosuz sensör ağlarının nasıl davrandığını anlamak ve optimize etmek için kullanılabilir. Örneğin, ağın enerji tüketimini veya iletişim protokollerini test etmek için kullanılabilir.

Protokol Geliştirme ve Testi: COOJA, yeni iletişim protokolleri geliştirmek ve mevcut protokolleri test etmek için kullanılabilir. Protokol değişikliklerinin ağ performansına etkisini gözlemlemek için idealdir.



Şekil 2.2.4. Cooja Simülatörü Arayüzüne Düğümlerin Eklenmesi

COOJA, IoT uygulamalarını geliştirmek, test etmek ve hata ayıklamak için güçlü bir araçtır. Simülasyonlar, gerçek dünya koşullarını taklit etmenize ve IoT cihazlarının davranışlarını anlamana olanak tanır. Bu, geliştirme sürecini hızlandırabilir ve IoT projelerini daha güvenilir hale getirebilir.

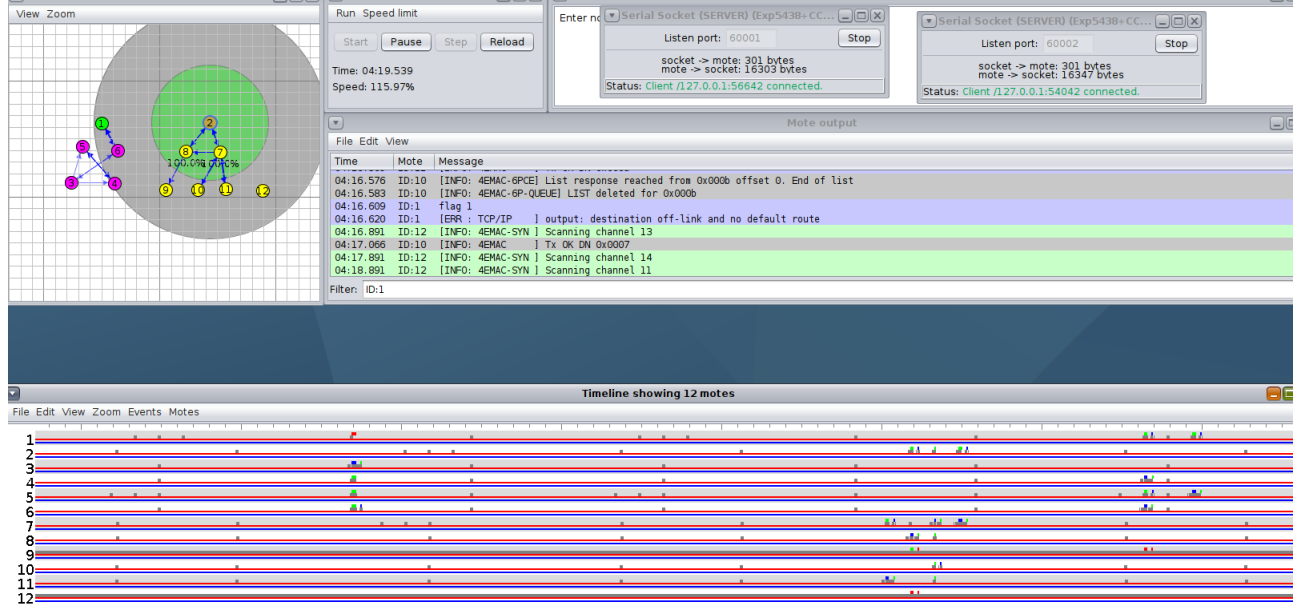
COOJA, genellikle Java tabanlı bir uygulama olarak sunulur ve aşağıdaki temel adımları izler:

Simülasyon Senaryosu Oluşturma: COOJA kullanıcı arayüzü aracılığıyla, simülasyon senaryosu oluşturulabilir. Bu, simüle edilecek cihazları, ağ topolojilerini, iletişim protokollerini ve senaryoları tanımlamayı sağlar.

Simülasyon Başlatma: Senaryo tanımlandıktan sonra, COOJA simülasyonu başlatır. Bu sırada simülasyon, sanal cihazlar arasındaki iletişimi ve olayları takip eder.

Simülasyon İzleme: COOJA, simülasyonun gerçek zamanlı izlenmesine olanak tanır. Bu, sanal cihazların davranışlarını ve ağ iletişimini gözlemlemenize yardımcı olur.

Sonuçları İnceleme: Simülasyon tamamlandığında, COOJA, sonuçları analiz etmek için çeşitli araçlar ve raporlar sunar. Bu, IoT uygulamanın performansını ve doğruluğunu değerlendirmeyi sağlar.



Şekil 2.2.5. Cooja Simülatorü Arayüzünde Kurulan Ağın İzlenmesi

Şekil 2.2.5'te kurulan örnek bir IOT ağı görülmektedir. Bu ağda kök düğümler ve onlara bağlı istemci düğümler bulunmaktadır. Düğümlerin birbiri ile olan ilişkileri anlık olarak simülasyon ekranı sayesinde izlenebilmektedir.

Kök düğümlerin hangi portları kullanacakları ve bağlantının durdurulup başlatılması, ağın çalıştığı toplam süre, senkronize olup olmadıkları takip edilebilmektedir.

Filtreleme ekranına uygun formatta (Örnek ID:1, 1 numaralı düğümün hareketlerini) yapılan aramalar ile istenen düğümlerin ağdaki davranışları gözlemlenebilmektedir.

3. YÖNLENDİRME PROTOKOLLERİ

Bilgisayar ağlarında kablolu ve kablosuz yönlendirme protokolleri, veri paketlerinin kaynaktan hedefe doğru yönlendirilmesini sağlayan önemli bileşenlerdir. Kablolu ve kablosuz ağlarda kullanılan yönlendirme protokollerini aşağıdaki gibidir.

3.1. Kablolu Ağ Yönlendirme Protokolleri:

3.1.1. OSPF (Open Shortest Path First)

OSPF, iç büyük çaplı kablolu ağlar ve büyük ölçekli kurumsal ağlarda yaygın olarak kullanılan bir iç yönlendirme protokolüdür.

Link-state bir protokol olarak çalışır ve ağdaki her yönlendirici hakkında ayrıntılı bilgileri toplar. Bu bilgileri kullanarak en kısa yolu hesaplar.

OSPF, iç büyük ölçekli ağlarda ve büyük kurumsal ağlarda kullanılan bir iç yönlendirme protokolüdür. Bu protokol, Link-State Routing (Bağlantı Durumu Yönlendirme) olarak adlandırılan bir yönlendirme algoritmasını temel alır. OSPF, ağdaki tüm yönlendiricilerin birbirleriyle iletişim kurmasına olanak tanır ve her yönlendiricinin ağ topolojisi hakkında ayrıntılı bilgi toplamasını sağlar. Bu bilgi, yönlendiricilerin en kısa yolu hesaplamalarına yardımcı olur.

Örnek olarak, bir büyük şirketin kurumsal ağı düşünelim. Bu ağ, farklı ofisler, departmanlar ve binlerce kullanıcı içeriyor olabilir. OSPF, bu karmaşık ağın yönlendirmesini kolaylaştırmak için kullanılabilir. Her yönlendirici, ağın topolojisini belirlemek ve ağdaki diğer yönlendiricilerle bağlantılarını sürdürmek için OSPF paketlerini kullanır. Bu sayede, veriler en kısa yolu takip ederek hedefe iletilir, ağda olası hata durumlarında otomatik toparlanma sağlanır ve ağın performansı optimize edilir.

OSPF ayrıca, farklı ağ segmentlerinin önceliklerini ve maliyetlerini dikkate alarak yönlendirme kararlarını verir. Örneğin, ağın bir bölümü fiber optik kablo ile bağlanırken, diğer bölümü daha yavaş bir bağlantıya sahip olabilir. OSPF, bu bağlantıların özelliklerini değerlendirir ve verileri en hızlı ve verimli şekilde yönlendirir. Bu sayede, ağ trafiği en iyi şekilde yönetilir ve kullanıcılar hızlı bir ağ deneyimi yaşarlar.

3.1.2. EIGRP (Enhanced Interior Gateway Routing Protocol)

EIGRP, Cisco tarafından geliştirilen ve genellikle Cisco ağ ekipmanlarıyla kullanılan bir iç yönlendirme protokolüdür.

Hem mesafe vektörü hem de bağlantı durumu özellikleri içerir ve hızlı toparlanma yetenekleri sunar.

EIGRP, Cisco tarafından geliştirilen ve özellikle Cisco ağ ekipmanları ile uyumlu olan bir iç yönlendirme protokolüdür. EIGRP'nin temel özelliği, hem distance-vector hem de link-state yönlendirme protokollerinin özelliklerini birleştirmesidir. Bu, EIGRP'nin ağda daha iyi performans ve hızlı toparlanma yetenekleri sunmasını sağlar.

Örnek olarak, büyük bir şirketin karmaşık bir ağı düşünelim. Bu ağ, birden fazla ofis, veri merkezi ve farklı alt ağlardan oluşabilir. EIGRP, bu ağı yönlendirmek için kullanılabilir. Her Cisco yönlendirici, ağın iç yapısını anlamak ve yönlendirme kararlarını almak için EIGRP mesajlarını kullanır. Bu sayede, veri paketleri en hızlı ve en güvenilir yolu takip ederek hedefe iletilir.

EIGRP ayrıca, ağdaki değişikliklere dinamik olarak yanıt verebilir. Örneğin, bir ağ bağlantısı kesildiğinde veya yeni bir bağlantı eklendiğinde, EIGRP otomatik olarak yönlendirme tablosunu günceller ve veri trafiğini yeni koşullara uygun şekilde yönlendirir. Bu, ağın daha yüksek kullanılabilirliği ve otomasyonunu sağlar. EIGRP, kurumsal ağlarda, veri merkezlerinde ve büyük ölçekli ağlarda yaygın olarak kullanılmaktadır ve cisco ağ ekipmanlarıyla entegre çalışarak ağ yöneticilerine büyük bir esneklik ve kontrol sunar.

3.1.3. RIP (Routing Information Protocol)

RIP, eski bir iç yönlendirme protokolüdür ve küçük ölçekli ağlarda kullanılabilir.

Distance-vector (mesafe vektörü) bir protokol olarak çalışır ve en kısa yol hesabında hop sayılarını kullanır.

RIP, bilgisayar ağlarında kullanılan eski bir iç yönlendirme protokolüdür. İlk olarak 1980'lerde geliştirilen ve RFC 1058 ile standartlaştırılan RIP, mesafe vektör yönlendirme algoritmasını kullanır. Bu protokol, bir ağdaki yönlendiriciler arasında rotaların değişimini ve güncellemelerini yönetmek için tasarlanmıştır.

RIP'in temel çalışma prensibi, her yönlendiricinin, komşu yönlendiricilere bağlı ağların durumu hakkında bilgi topladığı ve bu bilgileri diğer yönlendiricilere periyodik olarak yayımladığı bir döngüyü takip etmesidir. Bu bilgiler, yönlendiriciye uzak ağların

"mesafelerini" ve "hop" sayılarını bildirir. RIP, bu bilgilere dayanarak en kısa yolun hesaplanmasını sağlar ve yönlendirme tablosunu günceller.

RIP, basit ve küçük ölçekli ağlar için uygundur, ancak büyük ve karmaşık ağlarda sınırlamalara sahiptir. Özellikle büyük ağlarda yavaş güncelleme süreleri ve sınırlı ölçeklenebilirlik nedeniyle tercih edilmemektedir. RIP, daha modern iç yönlendirme protokolleri (örneğin OSPF veya EIGRP gibi) ile yer değiştirmiş olsa da, tarihsel bir öneme sahiptir ve ağ yönlendirme konseptlerini anlamak için hala önemli bir protokoldür.

3.2. Kablosuz Ağ Yönlendirme Protokolleri

3.2.1. 802.11 (Wi-Fi) Protokolleri

Kablosuz yerel ağlarda (Wi-Fi) kullanılan protokoller, 802.11 a, b, g, n, ac, ax gibi standartlara dayanır.

Bu protokoller, kablosuz iletişimi düzenler ve cihazların erişim noktalarına (access point) bağlanmasını sağlar.

802.11, kablosuz yerel ağ (Wi-Fi) iletişimi için kullanılan bir dizi IEEE standartlarını ifade eder. Bu standartlar, kablosuz iletişimin çeşitli yönlerini düzenler ve geliştirir. İşte bazı 802.11 (Wi-Fi) protokolleri:

- ✓ 802.11a: 802.11a, 5 GHz frekans bandında çalışır ve 54 Mbps'ye kadar veri hızları sunar. Bu protokol, daha yüksek hızlar ve daha az gürültü içeren 5 GHz bandını kullanarak, veri iletimini geliştirmiştir.
- ✓ 802.11b: 802.11b, 2.4 GHz frekans bandında çalışır ve 11 Mbps'ye kadar veri hızları sağlar. Bu, ilk popüler Wi-Fi standardıdır ve geniş bir kablosuz erişim noktası yelpazesi ile yaygın olarak kullanılmıştır.
- ✓ 802.11g: 802.11g, 2.4 GHz bandında çalışır ve 54 Mbps'ye kadar veri hızları sunar. Bu standardın temel özelliği, 802.11b ile uyumlu olması ve daha yüksek hızlar sunmasıdır.
- ✓ 802.11n: 802.11n, 2.4 GHz ve 5 GHz bantlarında çalışabilir ve 600 Mbps'ye kadar veri hızlarına ulaşabilir. MIMO (Multiple Input Multiple Output) teknolojisi sayesinde daha iyi kapsama alanı ve performans sunar.

- ✓ 802.11ac: 802.11ac, yüksek hızlarda kablosuz iletişim sağlar ve 5 GHz bandında çalışır. 1 Gbps'ye kadar veri hızları sunar ve MIMO teknolojisiyle daha fazla cihazın aynı anda yüksek hızda bağlanmasını destekler.
- ✓ 802.11ax (Wi-Fi 6): En yeni Wi-Fi standardı olan 802.11ax, daha fazla cihazın daha verimli bir şekilde ağa bağlanmasını sağlar. 2.4 GHz ve 5 GHz bantlarında çalışabilir ve 10 Gbps'ye kadar veri hızlarına ulaşabilir.
- ✓ Bu 802.11 standartları, kablosuz iletişim teknolojisinin zaman içinde nasıl geliştiğini ve hızlandığını göstermektedir. Her yeni standardın daha yüksek hızlar, daha iyi güvenlik ve daha fazla cihazın desteklenmesi gibi avantajları vardır. Wi-Fi kullanıcıları, ihtiyaçlarına ve donanım uyumluluklarına bağlı olarak bu farklı standartlardan birini veya birkaçını kullanabilirler.

3.2.2. Mobile IP

Mobile IP, mobil cihazların farklı kablosuz ağlar arasında hareket ederken IP adreslerini korumak ve bağlı kalmak için kullanılan bir protokoldür.

Bu, kablosuz ağlarda hareket halindeki cihazlar için önemlidir.

3.2.3. BGP (Border Gateway Protocol)

BGP, büyük ölçekli kablosuz ağlar ve geniş alan ağlar (WAN) için kullanılan bir dış yönlendirme protokolüdür.

BGP (Border Gateway Protocol), internetin temel yönlendirme protokolüdür ve büyük ölçekli ağlarda kullanılır. BGP, bağımsız sistemler arasındaki (AS - Autonomous System) yönlendirme bilgisini paylaşmak ve yönlendirme kararlarını almak için kullanılır. AS'ler, kendi iç ağ yapılarına sahip bağımsız ağlardır ve internet trafiğini yönlendiren önemli noktalardır. BGP, bu AS'ler arasında veri trafiğini yönlendirmek ve internetin küresel çapta çalışmasını sağlamak için kullanılır.

BGP'nin temel özelliği, yol vektörü (path-vector) yönlendirme protokolü olmasıdır. Bu, BGP'nin her bir yolunun özelliğini tanımlayan bir yol vektörü kullanarak en iyi yolun seçilmesini sağlar. BGP, farklı AS'ler arasındaki erişim noktalarını ve en iyi rotaları hesaplamak için çok sayıda ölçüt ve politika kullanır. Bu, ağ yöneticilerine esneklik sağlar ve trafik mühendisliği yapma yeteneği sunar.

BGP, internetin büyüklüğü ve karmaşıklığı göz önüne alındığında son derece önemlidir. İnternet trafiğini yönlendiren anahtar bileşenlerden biri olan BGP, ağların güvenliğini ve verimliliğini sağlamak için tasarlanmıştır. Bununla birlikte, yanlış yapılandırılmış veya kötü niyetli kullanılan BGP yönlendirmesi, internet üzerinde kesintilere ve güvenlik sorunlarına neden olabilir. Bu nedenle BGP, ağ yöneticileri ve güvenlik uzmanları için sürekli olarak gözden geçirilmesi gereken bir protokoldür.

3.3. RPL (Routing Protocol for Low-Power and Lossy Networks)

RPL, enerji açısından kısıtlı ve sık kayıp yaşanan ağlarda etkili yönlendirme sağlamak amacıyla tasarlanmış bir ağ yönlendirme protokolüdür. Genellikle Nesnelerin İnterneti (IoT) uygulamaları, akıllı şebekeler, endüstriyel otomasyon sistemleri ve çevresel izleme gibi alanlarda kullanılır. RPL, bu tür ağlarda önemli olan düşük enerji tüketimi, düşük veri kaybı ve düşük bant genişliği kullanımını dikkate alır.

RPL'nin temel özelliklerinden biri, düşük güç tüketimini desteklemesi ve ağdaki cihazların pil ömrünü uzatmasına yardımcı olmasıdır. Bu, sensör düğümleri ve düşük güçlü cihazlar gibi enerji kısıtlı cihazlar için kritiktir. RPL, cihazların iletişimlerini optimize ederek ve enerji açısından verimli yolları seçerek enerji tasarrufu sağlar.

Örnek olarak, bir tarım uygulamasını düşünelim. Çiftçiler, tarlalarında sensör ağları kullanarak toprak nemini, sıcaklığını ve diğer parametreleri izlerler. Bu sensörler, enerji açısından sınırlıdır ve tarla boyunca yayılmışlardır. RPL, bu sensörlerin verilerini veri merkezine düşük güç tüketimi ile güvenilir bir şekilde iletebilir. Örneğin, veri iletimi sırasında kaybolan paketlerin yerine geçen yolları seçebilir ve enerji tüketimini minimize edebilir.

RPL, ağın topolojisinin sürekli olarak değişebildiği dinamik ağlarda da kullanılabilir. Örneğin, bir acil durumda kullanılan kablosuz sensör ağı, felaket bölgesindeki veri toplama ve iletimi için RPL'yi kullanabilir. Bu, kurtarma ekiplerinin hızla ve verimli bir şekilde bilgiye erişmesini sağlayabilir.

RPL, LLN'lere özgü gereksinimleri ele alarak, düşük güç, kayıplı ağlar ve geniş coğrafi alanlar gibi zorlu koşullarda etkili bir şekilde çalışabilen bir yönlendirme protokolü olarak önemli bir rol oynamaktadır. Bu nedenle, IoT ve sensör ağı uygulamalarının yanı sıra çevresel izleme ve enerji yönetimi gibi birçok alanda RPL'nin etkisi büyük olmuştur.

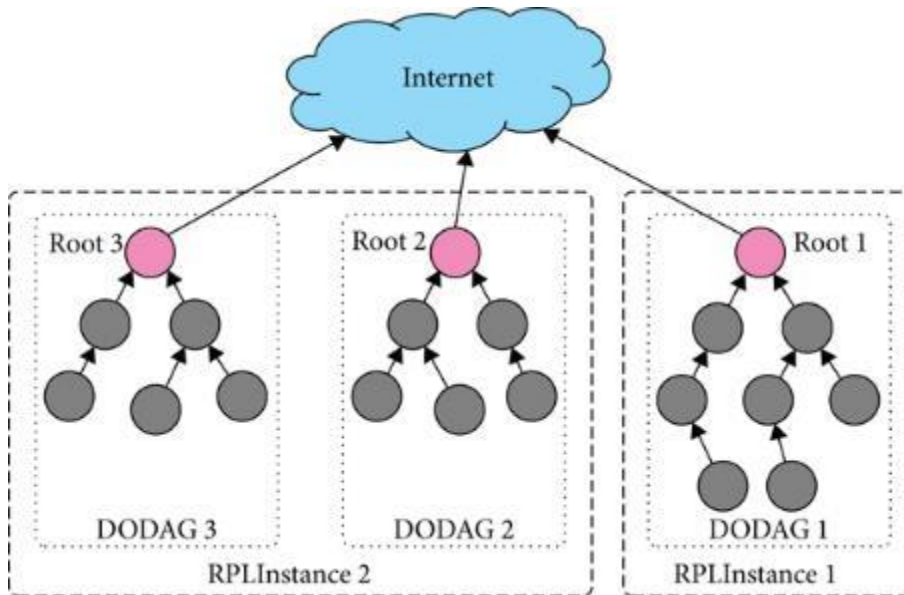
RPL, Internet Mühendisliği Görev Gücü (IETF - Internet Engineering Task Force) tarafından geliştirilen ve standardize edilen bir yönlendirme protokolüdür. IETF, internetin temel protokollerini ve standartlarını geliştiren bir organizasyondur ve RPL, düşük güç tüketimi ve kayıplı ağlar (LLN - Low-Power and Lossy Networks) için tasarlanmış bir yönlendirme protokolü olarak bu bağlamda önemlidir.

RPL, IETF'nin 6LoWPAN (IPv6 over Low-Power Wireless Personal Area Networks) ve CoAP (Constrained Application Protocol) gibi LLN ile ilgili çalışmalarıyla ilişkilidir. Bu protokoller, IoT (Nesnelerin İnterneti) cihazları, sensör ağları ve enerji açısından kısıtlı cihazlar gibi düşük güçlü ve kayıplı ağlarda kullanılmak üzere tasarlanmıştır. RPL, bu tür ağlarda veri iletimini optimize etmek ve enerji tüketimini minimize etmek için geliştirilen bir yönlendirme protokolü olarak bu çerçevede yer alır.

IETF, RPL'nin standardizasyon sürecini yönetti ve RPL'nin RFC (Request for Comments) belgeleri aracılığıyla resmi bir IETF standardı haline gelmesini sağladı. Bu, RPL'nin LLN ağlarında yaygın olarak kullanılmasını ve farklı üreticilerin ve geliştiricilerin bu protokolü uygulamalarında güvenle kullanabilmesini sağlar. Bu bağlamda, RPL'nin IETF ile ilişkisi, düşük güçlü ve kayıplı ağlar için standartlar ve protokoller geliştiren birinci sınıf bir standart olması açısından büyük öneme sahiptir.

3.3.1. RPL Yapısı

RPL, özelleştirilmiş terminoloji ve kavramlar içeren bir yönlendirme protokolüdür.



Şekil 3.3.1. RPL Örneği (URL-2)

RPL terminolojisi şunları içerir:

DAG (Directed Acyclic Graph - Yönlendirilmiş Asiklik Grafik): DAG, RPL'nin temel topolojik yapıdır. Bir DAG, düşük güçlü ve kayıplı ağlarda kullanılan RPL tarafından oluşturulan yönlendirme ağını temsil eder. DAG, yönlendiricilerin ve cihazların bağlantılarını gösteren düğümlerden ve bu bağlantıların yönlendirme yollarını gösteren yönlendirilmiş oklardan oluşur. DAG, her cihazın veya yönlendiricinin ağdaki konumunu ve ilişkilerini gösterir.

DODAG (Destination-Oriented Directed Acyclic Graph - Hedefe Yönlendirilmiş Asiklik Grafik): DODAG, DAG'ın bir alt kümesidir ve belirli bir hedefe yönlendirme yapmak için kullanılır. DODAG, veri paketlerinin hedef cihaza ulaşması için kullanılan yolun belirtildiği bir yolun ana çerçevesini sağlar. DODAG, veri iletimi için kullanılan yönlendirme yollarını temsil eder.

Root (Kök): DAG yapısındaki kök düğüm, DODAG'ın başlangıç noktasıdır ve ağdaki tüm diğer düğümlere göre daha üst konumdadır. Kök düğüm, ağdaki tüm DODAG yollarının başladığı ve sonlandığı yerdir.

Parent (Ebeveyn): Bir düğümün yolu DODAG yapısındaki üst bir düğüme yönlendirilirse, üst düğüm "ebeveyn" olarak kabul edilir. Ebeveyn düğüm, alt düğüme yönlendirme bilgilerini sağlar ve alt düğüm, veri iletimi için ebeveynine güvenir.

Child (Çocuk): Bir düğümün yolu DODAG yapısındaki alt bir düğüme yönlendirilirse, alt düğüm "çocuk" olarak kabul edilir. Çocuk düğüm, veri iletimi için ebeveynine güvenir ve yönlendirme bilgilerini ebeveyn düğümünden alır.

Rank (Sıralama): Her düğümün bir sıralaması vardır ve bu sıralama, DODAG içindeki konumunu belirtir. Düğümler, daha yüksek sıralamalı düğümlere yönlendirme yaparlar. Sıralama, düğümlerin DODAG yapısındaki rolünü belirler.

Bu RPL terminolojisi, düşük güçlü ve kayıplı ağlarda etkili yönlendirme sağlamak için kullanılan DODAG yapısının temel kavramlarını açıklar. RPL, bu terminolojiyi kullanarak veri iletimini optimize eder ve enerji tüketimini minimize eder.

RPL, belirli bir ağdaki yönlendirme işlemlerini tanımlayan ve izole eden RPL örneğini oluşturma yeteneğine sahiptir. Bir RPL örneği, aynı fiziksel ağ üzerinde birden fazla mantıksal veya bağımsız yönlendirme alanı yaratmak için kullanılır. Her RPL örneği, kendi ağ topolojisini, yönlendirme politikalarını ve yönlendirme davranışını tanımlar.

RPL örneği oluşturulurken, ağdaki yönlendiriciler belirli bir örneğe katılmak üzere yapılandırılır. Bir RPL örneği, genellikle belirli bir uygulama veya hizmet gereksinimlerini karşılamak için oluşturulur. Örneğin, bir bina içi sensör ağı ve aynı binadaki endüstriyel otomasyon ağı arasında iki ayrı RPL örneği oluşturulabilir. Bu, her iki ağın bağımsız yönlendirme politikalarına ve gereksinimlerine sahip olmasını sağlar.

RPL örnekleri, ağ yöneticilerine ağdaki farklı segmentleri ve uygulamaları yönetmek için daha fazla esneklik sunar. Her örnek, ağdaki düğümlerin birbirleriyle nasıl iletişim kuracaklarını, yönlendirme davranışlarını ve hedeflere nasıl ulaşacaklarını tanımlar. Bu, büyük ve karmaşık IoT (Nesnelerin İnterneti) ağlarında çoklu uygulama senaryolarına uyum sağlamak için önemlidir.

3.3.2. ICMPv6 Mesajları

ICMPv6 (Internet Control Message Protocol version 6), IPv6 ağı üzerinde iletişim sırasında ağ ve hata kontrol işlemlerini gerçekleştiren bir iletişim protokolüdür. ICMPv6, IPv4'teki ICMP'ye benzer bir rol oynar, ancak IPv6 özelliklerine uygun olarak tasarlanmıştır. IPv6 ağındaki düğümler arasında iletişimde hataları tespit etmek, hata bildirimleri göndermek ve bazı ağ hizmetlerini (örneğin, adres çözümlemesi) yürütmek için kullanılır.

ICMPv6 mesajları çeşitli amaçlar için kullanılır.

Bazı yaygın ICMPv6 mesaj türleri şunlardır:

RS - Yönlendirici İstemi (Router Solicitation): Bu mesajlar, yeni bir düğümün ağa katılmak istediğini bildirir. Düğüm, ağdaki yönlendiricileri bulmak ve kendisine bir IPv6 adresi atanması için RS mesajı gönderir.

RA - Yönlendirici İlanı (Router Advertisement): Yönlendirici, RS mesajını alan düğümlere kendisini tanıtmak ve ağ yapılandırma bilgilerini iletmek için RA mesajları gönderir. Bu mesajlar, düğümlerin ağa katılma ve ağ yapılandırma süreçlerini başlatmalarını sağlar.

NS - Komşu İstemi (Neighbor Solicitation): Bir düğüm, bir diğer düğümün IP adresine ilişkin bilgiyi almak veya bir IP adresinin çözümlemesini yapmak istediğinde NS mesajlarını kullanır. Özellikle, düğümün ağdaki bir komşusu ile iletişim kurmak için NS mesajlarını kullanır.

NA - Komşu İlanı (Neighbor Advertisement): NS mesajlarına yanıt olarak gönderilen NA mesajları, bir düğümün kendi adresini çözümlmek isteyen diğer düğümlere cevap verir. Bu mesajlar, bir düğümün MAC adresini IP adresiyle eşlemek için kullanılır.

ICMPv6 mesajları, IPv6 ağının düzgün çalışmasını ve hata tespiti ile düzeltme süreçlerini destekler. Ayrıca, ağ yapılandırma ve adres çözümlme gibi önemli ağ hizmetlerini sağlar. Bu mesajlar, ağ yöneticileri ve sistem yöneticileri tarafından ağın performansını izlemek ve sorunları tanımlamak için kullanılır.

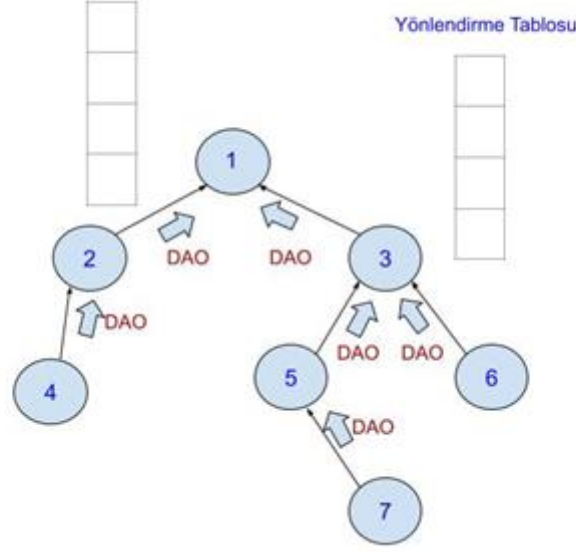
3.3.3. DIO (DODAG Information Object - Yönlendirilmiş Asiklik Grafik Bilgi Nesnesi)

DIO mesajları, RPL ağı içinde bir DODAG yapısını oluşturmak, güncellemek ve yönlendirmek için kullanılır. DIO mesajları, DODAG yapısının kök düğümünden yayımlanır ve ağdaki diğer düğümlere DODAG yapısını oluşturmak ve güncellemek için gerekli bilgileri ileten yönlendirme mesajlarıdır. DIO mesajları, düğümlerin DODAG yapısına katılmak için gereken bilgilere erişmelerini sağlar.

3.3.4. DAO (Destination Advertisement Object - Hedef İlan Nesnesi)

DAO mesajları, düğümlerin DODAG yapısı içindeki komşularını ve hedeflerini tanımlamak ve DODAG yapısındaki düğümlere yönlendirme bilgilerini iletmek için kullanılır. DAO mesajları, düğümler tarafından yayımlandığında, diğer düğümler bu bilgileri alır ve iletişim için yönlendirme tablolarını günceller. DAO mesajları, ağdaki düğümlerin DODAG yapısı içindeki konumlarını belirlemelerine yardımcı olur.

DIO ve DAO mesajları, RPL'nin düşük güçlü ve kayıplı ağlarda veri iletimini optimize etmek ve enerji tasarrufu sağlamak için nasıl çalıştığını yönetir. Bu mesajlar, düğümlerin DODAG yapısını oluşturmaya, güncellemesine ve iletimine yardımcı olarak, IoT cihazları ve diğer düşük güçlü cihazlarla çalışan ağlarda etkili yönlendirme sağlar.



Şekil 3.3.4. RPL depolanmayan modu (Vilajosana vd.,2022)

3.3.5. Yönlendirme Metrikleri

Bir ağdaki farklı yönlendirme yollarının kalitesini, maliyetini veya verimliliğini ölçen değerlerdir. Bu metrikler, veri paketlerinin en iyi yolunu seçerken kullanılır ve genellikle ağ performansını optimize etmeye yardımcı olur. Yaygın olarak kullanılan bazı yönlendirme metrikleri aşağıdaki gibidir:

Hop Sayısı: Hop sayısı, bir yönlendirme yolundaki yönlendiricilerin veya düğümlerin sayısını ölçer. Bu metrik, veri paketinin ağ boyunca kaç düğüm üzerinden geçeceğini belirlemek için kullanılır. Genellikle daha az hop sayısı, daha hızlı iletim anlamına gelir, ancak yol boyunca daha fazla yönlendirici kullanılması gerekebilir.

Maliyet: Maliyet metriği, her bir yolun kullanımının ağa maliyetini temsil eder. Bu maliyet, bant genişliği, gecikme süresi, enerji tüketimi veya diğer faktörlere dayalı olabilir. Daha düşük maliyetli yollar, genellikle tercih edilir, ancak hangi faktörün maliyete daha fazla etki ettiği, uygulamaya bağlı olarak değişebilir.

Bant Genişliği: Bant genişliği metriği, bir yolun üzerinden geçtiği bağlantıların toplam bant genişliğini ölçer. Daha yüksek bant genişliği, daha fazla veri taşıma kapasitesi sağlar ve daha hızlı iletim anlamına gelir.

Gecikme Süresi: Gecikme süresi, bir yönlendirme yolunun veri paketinin bir noktadan diğerine gitmesi için ne kadar zaman aldığını ölçer. Düşük gecikme süresi, daha hızlı iletim anlamına gelir ve gerçek zamanlı uygulamalar için önemlidir.

Jitter: Jitter, veri paketlerinin gönderim aralıklarındaki değişkenlik olarak tanımlanır. Daha düşük jitter, daha istikrarlı bir iletim anlamına gelir ve ses veya video gibi sürekli akış gerektiren uygulamalar için önemlidir.

Güvenilirlik: Güvenilirlik metriği, bir yolun ne kadar güvenilir olduğunu ve veri paketlerinin kaybolma veya hatalı iletilme olasılığını ölçer. Yüksek güvenilirlik, ağın kesintisiz çalışmasını sağlar.

Bu yönlendirme metrikleri, farklı ağ türleri ve uygulamalar için farklı önceliklere sahip olabilir. Yönlendirme protokolleri, bu metrikleri kullanarak en iyi yolun seçilmesini sağlar ve ağ performansını optimize eder.

3.3.6. RPL'nin Amaç Fonksiyonları

Amaç Fonksiyonu (Objective Function), RPL'nin temel bir bileşenidir ve ağdaki düğümlerin nasıl yönlendirme kararları alacaklarını belirler. Amaç Fonksiyonları, ağın özel gereksinimlerine göre düğümlerin yönlendirme kararlarını optimize eder. Aşağıda, Amaç Fonksiyonlarının ana işlevleri hakkında daha fazla ayrıntı bulunmaktadır:

- **Enerji Tüketimini Azaltma:**

Bu Amaç Fonksiyonu, düşük güçlü düğümlerin enerji tasarrufu yapmalarına yardımcı olur. Enerji tüketimini minimize ederek düğümlerin pil ömrünü uzatır.

Düşük enerji tüketimi, IoT cihazlarının daha uzun süre hizmet vermesini sağlar.

- **Gecikmeyi Azaltma:**

İletişimdeki gecikmeyi minimize etmeye odaklanır.

Bazı IoT uygulamaları, düşük gecikme süreleri gerektirir. (örneğin, acil durum bildirimleri)

En hızlı iletişim yollarını seçerek gecikmeyi en aza indirir.

- **Ömür Süresini Maksimize Etme:**

Bu Amaç Fonksiyonu, ağdaki düğümlerin ömür süresini maksimize etmeye odaklanır. Enerji tüketimi ve pil ömrü dikkate alınarak, ağın uzun vadeli kullanımını destekler.

- Veri Akışını Maksimize Etme:

Ağın toplam veri iletim kapasitesini maksimize etmeye çalışır. Veri yoğun IoT uygulamaları için önemlidir.

Veri iletim hızını artırarak daha fazla verinin iletilmesine yardımcı olur.

- Atlama Sayısını Azaltma:

İletişimdeki atlama sayısını minimize etmeye çalışır. Daha az atlama, daha düşük gecikme süresi ve daha enerji verimli iletişim anlamına gelir.

Amaç Fonksiyonları, RPL'nin esnek bir protokol olmasını sağlar. Bu, farklı IoT uygulamalarının özel gereksinimlerini karşılamak için Amaç Fonksiyonlarının uyarlanabilir olduğu anlamına gelir.

Amaç Fonksiyonları (OF), DODAG'ın oluşturulması ve yönlendirme kararlarının alınmasında kritik bir rol oynar. Bir IoT ağı kurulurken, hangi Amaç Fonksiyonunun seçildiği, DODAG'ın nasıl oluşturulduğunu ve düğümlerin nasıl yönlendirme kararları aldığını etkiler. Bu bağlantıyı daha ayrıntılı bir şekilde açıklayan bazı ana noktalar şunlardır:

DODAG Oluşturma: Amaç Fonksiyonları, DODAG'ın nasıl oluşturulacağını ve yönlendirme ağacının nasıl bir şekilde yapılacağını belirler. Örneğin, enerji tasarrufu odaklı bir Amaç Fonksiyonu seçildiğinde, DODAG, enerji tüketimini minimize etmeye yönelik olarak düşük güçlü düğümler arasında daha az enerji harcayacak yolları tercih edebilir.

Yönlendirme Kararları: Amaç Fonksiyonları, düğümlerin yönlendirme kararlarını belirlerken kullanılır. Örneğin, veri akışını maksimize etme odaklı bir Amaç Fonksiyonu, daha yüksek veri iletim hızına sahip yolları tercih edebilir ve bu da DODAG'ın yapısını etkileyebilir.

DODAG'ın Topolojisi: Amaç Fonksiyonları, DODAG'ın topolojisi üzerinde etkili olabilir. Özellikle hedef düğümlerin yerleşimi, düğüm yoğunluğu ve ağın fiziksel özellikleri, Amaç Fonksiyonlarının DODAG'ın yapısını şekillendirmesine yardımcı olur.

Sonuç olarak, Amaç Fonksiyonları ve DODAG, RPL protokolünün temel bileşenleridir ve IoT ağlarının yönlendirme işlemlerini düzenler. Amaç Fonksiyonları, DODAG'ın nasıl oluşturulacağını ve düğümlerin nasıl yönlendirme kararları alacağını belirler, böylece IoT ağlarının ihtiyaçlarına uygun şekilde çalışmasını sağlar.

3.3.7. OF-0 (Objective Function 0)

Amaç Fonksiyonu-0, RPL protokolü içinde kullanılan bir Amaç Fonksiyonu'dur. Amaç Fonksiyonları, bu tür ağlarda düğümler arasındaki iletişimi optimize etmek için kullanılır ve belirli hedeflere odaklanarak yönlendirme kararları alır.

OF-0, RPL'nin standartlaştırılmış Amaç Fonksiyonlarından biridir ve özellikle enerji tüketimini minimize etmeye odaklanır. Aşağıda OF-0'un ana özelliklerini ve işlevlerini açıklayan bilgiler bulunmaktadır.

OF-0'un Özellikleri ve İşlevleri:

Enerji Tasarrufu: OF-0, IoT ağlarında bulunan düşük güçlü düğümlerin enerji tüketimini minimize etmeyi amaçlar. Bu, IoT cihazlarının pil ömrünü uzatmaya yardımcı olur. OF-0, enerji tasarrufunu öncelikli olarak düşünerek düğümler arasındaki yönlendirme kararlarını alır.

Düğüm Seçimi: OF-0, yönlendirme kararlarında en düşük enerji tüketimine sahip düğümleri seçmeye çalışır. Bu, düğümlerin iletişim sırasında daha az enerji harcamasını ve bu sayede pil ömürlerini uzatmasını hedefler.

Düşük Gecikme: OF-0, enerji tasarrufu odaklı olmasına rağmen iletişim gecikmesini de dikkate alır. En düşük enerji tüketimine sahip yol seçimi yaparken, iletişim gecikmesinin kabul edilebilir düzeyde olduğundan emin olmaya çalışır.

Yapılandırılabilirlik: OF-0, ağ yöneticilerine belirli parametreleri ve tercihleri yapılandırma esnekliği sunar. Bu, OF-0'un belirli bir IoT uygulamasının veya ağın gereksinimlerine daha iyi uyum sağlayabilmesini sağlar.

OF-0, enerji tasarrufu ve düşük güçlü düğümlerin kullanıldığı IoT ağlarının gereksinimlerini karşılamak için tasarlanmıştır. Bu Amaç Fonksiyonu, ağ yönlendirme tablosunun oluşturulmasında ve düğümler arasındaki iletişimin yönlendirilmesinde kullanılır. OF-0, düğümlerin enerji kaynaklarını daha etkili bir şekilde kullanmalarına yardımcı olarak ağın daha uzun süre sorunsuz çalışmasını sağlayabilir.

OF-0, RPL protokolü içindeki sadece bir Amaç Fonksiyonu seçeneğidir. Belirli bir IoT uygulamasının veya ağın gereksinimlerine bağlı olarak farklı Amaç Fonksiyonları seçilebilir ve OF-0'un yerine farklı bir OF tercih edilebilir. Bu, IoT ağlarının özel gereksinimlerini ve hedeflerini karşılamak için protokolün esnekliğini gösterir.

3.3.8. OF-1 (MRHOF- Minimum Rank with Hysteresis Objective Function)

Amaç Fonksiyonu-1 (OF-1) olarak bilinen MRHOF, RPL protokolü içinde kullanılan bir Amaç Fonksiyonu'dur. RPL, enerji tasarrufu gerektiren IoT (Nesnelerin İnterneti) ve sensör ağları için tasarlanmış bir yönlendirme protokolüdür. Amaç Fonksiyonları, bu tür ağlarda düğümler arasındaki iletişimi optimize etmek için kullanılır ve belirli hedeflere odaklanarak yönlendirme kararları alır.

MRHOF, enerji tasarrufu ve düğümler arasındaki yönlendirme kararlarını optimize etme amacı güden bir Amaç Fonksiyonudur. İşte MRHOF'un ana özellikleri ve işlevleri:

- ✓ Minimum Rank (Minimum Sıra): MRHOF, her düğümün DODAG (Destination-Oriented Directed Acyclic Graph) yapısındaki sırasını (rank) hesaplar. Rank, bir düğümün DODAG yapısındaki hiyerarşik konumunu temsil eder. Daha yüksek sıralı düğümler, daha düşük sıralı düğümlere yönlendirme yaparlar. MRHOF, minimum rankı elde etmeyi hedefler.
- ✓ Hysteresis (Geçişkenlik): MRHOF, düğümler arasındaki yönlendirme kararlarını hızlı bir şekilde değiştirmekten kaçınmak için geçişkenlik veya histeresi kullanır. Bu, ağdaki hızlı değişikliklerin etkisini azaltır ve kararların istikrarlı olmasını sağlar.
- ✓ Enerji Tasarrufu: MRHOF, enerji tasarrufu odaklıdır ve düşük enerji tüketimine sahip yolları tercih eder. Bu, IoT cihazlarının pil ömrünü uzatmaya yardımcı olur.
- ✓ Hedefe En Kısa Yol: MRHOF, hedef düğüme en kısa yolu seçmeye çalışır. Bu, iletişim gecikmesini minimize eder ve veri iletimini hızlandırır.

MRHOF, IoT ağlarının yönlendirilmesi için enerji tasarrufu ve düşük gecikme gibi önemli özellikleri dikkate alır. Aynı zamanda düğümler arasındaki yönlendirme kararlarının kararsızlık veya hızlı değişikliklerden etkilenmesini azaltmaya çalışır. Bu, IoT cihazları için pil ömrünün uzun olması ve veri iletiminin düşük gecikme ile yapılabilmesi için önemlidir.

MRHOF, RPL protokolünün bir bileşeni olarak kullanılır ve belirli bir IoT uygulamasının veya ağın gereksinimlerine bağlı olarak tercih edilebilir veya yapılandırılabilir. Amaç Fonksiyonları, RPL protokolünün esnekliğini sağlar ve farklı senaryolara uyum sağlamasına yardımcı olur.

3.3.9. ETX (Expected Transmission Count)

ETX, RPL protokolünün bir başka amaç fonksiyonu olup, bağlantı kalitesini ve güvenilirliği optimize etmeyi amaçlar.

ETX, düğümler arasındaki iletişim kalitesini ölçer ve daha güvenilir bağlantıları tercih eder. Daha düşük ETX değerleri, daha güvenilir bağlantıları temsil eder.

ETX, veri iletimindeki başarısızlık oranını ve paket kayıplarını azaltmayı hedefler.

Bağlantı: MRHOF ve ETX, RPL protokolünün yönlendirme kararlarını farklı açılardan optimize etmeyi amaçlar. İşte bu iki amaç fonksiyonu arasındaki temel bağlantılar:

Enerji Verimliliği: Hem MRHOF hem de ETX, enerji tasarrufunu önemser, ancak farklı yollarla. MRHOF, enerji tasarrufu için düşük ranklı yolları tercih ederken, ETX daha güvenilir bağlantıları seçmeye odaklanır. Daha güvenilir bağlantılar, iletişimde daha az yeniden deneme gerektirir, bu da enerji tasarrufuna katkı sağlayabilir.

Gecikme ve Güvenilirlik: MRHOF, düşük gecikme süresini hedeflerken, ETX daha yüksek güvenilirliği amaçlar. Bu, ağın kullanım amacına ve uygulamaya bağlı olarak tercih edilecek amaç fonksiyonunu belirler.

Ağ İstikrarı: MRHOF, histeresi kullanarak yönlendirme kararlarını daha istikrarlı hale getirirken, ETX daha fazla güvenilirliği sağlamak için güçlü bağlantıları tercih eder.

Sonuç olarak, MRHOF ve ETX, RPL protokolü içinde kullanılan farklı amaç fonksiyonlarıdır ve ağın gereksinimlerine ve uygulama senaryolarına bağlı olarak tercih edilirler. Her iki amaç fonksiyonu da IoT ağlarının yönlendirmesini optimize etmeye yardımcı olur, ancak farklı önceliklere sahiptirler.

4. LİTERATÜR İNCELEMESİ

IEEEExplore, Google Scholar, Springer, Scopus gibi uluslararası veri tabanlarının yanı sıra YÖK Ulusal Tez Merkezi ve TÜBİTAK Dergipark gibi ulusal veri tabanları ile birlikte çevrimiçi olan akademik kaynaklar taranarak incelenmiştir.

Dehghan 2020 yılında yayınladığı “Dağıtık 6TiSCH Ağlarında Kaynak Yönetimi” başlıklı tez çalışmasında Java temelli kontrol mekanizması geliştirilmiş ve IETF 6TiSCH protokolü için bir eklenti sunulmuştur. Bu önerilen mekanizma, ağ geçitlerinin trafiğine bağlı olarak frekans kanallarını dinamik olarak tahsis etmektedir. Bu yaklaşım, hem yakındaki 6TiSCH ağları arasındaki çakışmaları azaltmakta hem de veri iletim başarısını artırmaktadır. (Dehghan, 2020).

Park 2020 yılında yayınladığı “Distributed Wireless Resource Management in the Internet of Things” isimli akademik çalışmasında önerilen öğrenme çerçevesinin, sınırlı bellek ve gözlem yeteneklerine sahip IoT cihazları için kritik mesajların sayısını öğrenmek için etkili olduğu gösterilmiştir. Sonuçlar, öğrenme çerçevesinin IoT cihazlarının bellek boyutuna ve gözlem yeteneğine bağlı olarak performans gösterdiğini ve bu çerçevenin büyük ölçekli bir IoT'ta düşük gecikmeli iletimi gerçekleştirebileceğini göstermektedir. Ardından, IoT cihazlarının kısmi bilgiye sahip olduğu durumda kaynak blokları ile cihazlar arasındaki ilişki sorunu incelenmiştir.(Park, 2020).

Amini vd. “Performance Evaluation of Orchestra Scheduling in Time-Slotted Channel Hopping Networks” isimli bildirisinde RPL (Low Power ve Lossy Ağlar için Yönlendirme Protokolleri) ile TSCH (Zaman Dilimli Kanal Atlama) ve Orchestra (TSCH programı oluşturmak için otomatik bir yöntem) kullanan ağların performansını incelemiştir. Ağı stabil bir RPL ağacına geri dönmek için ne kadar süre gerektiğini ve enerji tüketimi açısından ne gibi etkileri olduğunu incelemiştir. Analizlerinde, RPL ağın geçici bir durumda olduğu durumlarda, örneğin bir düğümün ağı terk ettiği durumda, Orchestra'nın enerji tüketimini azalttığını göstermektedir. Ayrıca, Orchestra kullanmadan geçici bir durumda enerji tüketimini hesaplayarak ardından her iki sonucu karşılaştırmıştır. Orchestra'nın Orchestra kullanmamaya göre enerji tüketimini üçte bir oranında azalttığını ifade etmektedirler. (Amini vd., 2020).

Liu vd. “Resource Allocation in Wireless Powered IoT Networks” isimli makalesinde, kablosuz enerjiyle çalışan IoT ağlarının yukarı yönlü iletimi için etkili kaynak

tahsisinin incelendiği yapılmaktadır. LoRa teknolojisi, IoT ağı için bir örnek olarak benimsenmiştir, ancak bu çalışma diğer iletişim teknolojileri için de uygun olarak kabul edilmektedir. Sınırlı kaynaklar, spektrum ve enerji kaynakları gibi, büyük bir kullanıcı kitlesi arasında tahsis edilmelidir ve bu zorluğun üstesinden gelmek için kullanıcılar önce kullanılabilir kanallara gruplanmaktadır. Daha sonra, aynı kanala gruplanmış kullanıcılar için ağın verimliliğini artırmak için güç tahsisi araştırılmaktadır. (Liu vd.,2019).

Chen vd. “Federated Learning over Wireless IoT Networks with Optimized Communication and Resources” isimli makalesinde, Federated Learning (FL), işbirlikçi öğrenme tekniklerinin bir paradigması olarak, iletişim verimliliği ve gelişmiş veri gizliliği avantajlarıyla giderek artan bir araştırma ilgisi kazanmıştır. Kayıplı iletişim kanalları ve sınırlı iletişim kaynakları (örneğin bant genişliği ve güç) nedeniyle kablosuz sistemler üzerinde hızlı tepki veren ve doğru FL yöntemlerini araştırmak ilgi çekicidir. Bu nedenle, kablosuz Nesnelerin İnterneti (IoT) ağlarında FL için iletişim verimliliği ve kaynakları birlikte optimize etme sorununu araştırıyoruz. Karmaşıklığı azaltmak için genel optimizasyon sorununu iki alt soruna, yani istemci planlama sorununa ve kaynak tahsis sorununa böleriz. Kablosuz IoT ağlarında FL için iletişim maliyetlerini azaltmak için, eski yerel model parametrelerini tekrar kullanarak yeni bir istemci planlama politikası önerilmektedir. Ağlar üzerinde başarılı bilgi alışverişini en üst düzeye çıkarmak için, güç değişkenleri, bant genişliği değişkenleri ve iletim göstergeleri de dahil olmak üzere değişkenleri ayıran Lagrange çarpanı yöntemi ilk olarak kullanılır. Ardından, güç ve bant genişliği tahsisi için lineer bir arama tabanlı bir yöntem geliştirilmiştir. Uygun hiper-parametreler verildiğinde, önerilen iletişim verimli federated learning (CEFL) çerçevesinin güçlü bir lineer hızda yakınsadığını gösteriyoruz. Kapsamlı deneyler yoluyla, önerilen CEFL çerçevesinin, eşit kaynak tahsisi ile temel bir FL yaklaşımına kıyasla kablosuz IoT ağları üzerinde hem iletişim verimliliğini hem de eğitim kaybı ve test doğruluğunun öğrenme performansını önemli ölçüde artırdığı ortaya çıkmaktadır. (Chen vd., 2021).

Madan vd. tarafından ifade edilen bir konsept, Uzun Vadeli Evrim-Gelişmiş (LTEA) kablosuz ağlarında, baz istasyonlarının farklı iletim güçlerini kullanarak kullanıcılar arasındaki trafik yoğunluğunu yönetmeyi amaçlamaktadır. Bu tür LTE ağlarında, tek tip güç kullanımı nedeniyle yüksek girişim ve karasızlık potansiyeli bulunmaktadır. Bu sorunu aşmak için heterojen ağlarda önerilen çözüm, ilişkilendirme ve kaynak bölümleme teknikleridir. Önerilen tekniklerin uygulanması sonucunda, farklı baz istasyonlarının konumlandırılmasıyla hücre boyutu küçültülerek spektral verimlilik artırılmıştır. Bu tezdeki

çalışma da, özellikle kaynak bölümlleme yaklaşımına odaklanmıştır, bu sayede kaynakları farklı istasyonlara atayarak ağ verimliliğini artırmıştır. Her iki araştırma da, bölümlleme yöntemiyle elde edilen kaliteye odaklanarak benzer bir yaklaşımı benimsemiştir. (Madan vd., 2010).

Ahmed vd., LTE ağlarında girişimi en aza indirgeyerek hizmet kalitesini artırmak amacıyla femto hücrelerinin rolünü incelemişlerdir. Bu bağlamda, femto hücre ağında hibrit erişim kontrolü için Dinamik Kaynak Tahsis Algoritması (Dynamic Resource Allocation Management Algorithm-DRAMA) önerilmiştir. Femtocell erişim noktası olarak bilinen Femtocell ya da FAP, heterojen bir ağdaki kapsama ve kapasite boşluklarını doldurmak için umut verici bir teknoloji olarak öne çıkmaktadır. Bu çerçevede, FAP'ler rastgele sayıda Homojen Poisson Nokta Dağılımı (Poisson Point Distribution-PPD) kullanarak alanda dağıtılmaktadır. Bu sayede, Mobil Ana Baz İstasyon (MBS), kullanıcı tıkanıklığını ve sinyalleme ek yükünü azaltmak amacıyla hareketliliği FAP'e yönlendirerek kullanılmayan kaynakları değerlendirmekte ve mobil kullanıcılar için üstün bir hizmet seviyesi sağlamaktadır. Bu çalışmaların temelini oluşturan dinamik kaynak tahsisi yaklaşımı, tezde de benzer bir şekilde yoğunluğu az olan ağdan kaynakların alınarak yoğunluğu fazla olan ağa aktarılmasını içermektedir. Bu yöntemle, alandaki kanal dağılımı kullanılarak optimizasyon sağlanmaktadır. (Ahmed vd., 2014).

Chen vd. nesil mobil ağlarda (5G) yüksek veri trafiğini karşılamak için küçük baz istasyonlarının (SBS) dağıtım yoğunluğunu artırmak, temel bir yöntemdir. Ancak, böylesine ultra yoğun ağlarda (UDN), birçok SBS'nin kullanılması ping-pong handovers (HO) denilen durumları beraberinde getirebilir; bu da gecikmeyi ve HO başarısızlığını artırabilir. Ayrıca, 5G'deki kontrol ve veri sinyalleşmesinin ayrılmasından dolayı HO prosedürü her iki katmanda da gerçekleştirilmelidir. Bu makalede, hem HO başarısızlıklarını hem de ping-pong HO's'yi en aza indirmek amacıyla bir SDN tabanlı akıllı dinamik HO parametre optimizasyon stratejisi tanıtılmaktadır. Önerilen stratejinin amacı, HO başarısızlık oranını ve gereksiz HO'yu (yani ping-pong HO) azaltarak kullanıcı ekipmanının (UE) BS'lerin yoğun dağıtımından elde ettiği avantajlardan tam olarak yararlanmasını sağlamaktır. Simülasyon sonuçları, bu makalede önerilen yöntemin ping-pong etkisini etkili bir şekilde bastırdığını ve tüm araştırılan senaryolarda bunu düşük seviyede tuttuğunu göstermektedir. Ayrıca, diğer algoritmalarla karşılaştırıldığında, HO başarısızlık oranının önemli ölçüde azaldığı ve UE'nin verimliliğinin özellikle yüksek BS yoğunluğunda büyük ölçüde arttığı görülmüştür. Bu nedenle, yoğun BS dağıtımının faydaları korunmuştur. (Chen vd., 2021).

Federico vd. tarafından Endüstriyel Kablosuz Sensör Ağları (IWSNs), modern endüstrinin doğası ve talepleri nedeniyle Kablosuz Sensör Ağlarının (WSNs) evriminde bir sonraki adım haline gelmiştir. Bu tür ağlarla, farklı özelliklere sahip trafik kaynaklarını eş zamanlı olarak destekleyen esnek ve ölçeklenebilir mimariler oluşturmak mümkündür.

Yazılım Tanımlı Ağlar'ın (SDNs) kullanımı, ağ üzerinde kontrol imkanlarını genişletir ve endüstriyel düzeyde dağıtımını mümkün kılar. Düğümler arasında ve denetleyici ile değişen sinyal trafiği yoğundur ve bu trafiğin veri trafiğiyle aynı kanalı kullanması gerekmektedir. Bu zorluk, trafiği akışlara bölmek ve MAC (Orta Erişim Kontrolü) seviyesinde doğru bir takvimleme ile, "slices" olarak adlandırılan yöntem ile başa çıkılabilir. (Federico vd., 2022).

Bu makalede, SDN denetleyicisine bir trafik yöneticisi entegre etmeyi, farklı akışlara göre farklı rotalar atayan bir yönlendirme sürecini ve Zaman Dilimli Kanal Atlama (TSCH) Takvimleyici'sini tanıtmayı önermektedir. Ayrıca, TSCH, SDN-WISE çerçevesine entegre edilmiş ve bu protokol, TSCH takvimini göndermek üzere değiştirilmiştir. Bu unsurlar, denetleyiciden gelen tek bir paket aracılığıyla düğümlere gönderilecek trafiği planlama ve segmentleme konusunda birlikte sorumludur ve performansları simülasyon ve bir test yatağı üzerinden değerlendirilmiştir.

Elde edilen sonuçlar, yönlendirme süreci ve TSCH Takvimleyici'nin birlikte kullanımının esneklik, adaptabilite ve belirliliği artırdığını göstermektedir. Bu, farklı hizmet kalitesi gereksinimlerine sahip akışlar arasında dilimleme oluşturmayı mümkün kılarak, QoS özelliklerini garanti etmeye, en yüksek öncelikli akış için Paket Teslim Oranını (PDR) artırmaya, Deadline Miss Oranını (DMR) sürdürmeye ve ağ ömrünü artırmaya yardımcı olmaktadır. (Santos vd., 2021).

Giacomo vd. Kablosuz Sensör Ağları (WSN'ler), geleceğin Nesnelerin İnterneti'nin (IoT) temel yapı taşlarından biri olarak görmektedirler. Yapılan çalışmaları ile 6LoWPAN standardının tanıtılması sayesinde, WSN'ler IP ağlarına sorunsuz bir şekilde entegre edilebilir. Önceki çalışmalarında, SD-6LoWPAN adını verdikleri bir mimari tanımladılar; bu mimari özellikle 6LoWPAN'lerde yazılım tanımlı ağları etkinleştirmek için tasarlanmıştır. Son çalışmada ise (Tanganelli vd., 2019) önerilen SDN çözümünün Contiki işletim sistemine sahip gerçek kısıtlı cihazlar üzerine uygulanmasına odaklanılmıştır. Contiki ağ yığını genişleterek ortaya çıkan temel zorlukları ve bunların nasıl ele alındığı açıklanmıştır, böylece SD-6LoWPAN'ın çalışan bir prototipini sunabilmektedirler. Sonrasında, bu prototip gerçek bir test ortamında bir dizi deney çalıştırılarak doğrulanmıştır.

Baddeley vd. 2017 yılında yayınladıkları bir makalede yüksek maliyetli SDN mimarisini kısıtlı, düşük güçlü kablosuz ağlara uyarlamakla ilgili bir zorluğa ele alınmaktadır. Son zamanlarda IEEE 802.15.4 ağları için protokollerdeki merkezileşme eğilimlerine dayanan düşük maliyetli bir SDN mimarisi olan μ SDN tanıtılmaktadır ve genellikle IPv6 olmayan ağları ele alan son çalışmalarda tanıtılan kavramları genişletmektedir. μ SDN, IPv6 ağlarıyla uyumluluk, mevcut dağıtık yönlendirme protokollerine uyumluluk gibi ek optimizasyon tekniklerini uygular. Ardından, μ SDN'yi kullanarak SDN trafiğinin ağ performansına etkisini değerlendirilir ve yüksek öncelikli akışlar için Geleneksel Yaklaşımlar üzerinde Hizmet Kalitesi'ni (QoS) artırabileceği bir senaryo ele alınmaktadır. (Baddeley vd., 2017).

Belavagi ve Muniyal (2020), RPL protokolü ile oluşturulan sistemde farklı düğüm sayılarını ve zafiyetli düğüm oranlarını ele almışlardır.[29] Çalışmalarında, 10, 40 ve 100 düğümlü sistemlerde %10, %20 ve %30 oranında zafiyetli düğümler eklemişlerdir. Bu yapıları kullanarak elde ettikleri verilerle, şebekenin davranışını analiz etmişlerdir. Ayrıca, makine öğrenimi algoritmalarını kullanarak ağ paketlerinin yanı sıra enerji tüketimi, GP ve YP oranlarını da dikkate almışlardır. Bu çalışma, şebeke performansını belirlemede kapsamlı bir yaklaşım sunmaktadır. (Belavagi & Muniyal, 2020).

Aydın ve diğerleri tarafından 2022 yılında yapılan bir çalışmada, IETF 6TiSCH ağlarında RPL tabanlı yönlendirme için iki farklı amaç fonksiyonu optimizasyonu önerilmektedir. Her iki yöntemin de temel hedefi, ebeveyn değiştirme sürecinde ortaya çıkabilecek ek kaynak ayırma maliyetini en aza indirmektir. Bu amaçla, düğümlerin ebeveyn değiştirme eşik değeri dinamik olarak ayarlanmakta, yüksek performanslı yollara sahip düğümlerin ebeveyn olarak tercih edilmesi ve bu kararın sürekliliğinin sağlanması amaçlanmaktadır. İlk yöntemde, yazarlar ebeveyn değiştirme eşik değerini, komşu düğüm sayısına bağlı olarak güncellemektedir. Diğer bir yöntemde ise bu eşik değeri, komşu düğüme tahsis edilmiş iletim yapabilecek hücre sayısını dikkate alarak belirlenmektedir. Yazarlar, bu önerilen yöntemleri test etmek için Cooja benzetim ortamında gerçek zamanlı çalışan Contiki-NG işletim sistemine sahip MSP-EXP430F5438 mikrodenetleyici tabanlı duyarga düğümlerinden oluşan 6TiSCH ağını kullanmış ve önerilen yöntemlerin başarımını RPL protokolünün varsayılan amaç fonksiyonu olan MRHOF ile karşılaştırmıştır. Çalışma sonuçlarına göre, her iki önerilen yöntem de paket iletim oranında MRHOF'a göre önemli

bir iyileşme sağlamaktadır. Ayrıca, paket başına ortalama gecikme süresinde ve ebeveyn değıştirme sayılarında anlamlı düşüşler gözlemlenmiştir.(Aydın vd., 2022).



5. YAPILAN ÇALIŞMA

RPL Protokolü, düşük güç tüketimi ve kısıtlı veri hızı özelliklerine sahip ağlarda yönlendirme işlemini yönetmek için kullanılan bir protokoldür. RPL, ağ topolojisini ağaç yapısı benzeri bir veri yapısı olan DODAG ile temsil eder. Bu yapıda, her düğümün bir kök düğümü ve çocukları vardır ve düğümler, kök düğümlerine ve altındaki çocuk düğümlere yönlendirme yapmak için kullanılır. DODAG yapısı, düğümler arasındaki iletişim bağlantılarını temsil eden kenarlarla oluşturulur ve kök düğümünden yayınlanan DIO kontrol mesajı ile başlatılır. DODAG yapısı, düşük güç tüketimi ve kaynak kısıtlamaları olan ağlarda verimli yönlendirme sağlar. Düğümler, bu mesajı alırken göndereni üst listelerine ekler ve MRH-OF (Histeresiz Amaç Fonksiyonu ile Minimum Derece) ve OF-0 (Amaç Fonksiyonu Sıfır) gibi amaç fonksiyonlarına bağlı olarak üst düğüme göre sırasını hesaplayarak DODAG'a katılmayı tercih edebilir.[6] RPL protokolünde, her düğümün DODAG üzerindeki konumunu belirleyen rank denilen bir "derece" değeri vardır. Derece değeri, düğümün kök düğümden olan mesafe ölçüsü olarak kullanılır ve DODAG yapısının yönlendirme işlemini etkiler. Düşük derece değeri, daha yakın bir konuma işaret eder ve daha iyi yönlendirme performansı sağlar.

RPL'de, kök düğüm değişimi yapılması esnasında seçilen düğümün derece değeri ile aday olan düğümün derecesinin mukayese edilmesi halihazırda var olan bir durumdur. Benzer derece değerlerine sahip düğümler arasında sürekli olarak değişiklik gerçekleşmemesi için aday düğümün derecesi belirlenmektedir.

Bu tez çalışmasında; aynı coğrafi ortamda bulunan komşu 6TiSCH ağların dengelenmesi hedeflenmiştir. 6TiSCH bünyesinde RPL ile yönlendirme işlemleri yapılmakta fakat düğümler komşularını bilmeden yerleştiklerinden dolayı, oluşturulan ağda kendilerine uygun bir kök düğüm (root) seçimi yaparak kök düğüme katılmaları gerekmektedir. Fakat oluşturulan IOT ağlarda yeni katılan düğümlerin katıldıkları ağdaki düğüm sayısını bilmemeleri ve oluşan orantısız düğüm sayısı sebebiyle ağın verimliliği düşmekte, enerji tüketimi artmaktadır.

Bu kapsamda çözüm yaklaşımı olarak; öncelikle düğümlerin belirli kanallar üzerinden çalışması sağlanarak ağın dengelenmesi hedeflenmektedir. Ağa yeni katılmak isteyen düğümlere, merkezi olan yönetilen bir denetleyici aracılığıyla hangi kanal setini

kullanacakları ve dolayısıyla hangi RPL Sınır yönlendiriciye dahil olacakları bildirildiğinde ağın dengeli ve kararlı bir şekilde çalışması sağlanabilecektir.

5.1. Önerilen Mekanizmalar

Kablosuz Duyarga Ağlar (WSN) çok sayıda küçük boyutlu, düşük maliyetli ve kısa mesafede kablosuz ortam üzerinden haberleşen duyargalardan meydana gelen ağıdır. Bu ağlar sıcaklık, ses, titreşim, basınç, hareket veya kirletici maddeler gibi çevresel koşulları izlemek ve verilerini ağ üzerinden iletmek için işbirliği yaparlar. KDA'ların yönlendirme mekanizmalarında yoğunlukla kök düğüm değiştirme, paket gecikme ve kayıpları yaşanabilmektedir.

RPL ile oluşturulan ortogonal (karşılıklı) iki komşu ağ ve bu ağlarda farklı sayılarda düğümler olduğu bir senaryoda bu ağların düğüm sayısı olarak dengelenmesi için aşağıdaki mekanizma önerilmiştir.

Bu doğrultuda çözüm mekanizması olarak; RPL altyapısı ile geliştirilen algoritma ile düğümlerin doğru ve dinamik bir şekilde yönetimi ve optimizasyonu hedeflenmiştir. Bu kapsamda kurulan iki komşu ağda, düğümleri sayısal olarak dengeleyebilmek için sayılarının eşit veya birbirine yakın olması gerekmektedir. Bunu tespit edebilmek için komşu ağların sahip olduğu düğüm sayısının tespiti gerekmektedir. Ağa yeni katılmak isteyen düğümlerin (Pledge) senkronizasyon öncesinde düğümlerin eşit bir şekilde RPL Sınır yönlendiricilere dağıtılması neticesinde ağın dengelenmesi sağlanabilecektir. Bunun için ağların farklı kanallar üzerinden çalışabilmesi ve ağa dahil olan düğümlerin; daha az yoğunluğa (düğüm sayısı az olan) sahip olan ağın kanal setine geçebilmeleri hedeflenmektedir.

"Pledge", ingilizce bir terim olup genelde "söz verme, taahhüt" gibi anlamlara gelir. Ancak IEEE 802.15.4e standartındaki 6TiSCH protokolünde "pledge" terimi, ağa yeni katılan bir cihazı tanımlamak için kullanılır.[40] Yeni katılan cihaz, ağa katılma talebini ileten ve ağ tarafından kabul edilip edilmediğini öğrenmek için bekleyen bir "pledge" olarak adlandırılır. Bu cihaz, ağ tarafından kabul edilirse "joined" (katılmış) durumuna geçer ve ağa katılmış olur. Bu terim, ağdaki düğümlerin katılma ve ayrılma durumlarını tanımlamak için kullanılır.

Bunun için öncelikle iki komşu 6TiSCH ağı kurulacak, bu ağların farklı kanallar üzerinden çalışması sağlanacaktır. Kök düğüm ve ona bağlı olan diğer istemci düğümler aynı kanallar üzerinden çalışacaktır. Bunu sağlamak için komşu 6TiSCH ağlarının farklı kanal setleri üzerinden kanal atamaları yapılacaktır. Akabinde ise ağı dahil olan yeni düğümlerin düğüm sayısı az olana katılabilmesi sağlanarak ağ dengelenmesi gerçekleşmiş olacaktır.

5.1.1. Komşu 6TiSCH Ağının Kurulumu

Kablosuz Duyarga Ağları (KDA), kısıtlı iletim ve sınırlı iletim mesafeli düğümlerden meydana gelmektedir.[41] KDA'lardaki temel hedef düğümlerin çakışmasını azaltmaktır. Kurulan komşu IoT ağlarında performansı yüksek tutabilmek, bu bahse konu çakışmaları minimize etmekten geçmektedir. Bunun için ağların dengeli bir şekilde kurulması gerekmektedir. Ağdaki düğüm sayısı diğer ağı göre daha fazla ise çakışmalar artacak, buna bağlı olarak ağın performansı olumsuz etkilenecektir.

6TiSCH ağlarda C (t, choff) mekanizması ile t zaman dilimi ve kanal ofsetleri belirlenmektedir. 6TiSCH ağda 11-26 arasında belirlenen farklı frekans kanalları zaman dilimleri içinde kullanılabilir. Ortak olarak kullanılan bu mevcut frekans kanallarının optimizasyonu büyük önem arz etmektedir, çünkü efektif kullanılmayan kanallar sebebiyle ağda paket kaybı ve bu paketlerin yeniden gönderilmesine bağlı enerji tüketimi olabilmektedir.[42]

5.1.2. Kurulum Mekanizması

6TiSCH ağlarda komşu IoT ağlara farklı prefix ve farklı tunslip ataması yaparak başlangıçta birbirinden izole iki komşu ağ kurulacaktır. RPL sınır yönlendiricilerin farklı kanal setleri üzerinden çalışması sağlanarak, kendilerine senkron olan düğümlerinde aynı kanal seti üzerinden çalışabilmesi amaçlanmıştır. Bu sayede, ağdaki çakışma ve paket kayıpları minimize edilerek ağın performansı artırılabilir. Bunun için aşağıdaki algoritma kullanılarak komşu ağlar kurulmuştur.

Öncelikli olarak aynı coğrafi ortamda bulunan komşu ağlar kurmak için Ubuntu İşletim Sistemi üzerinde Contiki İşletim Sistemi içinde çalıştırılan Cooja simülatöründen yararlanılmıştır. 6tisch-ng klasörü içinden erişilen tools-cooja alt dizininden simülatöre erişilebilmektedir. Oluşturulan similasyondan *Mote* tipi olarak MSP-EXP430F5438 mikrodenetleyici düğümleri kullanılmıştır. İki adet RPL Sınır Yönlendirici (RPL Sınır

Yönlendirici) düğüm ve Udp-Client olarak farklı sayıda düğümler (10-20-30 adet düğüm) eklenmiştir. Düğümler düzlem üzerine random olarak yerleştirilmiştir. RPL Sınır Yönlendirici olan kök düğüme (RPL Sınır Yönlendirici-1) Serial Socket (SERVER) olarak 60001 portu, diğer kök düğüme (RPL Sınır Yönlendirici-2) ise 60002 portu seçilmiştir. Tools/serial-io klasörü içinde aşağıdaki komutlar koşularak fd00 ve fc00 olarak farklı prefix atamaları yapılır.

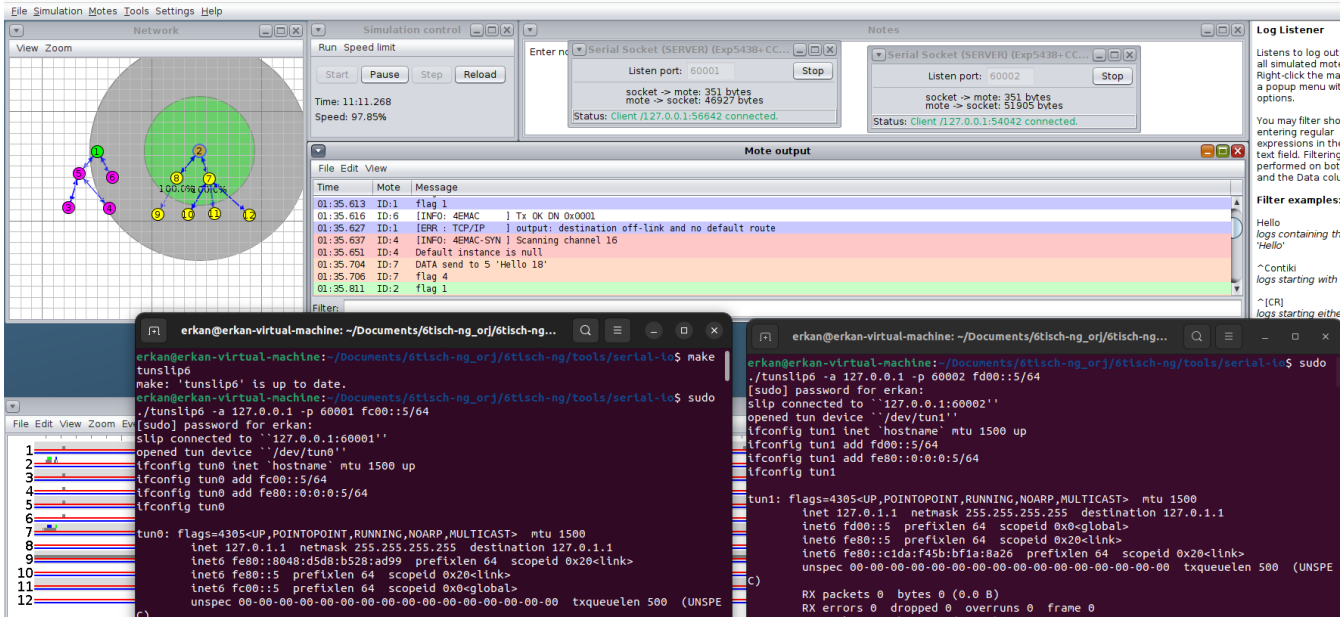
- make tunslip6
- sudo ./tunslip6 -a 127.0.0.1 -p 60001 fd00::5/64
- sudo ./tunslip6 -a 127.0.0.1 -p 60002 fc00::5/64

RPL Sınır Yönlendiricilerin farklı kanallar üzerinden çalışabilmesi için aşağıdaki gibi sınır yönlendiricilerin konfigürasyon dosyalarına (project-conf.h) kanal setleri statik olarak atanmıştır. Bu sayede 6TiSCH içinde default olarak atanan 11-26 arasındaki kanallar { 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26 } ikiye ayrılarak 8-8 olarak eşit bir şekilde komşu ağlara tahsis edilecektir.

- RPL Sınır Yönlendirici-1


```
#define FOURE_HOPPING_SEQUENCE_8 (uint8_t[])
{ 11, 12, 13, 14, 15, 16, 17, 18 }
```
- RPL Sınır Yönlendirici-2


```
#define FOURE_HOPPING_SEQUENCE_8 (uint8_t[])
{ 19, 20, 21, 22, 23, 24, 25, 26 }
```



Şekil 5.1.1. Komşu IoT Kurulumu

5.1.3. Çakışma (İnterferans)

Ağdaki tüm düğümler senkronize bir şekilde çalıştıklarından, iletimler sırasında interferans (çakışma) olabilir. RPL içinde interferans, genellikle ağdaki düğümlerin iletişimini etkileyen çeşitli faktörlerden biridir. İnterferans, kablosuz iletişimde diğer sinyaller veya cihazlar tarafından yayılan elektromanyetik dalgaların, bir cihazın alıcı veya vericisinde istenmeyen bir şekilde etkileşime girmesi sonucu oluşur.

RPL'nin tasarımında, interferans genellikle yönlendirme kararlarını etkileyen bir faktör olarak dikkate alınır. Çünkü ağdaki iletişim kalitesi, cihazların birbirlerini ne kadar etkilediğiyle doğrudan ilişkilidir. Bu nedenle, RPL gibi protokoller, interferansı azaltmaya veya en aza indirmeye yönelik çeşitli stratejiler ve algoritmalar içerebilirler. Bu sebeple interferans etkisini minimize etmek için kanaldan kanala atlarlar ve belirli bir timeslot üzerindeki interferans etkisini azaltırlar. Buradaki amaç, aynı timeslot'un tüm slot çerçevelerinde her zaman kötü kalitedeki aynı kanalı kullanmasını önlemektir. Bu teknik kanal atlama (channel hopping) olarak adlandırılır.

Bu amaçla, düğümler ağın oluşturulmasından bu yana geçen timeslot sayısını sayarlar. Bu değişken Absolute Slot Number (ASN) olarak adlandırılır ve ağı duyuran beacon isimli mesajlarla da duyurulur. Bir düğüm iletim veya alım yapmak zorunda olduğunda, kanal ofseti ve ASN'den yararlanarak kullanması gereken fiziksel kanalı belirler ve aşağıdaki formülü kullanarak bulur. [27]

$$CH = F((ASN + ChannelOffset) \% NumChannels) \quad (2)$$

Formüle göre; CH frekansı ifade eder. Frekans, ASN (Mutlak Sıra Numarası) ve kanal ofsetini mod operatörü ile birlikte formülüne edilir. Bu sebeple Fonksiyon F, kanal ofsetini frekansa dönüştürmek için kullanılır. Çok kanallı iletişim mekanizmasıyla, aynı zaman diliminde birden fazla iletişim gerçekleştirebilmek için farklı kanal ofsetleri kullanılır.

6TiSCH ağ ortamındaki önemli konulardan biri, tüm cihazların aynı anda yeni bir kanal listesine geçmesidir. Bu geçiş, ağın düzgün çalışmasını ve iletişimin kesintisiz olmasını sağlamak için hayati önem taşır. Bu işlem, senkronizasyonun kaybolmaması ve cihazların düşmemesi için özenle planlanmalıdır. Bu amaçla, yeni bir ASN (Adım Sayısı) değeri belirlenir. Bu sayede, ağdaki tüm cihazlar yeni ASN değerini öğrendiklerinde, yeni kanal listesine geçiş yapabilirler. ASN değeri, zamanla artan bir değerdir.

Bu sayede, ağdaki tüm cihazlar yeni bir kanal listesine uyum sağlayabilir ve iletişim kesintisiz devam edebilir.

Komşu ağların kurulum algoritması ise aşağıdaki gibidir:

Algoritma 1. Komşu 6Tisch Ağının Kurulum Algoritması

RPL Sınır Yönlendirici 1'i Başlat:

- IPv6 adresini ayarla: fd00::1/64
- Kanal setini { 11, 12, 13, 14, 15, 16, 17, 18 } olarak ata.
- Yönlendirme tablosunu başlat
- Tunslip6'yı 60001 numaralı bağlantı noktasından başlat

RPL Sınır Yönlendirici 2'yi Başlat:

- IPv6 adresini ayarla: fc00::1/64
- Kanal setini { 19, 20, 21, 22, 23, 24, 25, 26 } olarak ata.
- Yönlendirme tablosunu başlat
- Tunslip6'yı 60002 numaralı bağlantı noktasından başla.

İstemcileri Ekle:

RPL Sınır Yönlendirici 1'in kapsama alanında ise:

- IPv6 adresini ayarla: fd00::2/64
- Varsayılan ağ geçidini ayarla: fd00::1

RPL Sınır Yönlendirici 2'nin kapsama alanında ise:

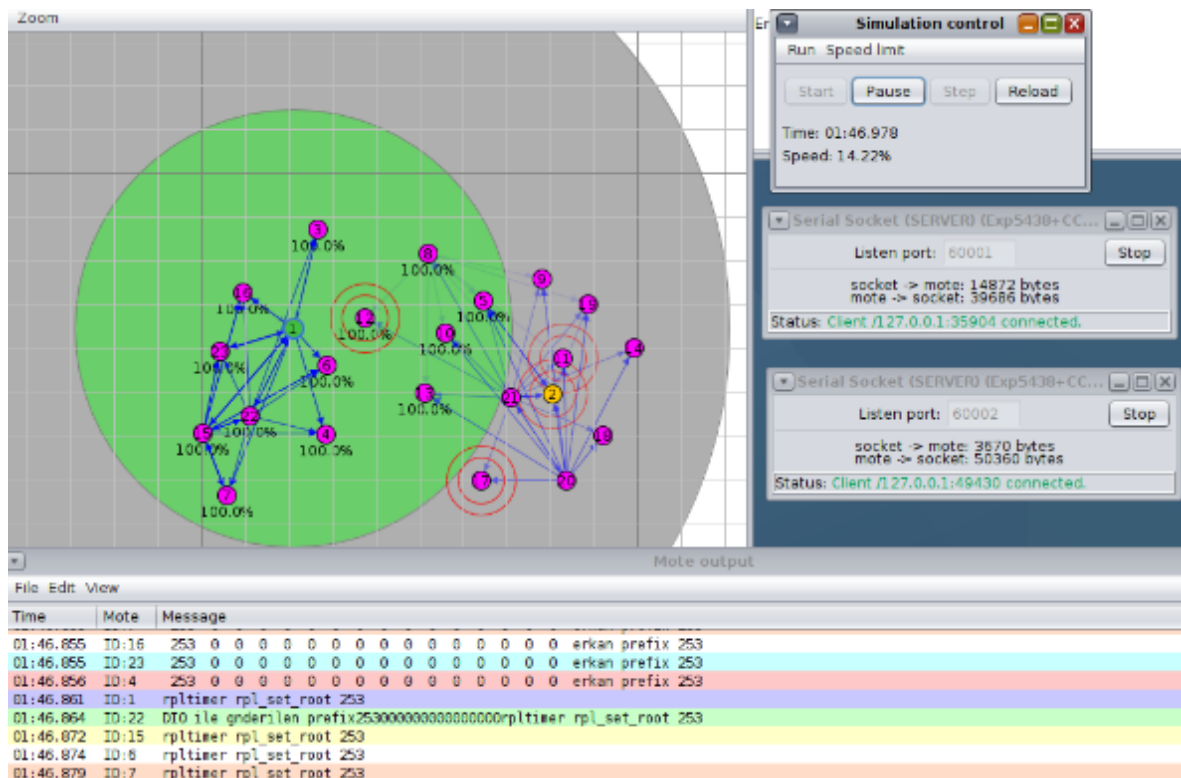
- IPv6 adresini ayarla: fc00::2/64
 - Varsayılan ağ geçidini ayarla: fc00::1
-

5.1.4. Algoritma-1 Sonuçları

RPL Sınır yönlendiricilerinin project-conf.h (konfigürasyon) dosyalarında biri fd00 olarak, diğeri ise fc00 olarak ayarlanmıştır. RPL Sınır Yönlendirici 1 ve onun kapsama alanındaki (yeşil alanda) olan düğümlere fd00 (hexadecimal değeri:253), RPL Sınır

Yönlendirici 2 ve onun kapsama alanındaki düğümlere fc00 (hexadecimal değeri:252) ataması yapılarak düğümlerin farklı prefix'lerde çalışması sağlanmıştır.

RPL Sınır Yönlendirici 1'e { 11, 12, 13, 14, 15, 16, 17, 18 } kanal seti, RPL Sınır Yönlendirici 2' ye de { 19, 20, 21, 22, 23, 24, 25, 26 } kanal seti statik olarak atanmıştır. RPL Sınır Yönlendiricilerin kapsama alanındaki düğümlerde otomatik olarak aynı prefix ve aynı kanal seti ile çalışması sağlanmıştır. Bu sayede iki komşu ağ birbirinden izole edilmiştir.



Şekil 5.1.4. Cooja Simülâtörü ile Kurulan Komşu Ağ Yapısı ve Döğümler

Uygulamanın sonuçlarını Cooja simülatöründe görebilmek için `project.conf` dosyasında `LOG_LEVEL_ERR`, `LOG_LEVEL_INFO`, `LOG_LEVEL_WARN` açılarak logların ekrana düşmesi sağlanmıştır.

Kurulum algoritması sonuçlarına bakıldığında, ağa yeni dahil olmak isteyen düğümler RPL Sınır Yönlendiricilerin kapsama alanı içindeyse ilgili sınır yönlendiriciye dahil olduğunu, onun prefix üzerinden IP aldığını, ilgili sınır yönlendiricinin sahip olduğu kanal setinden bir kanalı kullandığını simülasyon sonuçlarından görülmektedir. Bu sayede birbirinden izole komşu 6TiSCH ağları kurulabilmektedir.

Komşu 6TiSCH ağlar başarılı bir şekilde kurulduktan sonra, ağların dengeli bir şekilde oluşturulmasını sağlamak için (düğüm sayısı olarak eşit veya yakın) ek bir algoritma geliştirilmiştir. Aşağıda detayları verilecek olan algoritma sayesinde ağa yeni katılan düğümler, düğüm yoğunluğu daha az olan RPL Sınır Yönlendiriciye dahil olabilecek ve bu sayede dengeli bir komşu IoT ağı kurarak, ağların optimizasyonu sağlanabilecektir.

5.1.5. Kanal Seçim Mekanizması

Kanal seçimi (Channel selection), TSCH ağlarında önemli bir rol oynar çünkü cihazlar belirli bir zamanda belirli bir kanalda iletişim kurarlar. Yanlış kanal seçimi, iletişim kalitesini düşürebilir ve ağ performansını olumsuz etkileyebilir. Bu nedenle, TSCH'de kanal seçimi genellikle iki aşamalı bir süreç olarak gerçekleştirilir:

Başlangıç Kanalı Seçimi (Initial Channel Selection): Cihazlar ağa katıldıklarında veya ağda değişiklikler olduğunda başlangıç kanalı seçimi yaparlar. Bu genellikle rastgele veya önceden belirlenmiş bir algoritma kullanılarak yapılır.

Adaptif Kanal Seçimi (Adaptive Channel Selection): TSCH ağlarında adaptif kanal seçimi, cihazların çevrelerindeki koşulları sürekli olarak izleyerek ve arka plan gürültü seviyelerini ölçerek en iyi kanalı seçmelerini sağlar. Bu sayede iletişim kalitesi optimize edilir ve ağ performansı artırılır.

Adaptif kanal seçimi genellikle RSSI (Received Signal Strength Indicator - Alınan Sinyal Gücü Göstergesi) metriği üzerinden gerçekleştirilir. Cihazlar çevrelerindeki diğer cihazların sinyal gücünü ölçerek en düşük gürültü seviyesine sahip kanalı seçerler. Bu süreç, iletişim kalitesini sürekli olarak iyileştirir ve ağın performansını optimize eder.

RSSI, bir kablosuz iletişim bağlantısında alınan sinyalin gücünü ölçen bir metriktir. Bu metrik, genellikle dBm (desibel milivat) cinsinden ifade edilir ve alınan sinyalin gücünü belirlemek için kullanılır. RSSI değeri, alıcı cihazın anteninden alınan sinyalin gücünü gösterir. Bu değer ne kadar yüksekse, alınan sinyal o kadar güçlüdür. Öte yandan, RSSI değeri ne kadar düşükse, alınan sinyal o kadar zayıftır. RSSI, genellikle kablosuz iletişim sistemlerinde sinyal kalitesini değerlendirmek ve iletişim performansını optimize etmek için kullanılır.

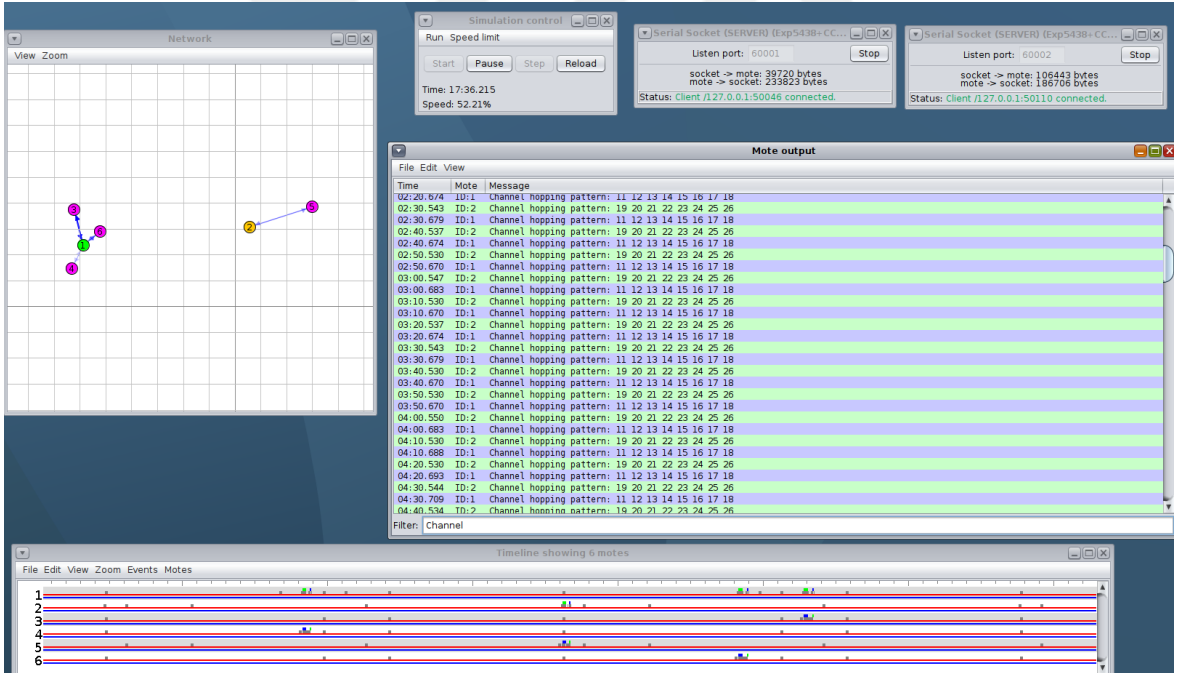
RSSI değeri, fiziksel çevre şartlarına, sinyal engellerine ve uzaklığa bağlı olarak değişebilir. Bu nedenle, RSSI değerleri genellikle iletişim kalitesini anlamak için bir referans

olarak kullanılır. Örneğin, daha yüksek RSSI değerleri genellikle daha iyi bir iletişim kalitesini gösterirken, düşük RSSI değerleri daha zayıf bir iletişimi işaret edebilir.

IEEE 802.15.4 standardı, genellikle 16 adede kadar kanalı (Radio Channel) destekler. Bu kanallar genellikle 2.4 GHz frekans bandında bulunur ve 11-25 aralığında 16 kanaldan birini seçerek ağ üzerinde iletişim kurarlar. Dolayısıyla, bir RPL tabanlı ağda genellikle 16 adet kanal kullanılabilir.

6TiSCH ağlarda, ağa yeni katılan bir düğüm standart olarak 11-26 arasında herhangi bir kanal üzerinden iletişim kurmaktadır.

Yukarıda kurulan komşu 6TiSCH ağında RPL Sınır Yönlendirici 1 ve RPL Sınır Yönlendirici 2 olarak iki adet RPL Sınır Yönlendirici ve bunlara bağlı da farklı sayıda düğümler bulunmaktadır. IoT ağının farklı kanallar üzerinden çalışabilmesi maksadıyla geliştirilen algoritma ile RPL Sınır Yönlendirici-1 11-18 arası kanal setini kullanırken; RPL Sınır Yönlendirici-2 ise 19-26 arası kanal setini kullanmaktadır.



Şekil 5.1.5. Cooja Simülatörü ile Kurulan Komşu Ağ Yapısı ve Atanan Kanallar

5.1.6. Ağ Dengeleme Algoritması

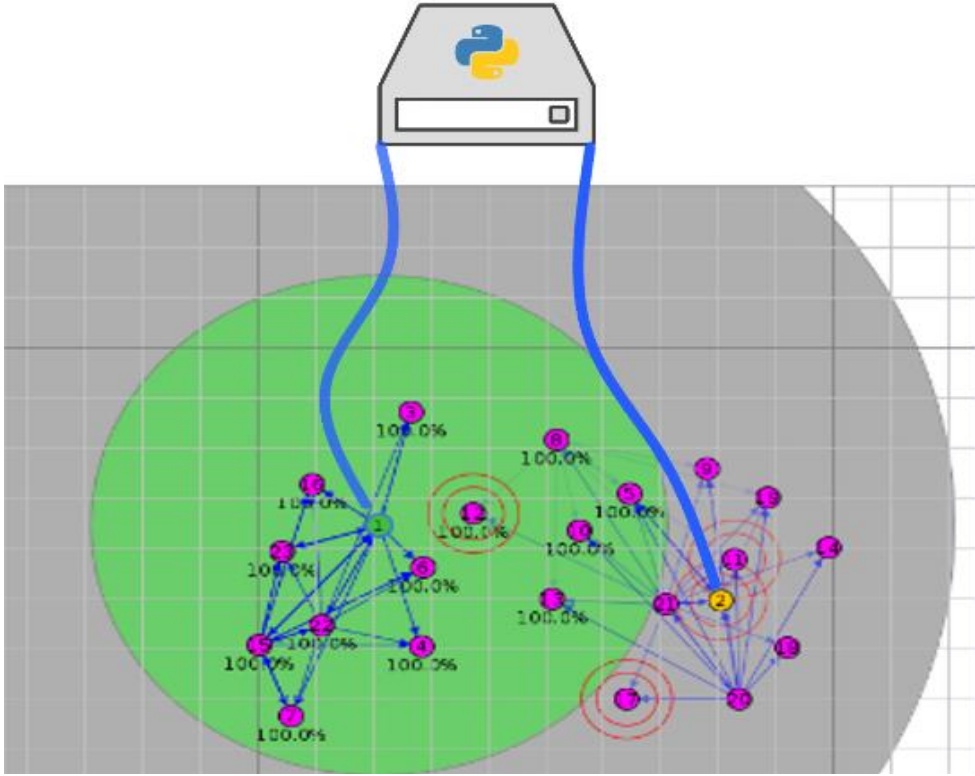
Ağa yeni katılan düğümlerin hangi kanal setini kullanacağını belirlemek için aşağıdaki algoritma önerilmiştir. Python programlama dili ile geliştirilen bir program ile düğümlerin ağa dahil olmadan senkronizasyon öncesinde öncelikle komşuluk tablosundaki bilgiler

periyodik olarak merkezi bir denetleyiciye gönderilecektir. RPL Sınır Yönlendiricilerin düğüm sayıları tespit edilecek, daha sonra ise denetleyici ağı yeni katılan düğüme (Pledge) hangi kanal setini kullanacağını, hangi ağı dahil olacağını hem tespit hemde tayin edebilecektir. Bu algoritmaya göre ağlar dinamik ve dengeli bir şekilde yönetilebilecektir.

Bu çalışma, 6TiSCH ağlarının modellenmesini üç aşamada özetlemektedir. İlk olarak, aynı coğrafi alanda konumlandırılan 6TiSCH duyargaları belirlenir. İkinci aşama, bir Python tabanlı merkezi denetleyici uygulamasının geliştirilmesini içerir. Üçüncü aşama ise, 6TiSCH ağlarındaki ağ geçitlerinin merkezi denetleyici ile ilişkilendirilmesini sağlar.

Ağı katılmak ve diğer duyargalarla iletişim kurmak isteyen her duyarga, başlangıç olarak EB mesajları yayımlar. Bu mesajlar, senkronizasyon, kanal atlama ve zaman dilimi gibi bilgileri içerir. Ayrıca, her 6TiSCH ağı, RPL protokolünü kullanarak duyargalardaki yönlendirme tablolarını dış denetleyiciye gönderir.

İkinci aşama, bir dış denetleyiciyi tanımlamayı içerir ve Python tabanlı bir denetleyici geliştirilir. Bu aşama, ağ geçitlerinin ağı katılmak istediklerinde gerekli mesaj içeriğini merkezi denetleyiciye iletmesini sağlar. Merkezi denetleyici, dinlediği port üzerinden gelen verileri kullanarak düğümlerin eşit bir şekilde ağı katılımını ve senkronizasyonunu sağlar.



Şekil 5.1.6. Denetleyici ile 6TiSCH Ağının Dengelenmesi

Python dili ile yazılan ağın dengelenmesi için düğümlerin dengeli bir şekilde yerleşmesini sağlayan “calculate_pledge_within_network” isimli fonksiyon, bir düğümün ağdaki hangi küme (cluster) içinde yer alması gerektiğini belirlemek için kullanılır.

Fonksiyonun aldığı parametreler:

- dest_addr: Bağlı olduğu hedef adres.
- pledge_ll_addr: Düğümün bağlı olduğu küme indeksi.

Fonksiyon, önce verilen “dest_addr” adresinin son bir byte'ını alır ve bunu integer bir değere çevirir. Bu değer, düğümün hangi küme ile ilgili olduğunu belirlemek için kullanılır.

Fonksiyon aşağıdaki adımları izler:

1. “dest_addr” isimli parametreden son bir byte alınarak “last_byte” isimli değişkenine atanır. Bu sayede hedef adresin son byte'ı elde edilmiş olur.
2. Eğer “last_byte” olarak tanımlanan hedef adresindeki son byte, “br_node_list” isimli liste içinde yoksa, yani o küme henüz oluşturulmamışsa, yeni bir küme oluşturulur.
3. “node_counts” adında bir sözlük oluşturulur. Bu sözlük, her küme indeksi için o kümede kaç düğüm olduğunu tutar.
4. Toplam düğüm sayısını hesaplar (total_nodes), bu sayı şu anki tüm kümelere bağlı düğüm sayısını verir.
5. Eğer hiç düğüm yoksa, yani toplam düğüm sayısı sıfırsa, yeni bir küme oluşturulur ve bu düğüm bu kümeye eklenir.
6. Eğer düğüm zaten bir kümede ise, o kümenin indeksi geri döndürülür.
7. Eğer düğüm hiçbir kümede değilse, en küçük küme indeksi bulunur ve bu indekse sahip kümenin listesine düğüm eklenir.

‘calculate_pledge_within_network’ fonksiyonu, yeni bir düğümün hangi küme içinde yer alması gerektiğini belirlemek için kullanılmaktadır. Bu fonksiyon, hedef adres ve yeni düğümün bağlanmak istediği küme indeksi gibi parametreler alarak, düğümün ağdaki konumunu belirler. Fonksiyonun amacı, ağdaki düğümleri dengeli bir şekilde kümelere yerleştirmek ve ağdaki küme boyutlarını eşit hale getirmektir. Bu sayede ağdaki trafik dengelenmiş olur.

Python ile geliştirilen ağ dengeleme algoritması ise aşağıdaki gibidir:

1.Adımda, UDP soketi oluşturulur.

2.Adımda: Veriler alınıp işlenir. Gelen verinin ilk byte'ının 'r' olup olmadığını kontrol eder. Bu durumda, 'r' olan veri, yönlendirme tablosunu güncellemek için kullanılır. Son bir byte'ın alınması, IPv6 adresinin sonunda bulunan son byte'ın alınması anlamına gelir. Bu byte, genellikle IPv6 adresinin son iki karakterini temsil eder.

- Eğer gelen veri 'n' ise; son bir byte'a göre bir küme oluştur.

Komşu tablosunu güncellenir.

- Eğer gelen veri 'c' ise:

Kanal tablosunu güncellenir.

- Eğer gelen veri 'a' , 'd' , 'd' , 'r' ise:

Denetleyici adres tablosunu güncellenir ve denetleyici IP adresini gönderir.

- Eğer veri 'a' , 'c' , 'k' ise:

Pledge düğümünün hangi kümede olacağını hesaplar ve yanıtı gönderir.

3.Adımda: Sonuç tabloları ekrana basılarak güncellenir.

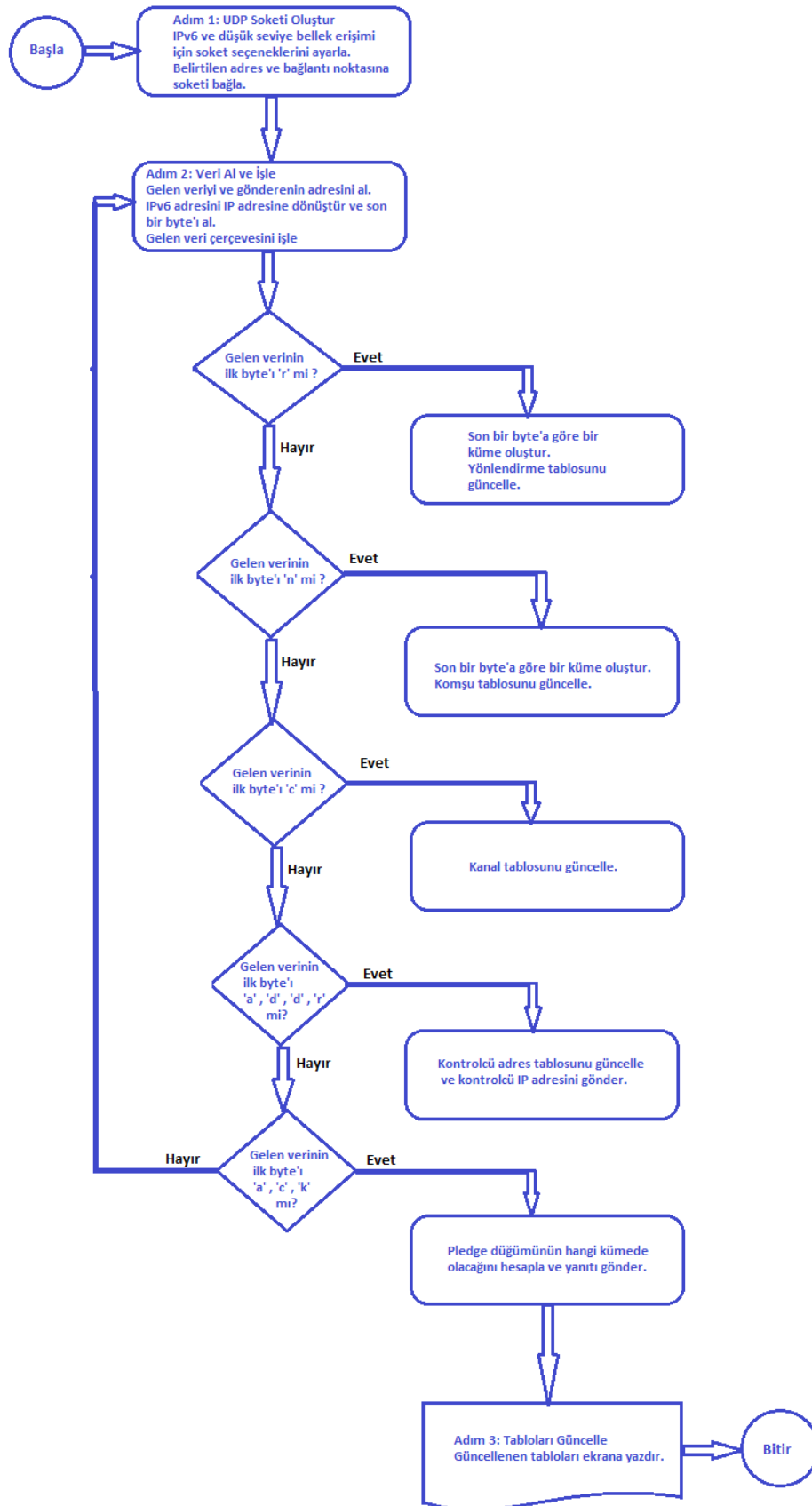
4.Adımda ise: Döngüye devam edilir.

Bu algoritma, verileri işlemek ve ağdaki düğümleri yönetmek için temel bir yapı sağlar.

Bu algoritmayı gerçekleştirmek için Python programlama dili ile bir denetleyici programlanmıştır. Ağa yeni dahil olmak isteyen bir düğüm (pledge) olduğunda, bu istek öncelikle denetleyiciye ulaştırılacak, denetleyici iki komşu ağın düğüm sayısını baz alarak daha az sayıda düğüm olan ağa yeni düğümün katılmasına imkan sağlayacak, yeni düğümün hangi kanal setini kullanacağına karar verebilecektir.

Ağa dahil olmak isteyen yeni düğüm denetleyicinin kanalize ettiği teknikle ilgili RPL Sınır yönlendiriciye bağlandıktan sonra bağlı bulunduğu RPL Sınır yönlendiricisinin önceden tahsis edilmiş kanallarını kullanacaktır.

Algoritmaya ilişkin akış diyagramı aşağıdaki Şekil 5.1.2.1'de gösterilmiştir.



Şekil 5.1.6.1. Ağ Dengeleme Algoritması

5.1.7. Gelişmiş İşaretçiler (Enhanced Beacons - EB)

Gelişmiş İşaretçiler (Enhanced Beacons - EB), zaman dilimli erişim kullanarak düğümler arasındaki çarpışmaları önler ve ağ verimliliğini artırır. Birden fazla kanal kullanımıyla ağ kapasitesi artar ve iletişim güvenilirliği artar. EB'ler, ağa katılan düğümlere gerekli bilgileri içeren özel paketlerdir. TSCH ağlarında senkronizasyon, kanal atlama ve zaman dilimi bilgisi sağlarlar. kanal atlatma özelliği, kanalın sinyal üzerindeki yıkıcı etkilerini minimize eder. Böylece iletişim kalitesi ve güvenilirliği artırılmış olur.

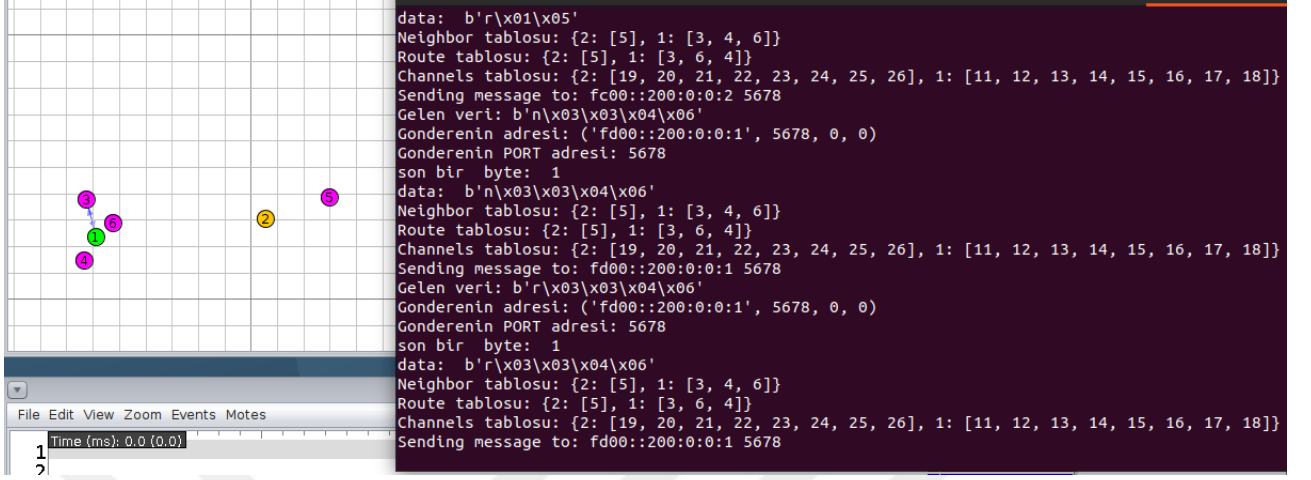
Her yönlendirici için sınırlı sayıda yayım hücresi vardır ve ağa katılan düğümler bu işaretçileri kullanarak ağa katılır. TSCH, EB'lerin ne sıklıkla gönderileceğini ve hangi düğümün göndereceğini uygulama geliştiricisine bırakır. Bu, ağ oluşumunu hızlandırabilir ancak güç tüketimini artırabilir. Bu nedenle uygun bir gönderim sıklığı seçilmelidir.

5.2. Test Ortamı Özellikleri

Tablo 1. Test Ortamı Özellikleri

Parametre	Değer
MAC Modeli	6TiSCH
Yayılım modu	Cooja UDGM Mode
Root Sayısı	2
Düğüm sayısı	4-10-20-30
Başarılı alım oranı(%)	90
Amaç fonksiyonu	MRHOF
Zaman dilimi çerçeve uzunluğu	6
Zaman dilimi aralığı	15 ms
Paylaşımlı zaman dilimi sayısı	6
İletim mesafesi	50 m
Paket gönderme aralığı	1-5-10-12.5 s
RX(%)	50-60-70-80
Test süresi	30 dk

Deney, tek bir amaç fonksiyonu ve farklı başlangıç değerleri (seed) kullanılarak 48 kez (düğüm, RX ve paket gönderme aralığı değiştirilerek) 30 dakika boyunca çalıştırılmıştır. Sonuçlar ortalamaları alınarak analiz edilmiştir. Deneyde, 2 kök düğüm (RPL Sınır Yönlendirici) ve farklı sayıda (4-10-20-30) adet istemci düğümler kullanılmıştır. Düğüm tipi olarak Exp5438 modülü kullanılmış ve düğümler senaryo alanına rastgele yerleştirilmiştir. Testlerde kullanılan parametreler Tablo 1'de verilmiştir.



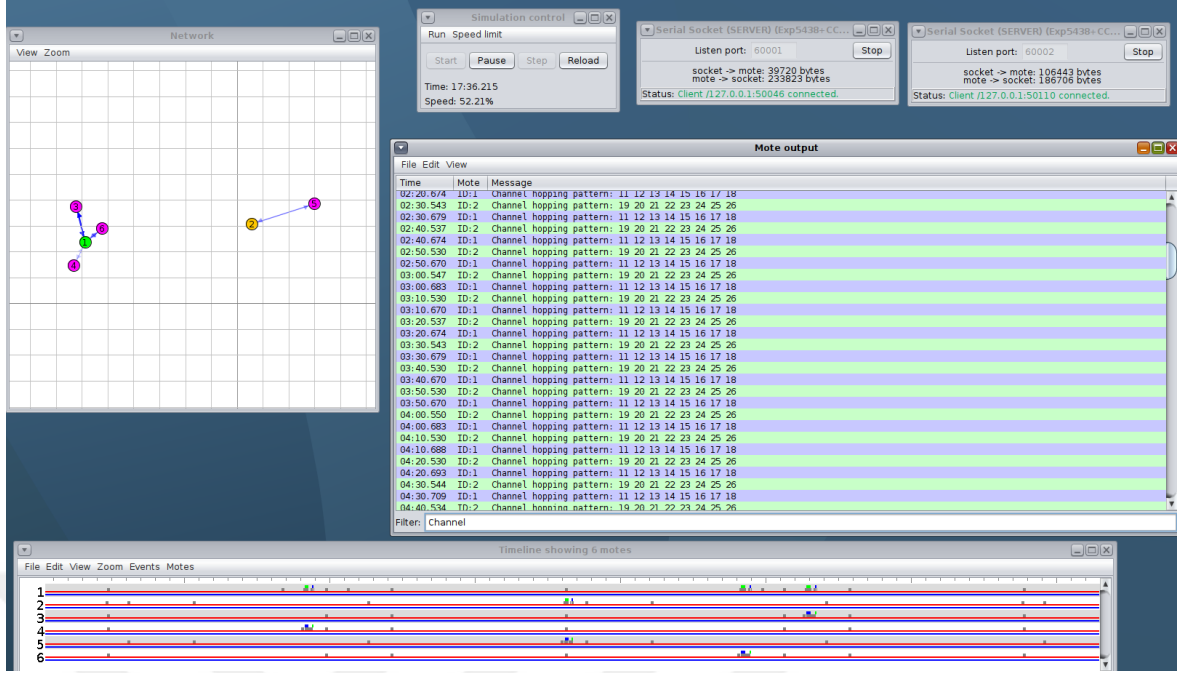
Şekil 5.2. Algoritma-1 Simülasyon Ortamı

Şekil 5.2 'te Algoritma-1 simülasyon sonuçlarına bakıldığında iki adet RPL Sınır Yönlendirici ve onlara bağlı 4 düğüm olmak üzere toplam 6 düğüm ile yapılan deneyde;

RPL Sınır Yönlendirici-1 ve ona bağlı 3 düğüm bulunmaktadır. RPL Sınır Yönlendirici-2 ve ona bağlı 1 düğüm bulunmaktadır. Düğümlerin yönlendirme (routing) durumları konsol ekrana Python programlama dili ile yazıldığında ise;

Komşuluk (Neighbor), yönlendirme (Route), kanallar (Channels) tabloları incelendiğinde, düğümlerin bağlı olduğu RPL Sınır Yönlendiricilerin mevcut kanal setlerini { 1: [11, 12, 13, 14, 15, 16, 17, 18] , 2: [19, 20, 21, 22, 23, 24, 25, 26] } başarıyla aldığı kanal tablosundan gözlemlenmektedir.

Komşuluk tablosu, düğümlerin komşu düğümlerini içerir. Komşuluk tablosu { 2: [5] , 1: [3,4,6] } şeklinde olan 2 numaralı küme indeksine sahip düğümün komşusunun [5] olduğu, 1 numaralı küme indeksine sahip düğümün komşuları ise [3,4,6] olduğu görülmektedir. Yönlendirme tablosu ise , düğümler arasındaki rotaları içerir. Örneğin, 1 numaralı küme indeksi için belirtilen rota [3, 6, 4] olduğu görülmektedir.



Şekil 5.2.1. RPL Sınır Yönlendiricilere Kanal Ataması

5.2.1. Başarım Metrikleri

Test sonuçları, üç farklı performans ölçütünü değerlendirmek amacıyla incelenmiş ve sonuçlar grafiklerle görselleştirilmiştir. Bu performans ölçütleri şunlardır:

- ✓ Paket İletim Oranı
- ✓ Ortalama Paket Gecikmesi
- ✓ Paket Kayıpları

Bu başarıım ölçütleri, test sonuçlarını ayrıntılı bir şekilde değerlendirmek ve ağ performansını analiz etmek için kullanılmıştır.

5.2.3. Paket İletim Oranı

Paket İletim Oranı (PDR), bir iletişim ağında veya protokolünde gönderilen veri paketlerinin başarıyla hedefe ulaşma yüzdesini ifade eden bir performans ölçütüdür. PDR, iletişim ağının güvenilirliğini ve etkinliğini değerlendirmek için kullanılır. Genellikle yüzde olarak ifade edilen bu oran, gönderilen paketlerin ne kadarının alıcıya sorunsuz bir şekilde teslim edildiğini gösterir. Yüksek bir PDR, iletişim ağının veri iletiminde daha güvenilir olduğunu gösterirken, düşük bir PDR ise iletimde sorunların olduğunu veya kayıpların yaşandığını gösterir.

PDR'nin hesaplanması için, gönderilen paket sayısı ile alıcıya başarıyla teslim edilen paket sayısı karşılaştırılır. Bu oran, ağın kalitesini değerlendirmek ve ağ performansını izlemek için önemlidir, özellikle kablosuz iletişim sistemlerinde ve ağ protokollerinin tasarımında kullanılır. Ağ mühendisleri ve tasarımcılar, PDR'yi ölçerek iletişim ağlarını optimize etmek ve daha güvenilir iletişim sağlamak için gerekli önlemleri alabilirler.

Paket İletim Oranı'nı (Packet Delivery Ratio - PDR) hesaplamak için aşağıdaki formülü kullanılabilir:

$$PDR(\%) = \frac{\text{Başarıyla Teslim Edilen Paket Sayısı}}{\text{Gönderilen Toplam Paket Sayısı}} \times 100 \quad (2)$$

Bu formülde, PDR yüzde cinsinden ifade edilir. Başarıyla teslim edilen paket sayısı, gönderilen toplam paket sayısına bölünür ve sonuç 100 ile çarpılarak yüzde cinsinden elde edilir. Bu sayede iletişim ağının veya protokolün ne kadar etkili ve güvenilir olduğunu değerlendirebilirsiniz. PDR, ağ performansını izlemek ve sorunları tespit etmek için önemli bir ölçüttür.

5.2.4. Ortalama Paket Gecikmesi

Ortalama Paket Gecikmesi (Average Packet Delay), bir iletişim ağı veya veri iletim sistemi içinde veri paketlerinin gönderilmesi ve alıcının paketi alması arasındaki ortalama zaman dilimini ifade eden bir performans ölçütüdür. Genellikle milisaniye (ms) cinsinden ifade edilir ve veri iletim sürecindeki zaman gecikmelerinin ortalamasını gösterir. Bu gecikmeler, paketin gönderildiği kaynaktan hedef alıcıya ulaşana kadar ağ üzerindeki çeşitli etkenler nedeniyle oluşabilir. Bunlar arasında yönlendirme işlemleri, ağ trafiği yoğunluğu, iletim hatası düzeltmeleri ve benzeri faktörler yer alır.

Ortalama Paket Gecikmesi, iletişim ağı veya protokolün hızını ve veri iletiminin ne kadar hızlı ve etkili olduğunu değerlendirmek için kullanılır. Düşük bir ortalama paket gecikmesi, ağın hızlı ve verimli olduğunu gösterirken, yüksek bir ortalama gecikme süresi, iletişimde gecikmelerin yaşandığını ve gecikme hassasiyeti gerektiren uygulamalar için sorun olabileceğini gösterir. Bu nedenle ağ mühendisleri ve tasarımcıları, ortalama paket gecikmesini izlemek ve optimize etmek için çeşitli teknikler ve önlemler kullanırlar,

özellikle gerçek zamanlı iletişim ve duyarlı uygulamalar için düşük gecikme süreleri kritik öneme sahiptir.(Bekar,2023).

Elde edilen gecikme sürelerini toplayıp ve toplamı paket sayısına böldüğümüzde ortalama paket gecikmesini elde ederiz. Bu hesaplamanın formülünü aşağıdaki gibi ifade edebiliriz:

$$Ortalama\ Gecikme(ms) = \frac{\sum_{i=1}^n Gecikme_i}{n} \quad (3)$$

Burada, "n" IoT ağındaki toplam paket sayısını temsil eder.

5.2.5. Paket Kayıpları

TSCH (Time-Slotted Channel Hopping) MAC (Medium Access Control) katmanındaki paket kayıpları, kablosuz iletişim ağlarında kullanılan bu özel protokolün işleyişi sırasında meydana gelen iletişim kayıplarını ifade eder. TSCH, özellikle Nesnelerin İnterneti (IoT) ağları için tasarlanmış bir MAC katman protokolüdür ve düşük güç tüketimi ve güvenilir iletişim sağlama amacı güder. Ancak, çeşitli faktörler nedeniyle paket kayıpları yaşanabilir.

Bu kayıplar, kablosuz iletişimde karşılaşılan gürültü, sinyal zayıflaması, çakışan iletimler, hava koşulları ve diğer çevresel etkenler gibi nedenlerden kaynaklanabilir. Ayrıca, TSCH protokolünün belirli zaman dilimlerinde çalışması nedeniyle, zamanlama sorunları ve senkronizasyon eksiklikleri de paket kayıplarına yol açabilir. Paket kayıpları, ağın güvenilirliğini etkileyebilir ve iletişim başarısızlıklarına neden olabilir. Bu nedenle, TSCH protokolü kullanan IoT ağları, paket kayıplarını izlemek ve azaltmak için çeşitli mekanizmalar ve hata düzeltme teknikleri kullanır.

$$\text{Paket Kayıpları Oranı (\%)} = \frac{(\text{Gönderilen Toplam Paket Sayısı} - \text{Alınan Paket Sayısı})}{\text{Gönderilen Toplam Paket Sayısı}} \times 100 \quad (4)$$

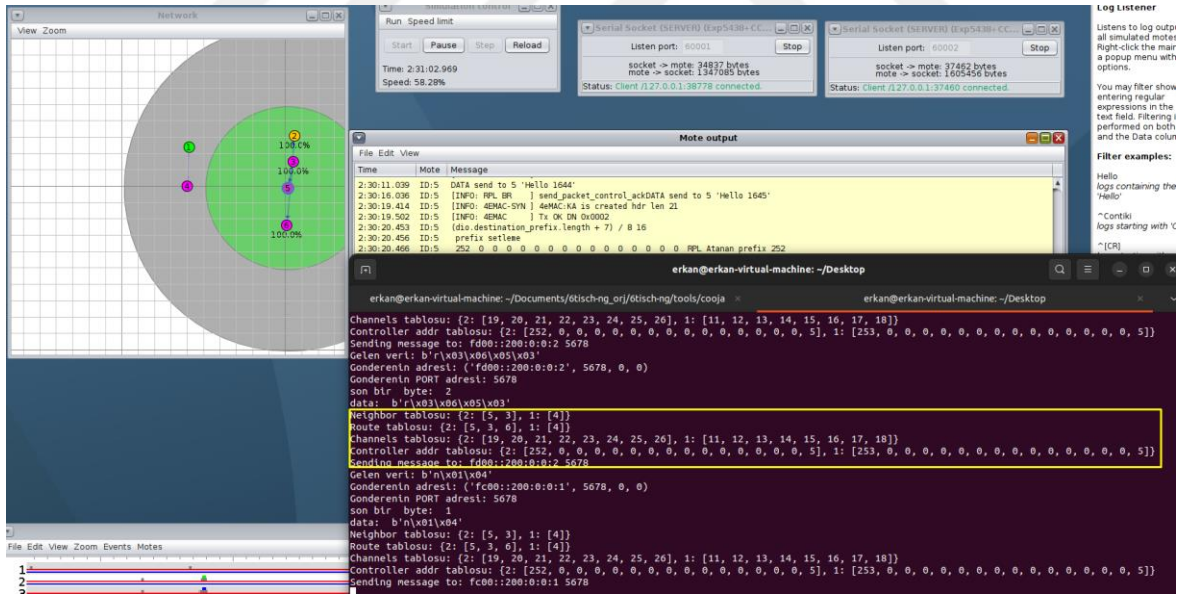
6. SONUÇLAR

Bu tez çalışmasında RPL Protokolü kullanılarak IoT komşu ağların dengelenmesine yönelik olarak yeni algoritmalar ve bu algoritmalar için C dilinde ve Python dilinde bazı yazılımlar geliştirilmiştir.

6.1. Ağ Kurulum Sonuçları

Algoritma-1 simülasyon sonuçlarına bakıldığında iki adet RPL Sınır Yönlendirici ve onlara bağlı 4-10-20-30 adet düğüm ile yapılan deneylerde;

- RPL sınır yönlendiricilerin atanan kanalları başarıyla aldıkları gözlemlenmiştir.
- Komşu ağlara kanal ataması yapılarak RPL sınır yönlendiricilere bağlı olan düğümlerinde aynı kanalları alabilmesi sağlanmıştır.



Şekil 6.1. Düğümlerin Kanal Atamaları

Şekil 6.1' de görüldüğü üzere ağ kurulum mekanizması sayesinde RPL Sınır yönlendiricilere { 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26 } kanal setinin 8-8 olarak { 1: [11, 12, 13, 14, 15, 16, 17, 18], 2: [19, 20, 21, 22, 23, 24, 25, 26] } başarıyla tahsis edildiği görülmektedir.

Tablo 2. Ağ Kurulumu-Kanal Atama Sonuçları

Standart Yöntem		Kanal Tahsis Yöntemi
RPL Sınır Yönlendirici -1 kanal seti	{ 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26 }	{ 11, 12, 13, 14, 15, 16, 17, 18 }
RPL Sınır Yönlendirici -2 kanal seti	{ 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26 }	{ 19, 20, 21, 22, 23, 24, 25, 26 }

6.2. Ağ Dengeleme Algoritması Sonuçları

4-10-20-30 adet düğüm ile yapılan simülasyonlarda aşağıda Tablo 3'te görüldüğü üzere Standart RPL çözümlerinde düğümler toplayıcılara rastgele ve düzensiz dağılırken, Python dili ile geliştirilen Ağ Denetleyici (Network Controller) uygulaması ile düğümlerin dinamik ve eşit bir şekilde ağa senkron oldukları gözlemlenmiştir.

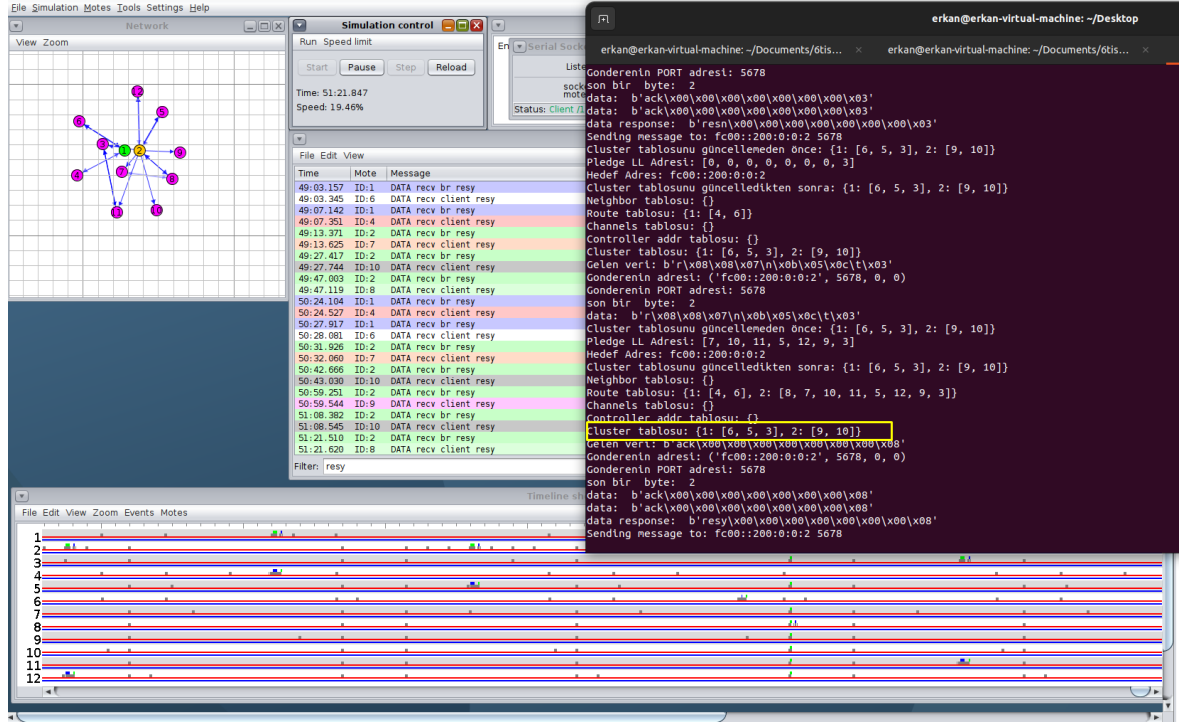
Tablo 3. Farklı Düğüm Sayılarında Ağ Dengeleme Sonuçları

Düğüm Sayısı		Standart Çözüm	Ağ Denetleyici Çözüm
RPL Sınır Yönlendirici -1'e bağlı düğüm sayısı	4 Düğüm	1	2
RPL Sınır Yönlendirici -2'ye bağlı düğüm sayısı		3	2
RPL Sınır Yönlendirici -1'e bağlı düğüm sayısı	10 Düğüm	2	5
RPL Sınır Yönlendirici -2'ye bağlı düğüm sayısı		8	5
RPL Sınır Yönlendirici -1'e bağlı düğüm sayısı	20 Düğüm	6	10
RPL Sınır Yönlendirici -2'ye bağlı düğüm sayısı		14	10
RPL Sınır Yönlendirici -1'e bağlı düğüm sayısı	30 Düğüm	12	15
RPL Sınır Yönlendirici -2'ye bağlı düğüm sayısı		18	15

Ağ dengeleme sonuçlarına bakıldığında, Python dili ile geliştirilen Ağ Denetleyici (Network Controller) uygulaması ile RPL Sınır Yönlendiricileri ifade eden kümelere (Cluster) eşit ve dengeli bir şekilde düğümlerin tahsis edildiğini Cooja Simülatöründe

yapılan deneylerde gözlemlenmiştir. 1 ve 2 nolu Sımr Yönlendiricilere ağı dahil olan düğümlerin sayısal olarak dengeli bir şekilde bağlandıkları gözlemlenmiştir.

- 1: {6} 2: {9}
 1: {6,5} 2: {9}
 1: {6,5} 2: {9,10}
 1: {6,5,3} 2: {9,10}
 1: {6,5,3} 2: {9,10,8}
 1: {6,5,3,11} 2: {9,10,8}



Şekil 6.2. Denetleyici ile Kümelere Düğümlerin Tahsis Edilmesi

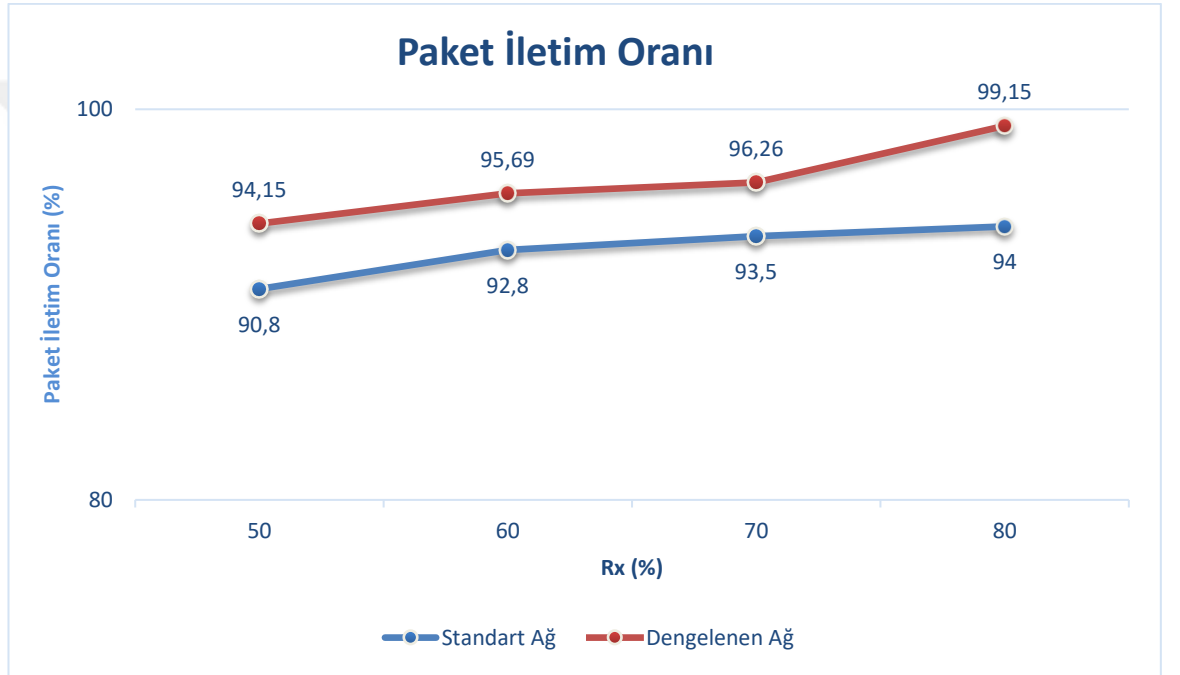
6.2.1. Ağ Dengeleme ile Paket İletim Oranı (PDR) Sonuçları

Paket İletim Oranları incelendiğinde;

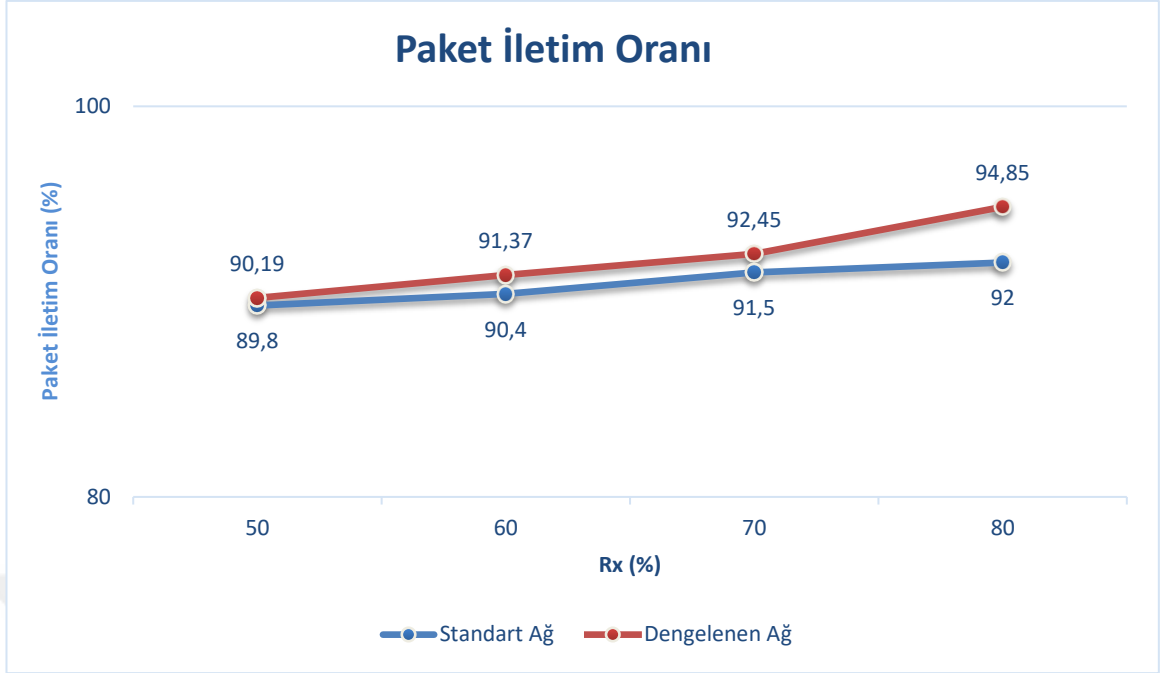
Python dili ile geliştirilen Ağ Denetleyici (Network Controller) uygulaması ile ağ dengelemesi yapılarak kurulan ağlarda farklı düğüm sayıları ile yapılan deneylerde, standart (RPL ile oluşturulan komşu ağ) ağlara göre dengeli ve dinamik bir şekilde komşu ağlara eşit düğüm dağılımı yapılmasının yanısıra, paket iletim oranlarında da (PDR) aynı şekilde standart kurulan ağı göre daha başarılı sonuçlar elde edilmiştir.

Aşağıdaki tablolarda 4-10-20 adet düğüm için yapılan deneylerin sonuç grafikleri bulunmaktadır. % 50 - %60 - %70 - %80 RX ile yapılan deneylerde RX arttıkça paket iletim oranlarında RX oranı %50’den %80’e kadar artış gözlemlenmektedir.

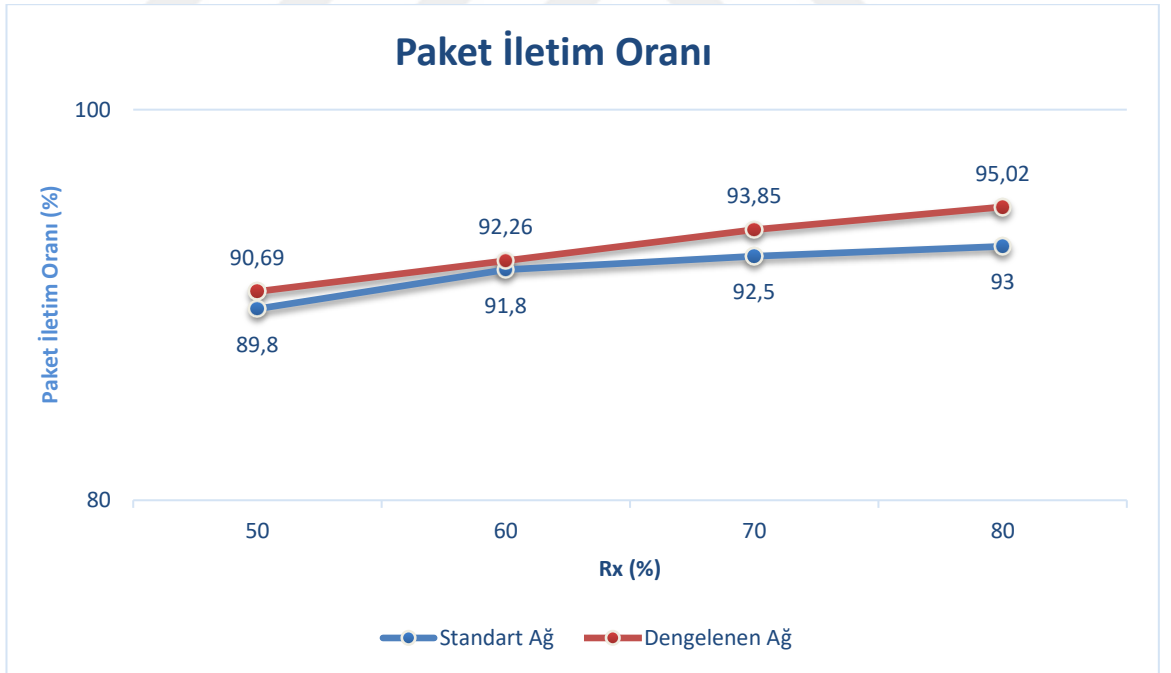
“Standart Ağ” olarak gösterilen grafik çizgileri komşu ağlardaki düğümlerin rastgele yerleştiği durumu ifade ederken, “Dengelenen Ağ” çizgileri ise geliştirilen Ağ Denetleyici uygulaması ile düğümlerin eşit ve dinamik bir şekilde komşu ağlara bağlandığı durumu şematize etmektedir.



Şekil 6.2.1.1. Paket İletim Oranları (2 Sınır Yönlendirici & 4 Düğüm)



Şekil 6.2.1.2. Paket İletim Oranları (2 Sınır Yönlendirici & 10 Düğüm)



Şekil 6.2.1.3. Paket İletim Oranları (2 Sınır Yönlendirici & 20 Düğüm)

6.2.2. Ağ Dengeleme ile Paket Hızına Göre Paket İletim Oranı (PDR) Sonuçları

Yukarıdaki tablolarda 4-10-20 adet düğüm için yapılan deneylerin sonuç grafikleri bulunmaktadır. Paket gönderme aralığı aşağıdaki tanıma göre udp-client sınıfı içinde başlangıçta 5 saniye olarak belirlenmiştir. CLOCK_SECOND değeri 1 saniyedir.

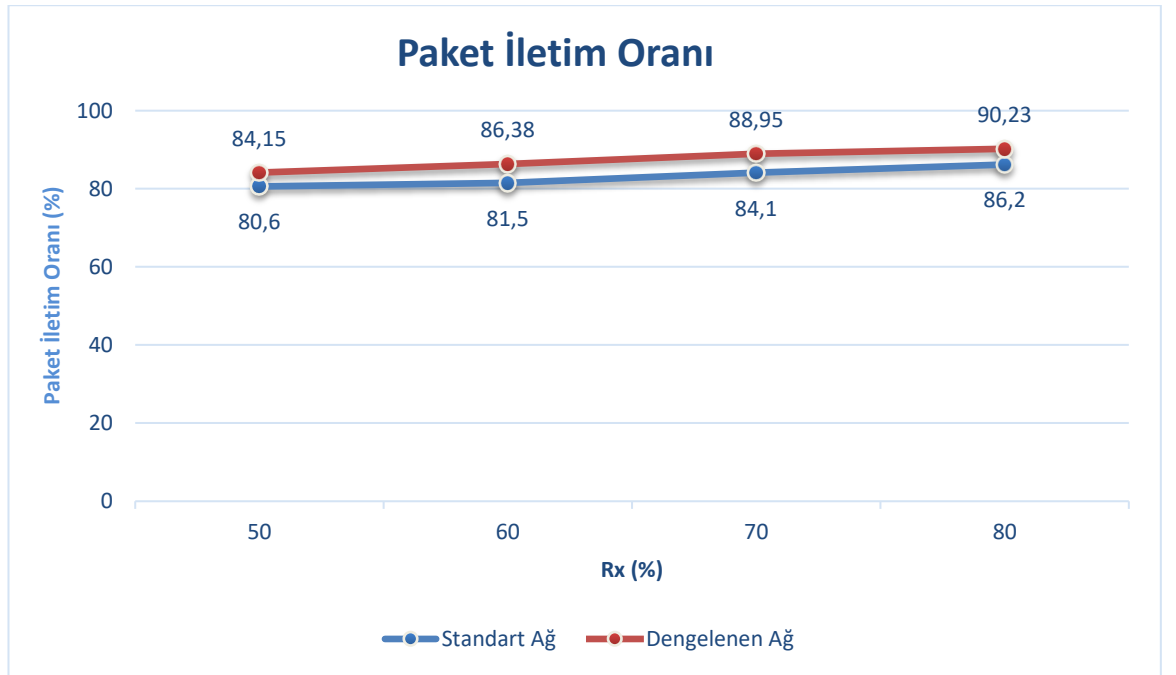
```
#define SEND_INTERVAL (5 * CLOCK_SECOND) (5)
```

Paket gönderme aralığı aşağıdaki tanımdaki gibi 12.5 saniye yapılarak 4-10-20 adet düğüm için paket iletim oranı test simülasyonları yapılmıştır.

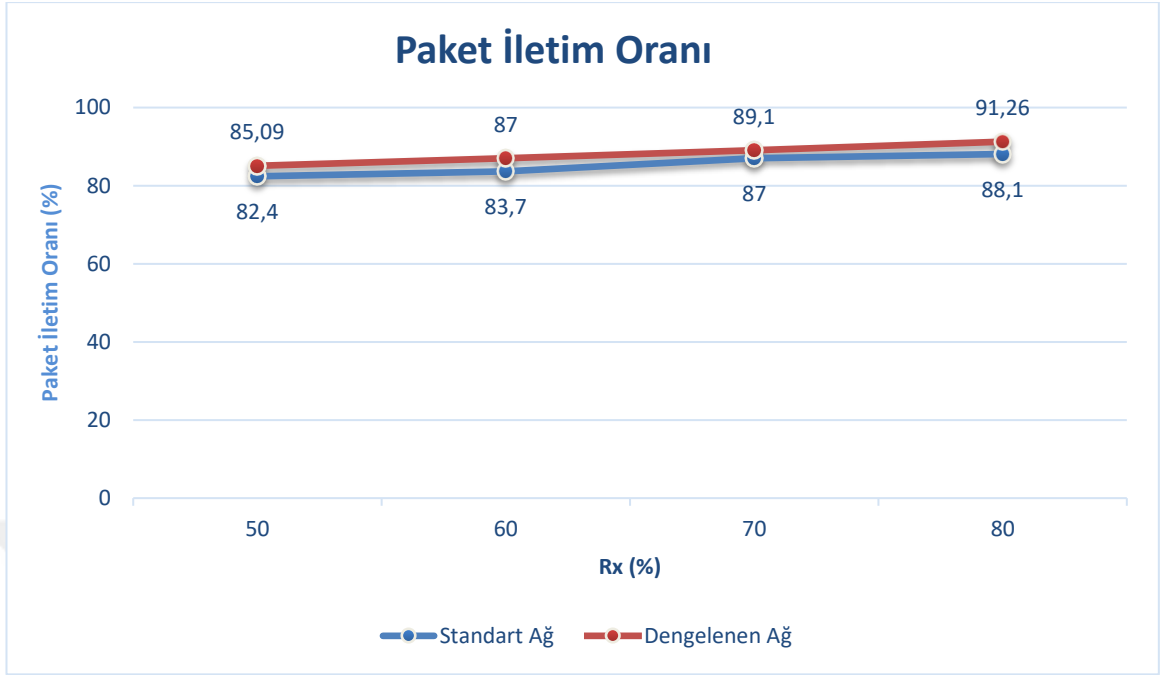
```
etimer_set(&timer_hello_packet, SEND_INTERVAL*2.5); (6)
```

Paket gönderme aralığı değiştiğinde, paket iletim oranlarının nasıl değişeceği sorusuna yanıt aramak içinde paket gönderme aralığı 1,5,10 saniye olarak ayrı ayrı tanımlanıp, 20 düğüm ile simülasyon testleri yapıldığında ise aşağıdaki grafikler elde edilmiştir.

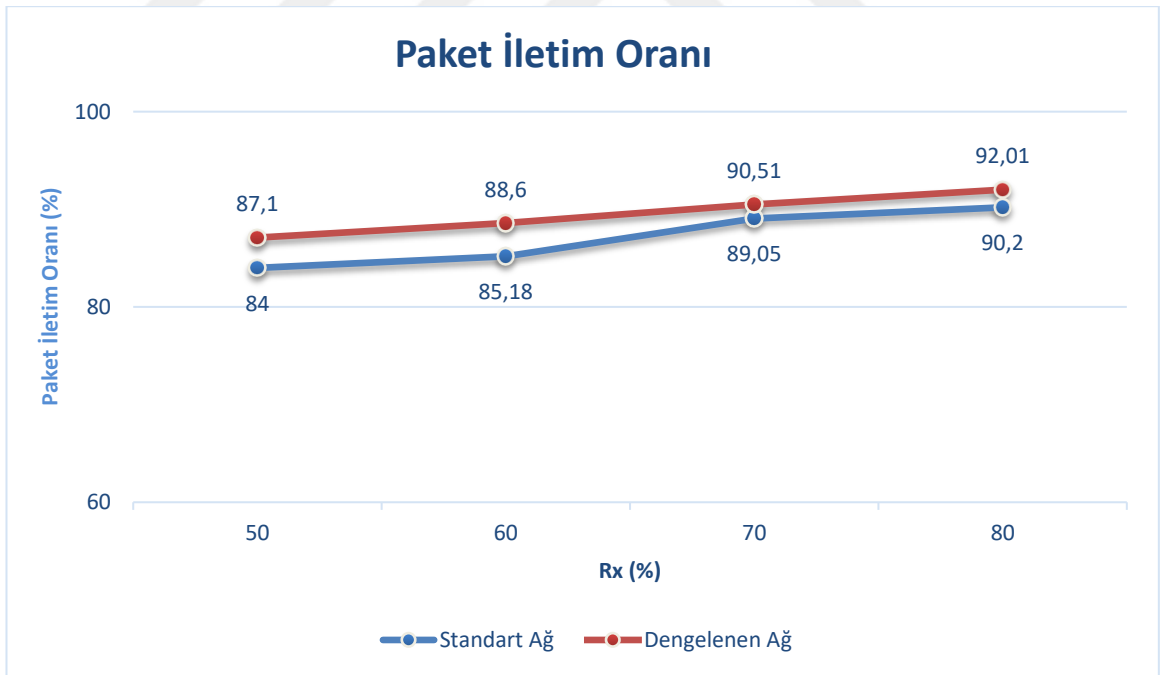
```
#define SEND_INTERVAL (1 * CLOCK_SECOND)
#define SEND_INTERVAL (5 * CLOCK_SECOND) (7)
#define SEND_INTERVAL (10 * CLOCK_SECOND)
```



Şekil 6.2.2.1. Paket İletim Oranları (2 Sınır Yönlendirici & 20 Düğüm & 1 saniye)



Şekil 6.2.2.2. Paket İletim Oranları (2 Sınır Yönlendirici & 20 Düğüm & 5 saniye)



Şekil 6.2.2.3. Paket İletim Oranları (2 Sınır Yönlendirici & 20 Düğüm & 10 saniye)

Yukarıdaki grafiklerde 20 düğüm ile yapılan farklı paket hızlarındaki simülasyon test sonuçları analiz edildiğinde; paket gönderme aralığı arttıkça paket iletim oranlarının arttığı,

paketleri hızlı gönderince ise (saniye düştükçe) paket iletim oranının düştüğü gözlemlenmiştir.

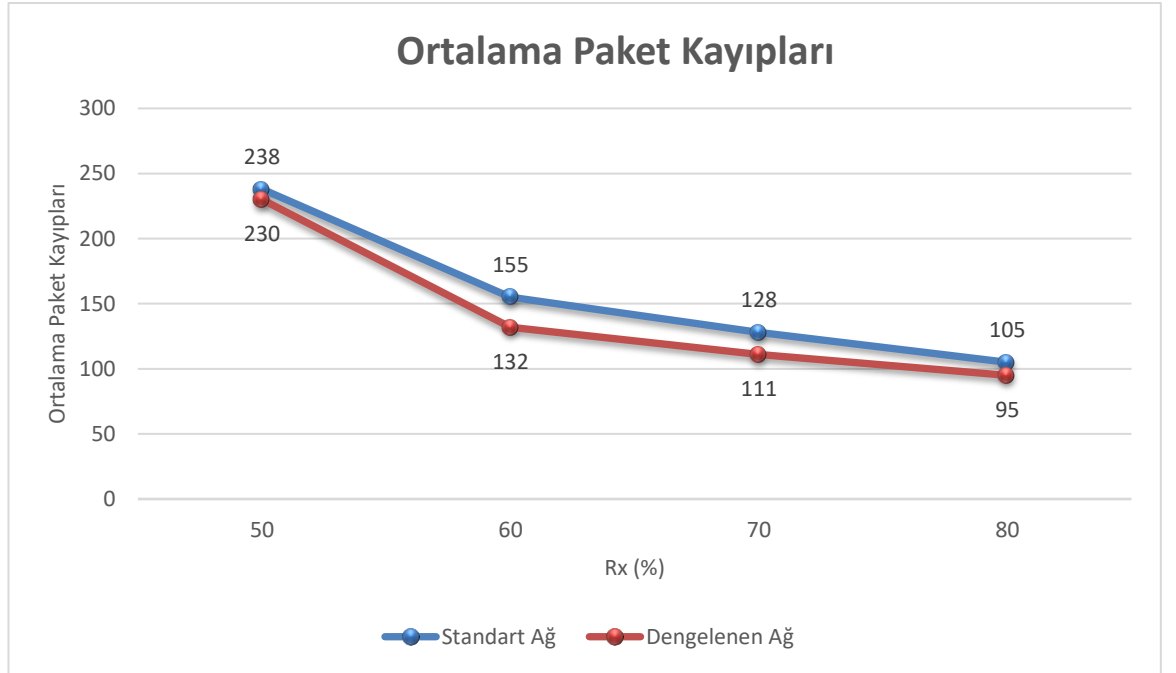
6.2.3. Ağ Dengeleme Paket Kayıpları Sonuçları

Paket kayıpları sonuçları incelendiğinde;

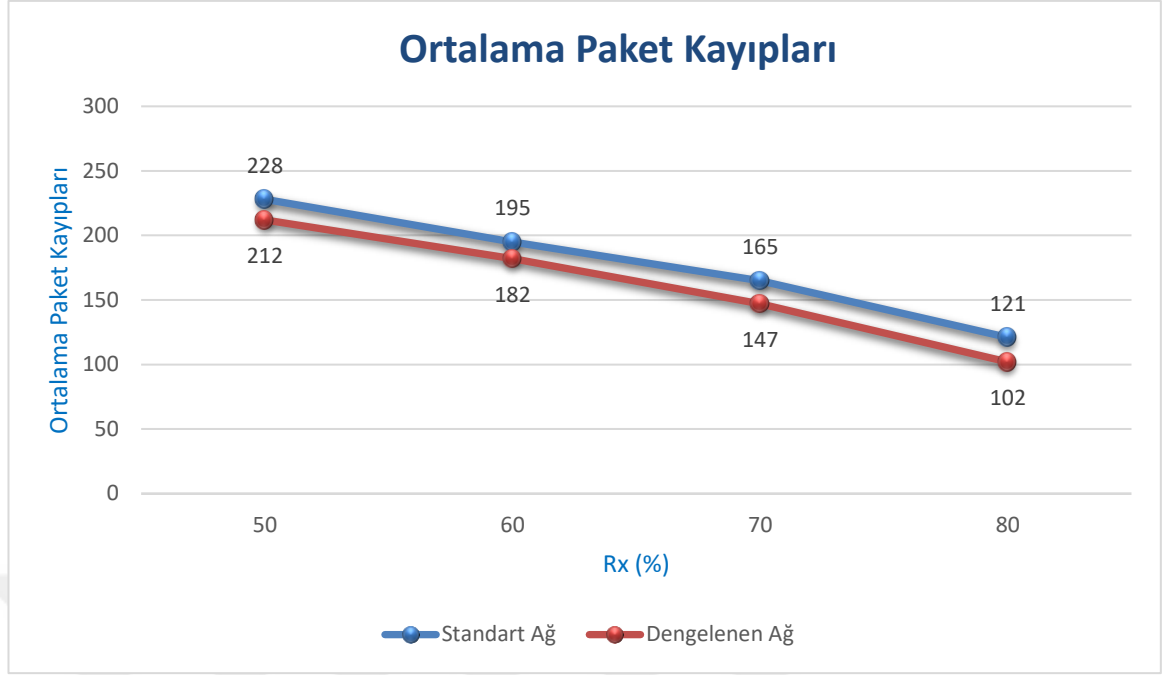
Python dili ile geliştirilen Ağ Denetleyici (Network Controller) uygulaması ile ağ dengelenmesi yapılarak kurulan ağlarda, standart (RPL ile oluşturulan komşu ağ) ağlara göre ağın dengeli olmasının yanısıra paket kayıplarında da azalış olduğu, paketlerin daha az kayıpla iletildiği tespit edilmiştir.

4-10-20 düğüm ve % 50 - %60 - %70 - %80 RX ile yapılan deneylerde RX arttıkça paket kaybı oranlarında RX oranı %50'den %80'e kadar azalış gözlemlenmektedir.

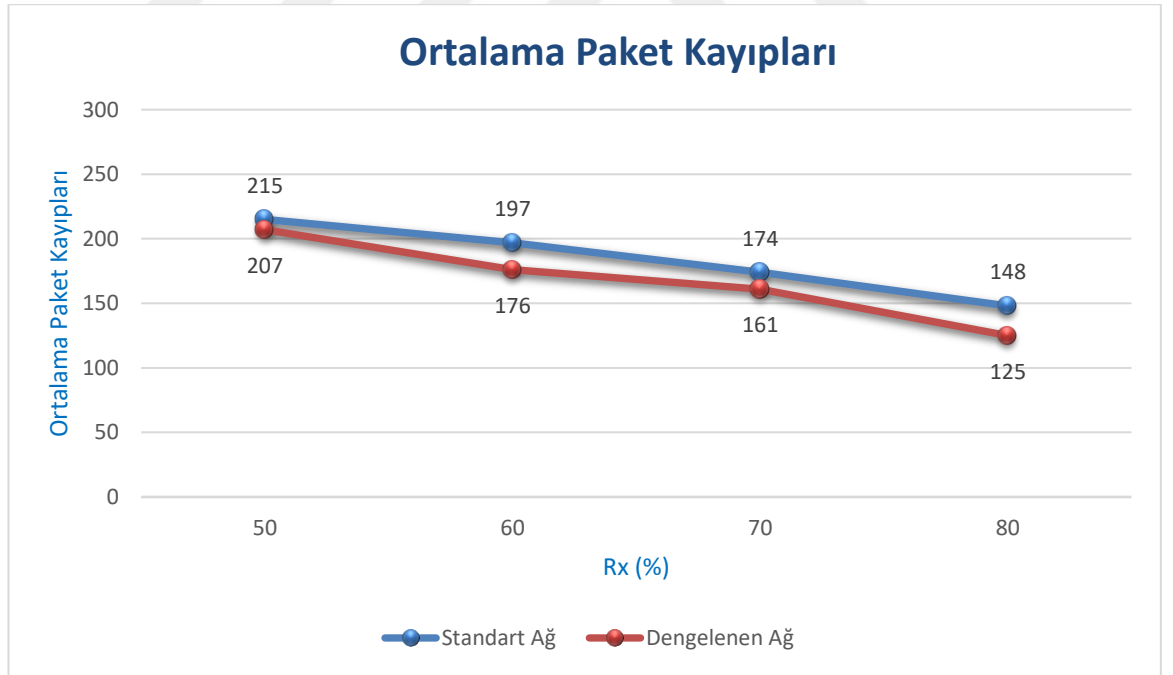
“Standart Ağ” olarak gösterilen grafik çizgileri komşu ağlardaki düğümlerin rastgele yerleştiği durumu ifade ederken, “Dengelenen Ağ” çizgileri ise geliştirilen Ağ Denetleyici uygulaması ile düğümlerin eşit ve dinamik bir şekilde komşu ağlara bağlandığı durumu şematize etmektedir. 4-10-20 adet düğüm için yapılan deneylerin grafikleri Şekil 6.2.2.1'den itibaren aşağıda gösterilmiştir.



Şekil 6.2.3.1. Ortalama Paket Kayıpları (2 Sınır Yönlendirici & 4 Düğüm)



Şekil 6.2.3.2. Ortalama Paket Kayıpları (2 Sınır Yönlendirici & 10 Düğüm)



Şekil 6.2.2.3. Ortalama Paket Kayıpları (2 Sınır Yönlendirici & 20 Düğüm)

7. BULGULAR VE SONUÇLAR

Aynı coğrafi ortamda bulunan komşu 6TiSCH ağların dengelenmesi için yapılan komşu ağların kurulum mekanizması ve geliştirilen algoritmik “Ağ Denetleyici (Network Controller)” uygulaması sonucunda;

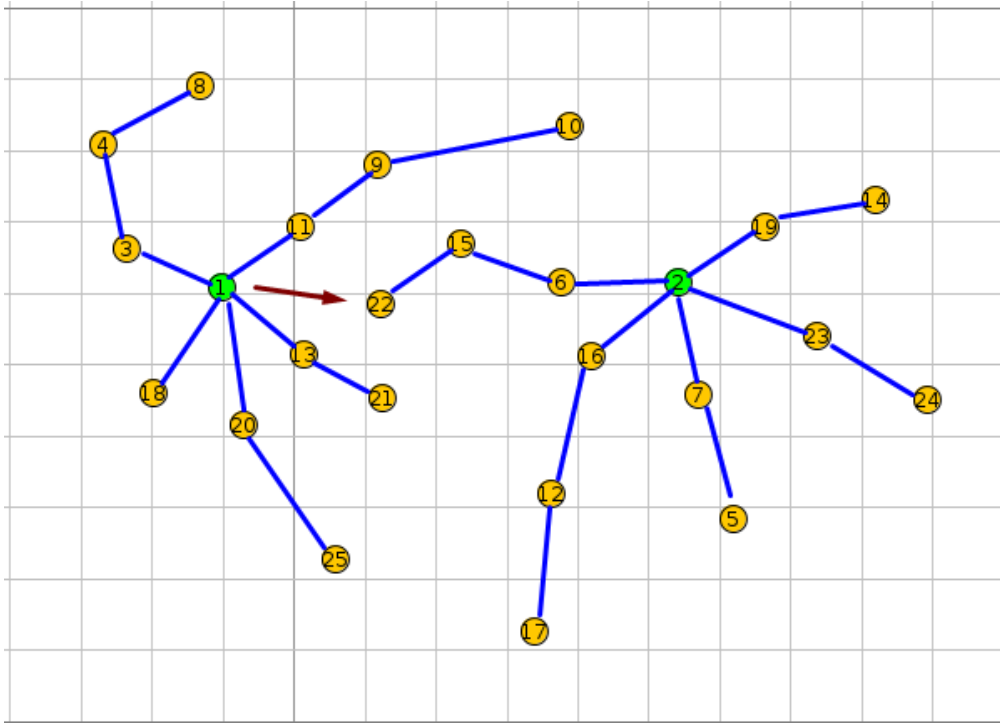
- ✓ Düğümlerin başlangıçta tahsis edilen belirli kanallar üzerinden çalışması sağlanmıştır.
- ✓ Ağa yeni katılmak isteyen düğümlerin, programlanan bir denetleyici aracılığıyla hangi kanal setini kullanacakları ve dolayısıyla hangi RPL sınır yönlendiriciye dahil olacakları tayin edilerek ağın dengeli ve kararlı bir şekilde kurulumu gerçekleştirilmiştir.
- ✓ Farklı düğüm sayıları üzerinden komşu ağlar test edilerek, dengelenen ağın standart ağa göre paket iletim oranlarında da artış olduğu gözlemlenmiştir.
- ✓ Ağda düğümler arasında kök düğüm değiştirme oranı önemli ölçüde azalmıştır.
- ✓ Kurulan komşu ağlara yeni katılacak olan düğümlerin düzensiz olarak değilde dinamik ve eşit bir şekilde ağa katılımı sağlanarak komşu ağlarda dengeleme sağlandı.
- ✓ RPL ve DODAG yapısının Ağ Denetleyici uygulaması ile entegrasyonu sayesinde ağın yönlendirme performansı iyileşti.
- ✓ Topolojideki düğümler arasındaki iletişim daha kararlı ve güvenilir hale geldi.
- ✓ Ağın enerji verimliliği arttı ve düğümlerin batarya ömrü uzadı.
- ✓ Çevresel koşullardaki değişikliklere daha iyi uyum sağlanarak ağın adaptasyon yeteneği arttı.

Bu çalışmada, Internet of Things (IoT) ağının daha efektif ve güvenilir bir şekilde kurulması ve işletilmesi hedeflenmektedir. Bunun için RPL protokolüne yeni bir ağ dengeleme algoritması ve kanal tahsis mekanizması önerilmektedir. Bu kapsamda farklı tunsip ve farklı kanal setleri ile komşu iki tane RPL Sınır Yönlendirici ve onlara bağlı düğümler simülasyon ortamında farklı düğümler ile test edilmiştir. RPL Sınır Yönlendiricilere düğümlerin geliştirilen program sayesinde dengeli olarak bağlanmaları sağlanmıştır.

RPL Sınır Yönlendiricilere başlangıçta iki ayrı kanal seti tanımlanmıştır. Bu sayede sınır yönlendiricilere bağlı olan düğümlerinde aynı kanal seti üzerinden çalışabilmesi

sağlanmıştır. Python programlama dili ile “Ağ Denetleyici (Network Controller)” uygulaması geliştirilerek, programlanan bu denetleyicinin ağın düğüm sayısını ve mevcut kanal setlerini kontrol edebilmesi sağlanmıştır.

Ağ denetleyici sayesinde komşu ağlara yeni dahil olmak isteyen düğümlere, hangi sınır yönlendiriciye bağlanacakları ve hangi kanal setini kullanabilecekleri tayin edilebilmiştir. Bu sayede ağın düğüm sayısı olarak dengelenmesi sağlanarak çakışmalar minimize edilmiş, enerji tüketimi düşürülmüş, paket iletim oranları artırılmıştır. Bu çözüm sayesinde, IoT ağında daha efektif, dengeli ve güvenilir bir iletişim sağlanabilecektir.



Şekil 7. Komşu IoT Ağı ve Düğümler

8. ÖNERİLER

Yüksek Lisans tezi olarak sunulan bu çalışmada;

- Nesnelerin İnterneti (IoT) ağının daha efektif ve güvenilir bir şekilde kurulması ve işletilmesi hedeflenmektedir. Bunun için RPL protokolüne yeni bir kanal tahsis mekanizması önerilmektedir. Bu kapsamda farklı tunsip ile komşu iki tane komşu RPL Sınır Yönlendirici ilk kez simüle edilmiştir.
- RPL Sınır Yönlendiricilere başlangıçta farklı kanal setleri atanarak, bağlı oldukları düğümlerinde aynı kanal setleri üzerinden çalışabilmeleri sağlanmıştır.
- Ağa yeni dahil olmak isteyen düğümlerin düğüm yoğunluğu az olan ağa katılabilmelerini sağlamak için bir **Ağ Denetleyici (Network Controller)** isminde bir denetleyici Python programlama dili ile geliştirilmiştir. Bu uygulama sayesinde hem komşu IoT ağların düğüm yoğunluğu tespit edilebilmiş hem de ağa dahil olmak isteyen yeni düğümlerin düğüm yoğunluğu az olan ağa katılabilmesi için gerekli çalışmalar yapılmıştır.
- Bu çözüm sayesinde, IoT ağında daha efektif ve dengeli bir ağ kurulmuştur. Çakışmalar minimize edilmiş, enerji tüketimi azaltılmıştır. Paket iletim oranlarında da ağlar izole çalıştığından dolayı ciddi artışlar elde edilmiştir.
- İlerleyen çalışmalarda ise komşu 6TiSCH ağlardaki düğümlerin belirli zaman aralıklarıyla düğüm sayılarının sorgulanması, ağın periyodik olarak senkronizasyonu, geliştirilen algoritmaya göre göre düğümlerin enerji tüketim hesaplamalarının yapılması için Ağ Denetleyici (Network Controller) uygulamasının geliştirilmesi planlanmaktadır.

9. KAYNAKLAR

Sharma, P., K., Chen, M., Y., & Park, J., H. (2018). *A Software Defined Fog Node Based Distributed Blockchain Cloud Architecture for IoT*, IEEE Access, 115-124.

Vilajosana, X., Pister, K., & Watteyne, T. (2022). *Minimal IPv6 over the TSCH Mode of IEEE 802.15.4e (6TiSCH) Configuration*.

URL-1. <https://www.youtube.com/watch?v=jIhk2uTMKoQ> (IEEE 802.15.4 TSCH MAC protocol | TSCH | MAC layer | ALOHA | IoT Protocols) adresinden alınmıştır.

URL-2. https://www.researchgate.net/figure/Example-of-an-RPL-network-with-two-instances-and-three-DODAG_fig2_353820616 adresinden alınmıştır.

Bekar, T. (2023). *IETF 6TiSCH Ağlar İçin Q-Öğrenme Algoritmasından Esinlenilmiş Amaç Fonksiyonu Optimizasyonu*, Yüksek Lisans Tezi, K.T.Ü., Fen Bilimleri Enstitüsü, Trabzon.

Ahmed, A.U., Islam, M.T., Ismail, M., & Ghanbarisabagh, M.(2014). Dynamic resource allocation in hybrid access femtocell network, ScientificWorldJournal, DOI: 10.1155/2014/539720

Qasem, M. (2018). *Load Balancing and Context Aware Enhancements for RPL Routed Internet of Things*, Doktora Tezi, Edinburgh Napier University, Edinburgh.

Dehghan, P. (2020). *Dağıtık 6TiSCH Ağlarında Kaynak Yönetimi*, Yüksek Lisans Tezi, K.T.Ü., Fen Bilimleri Enstitüsü, Trabzon.

Park, T. (2020). *Distributed Wireless Resource Management in the Internet of Things*, Doctor of Philosophy in Electrical Engineering.

- Amini R., Imani M., Todorov I., & Ali M. (2020). *Performance Evaluation of Orchestra Scheduling in Time-Slotted Channel Hopping Networks*, DOI: 10.1109/ICCCKE50421.2020.9303678
- Madan R., ve Borran J., ve Sampath A.,ve Bhushan N., ve Khandekar A., & Ji T. (2010). *Cell association and interference coordination in heterogeneous LTE-A cellular networks*, IEEE Journal on selected areas in communications, 28,9,1479-1489.
- Liu X., Qin Z., Gao Y., & McCann J.A. (2019). *Resource Allocation in Wireless Powered IoT Networks*, 1-11.
- Chen H., Huang S., Zhang D., Xiao M., Skoglund M., & Poor V. (2021). *Federated Learning over Wireless IoT Networks with Optimized Communication and Resources*.
- Huang W., Wu M., Yang Z., & Sun K. (2022). *Self-Adapting Handover Parameters Optimization for SDN-Enabled UDN*, 1-31.
- Federico, O.S., Víctor, S.P., Javier, S.B.,& Jose, V.P. (2022). Scalability Enhancement on Software Defined Industrial Wireless Sensor Networks Over TSCH, DOI: 10.1109/ACCESS.2022.3212728, IEEE. Page(s): 107137 – 107151.
- Santos F.O., Payá V.S., Alberio T., & Blanes J.S. (2021). *Enhancing SDN WISE with Slicing Over TSCH*, 1-3.
- Tanganelli G., Virdis A., & Mingozzi E. (2019). *Implementation of Software-Defined 6LoWPANs in Contiki OS*, 1-3.
- Baddeley M., Nejabati R., Oikonomou G., Gormus S., Sooriyabandara M., & Simeonidou D. (2017). *Isolating SDN control traffic with layer-2 slicing in 6TiSCH industrial IoT networks*, 2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), Berlin, pp. 247-251.
- Luo, T., Tan, H.P., & Quek, T.Q.S. (2012). *Sensor OpenFlow: Enabling Software-Defined Wireless Sensor Networks*. IEEE Commun. Lett, 16, 1896–1899.

- Kermajani, H., & Gomez, C.(2014). On the Network Convergence Process in RPL over IEEE 802.15.4 Multihop Networks: Improvement and Trade-Offs. *Sensors* 2014, 14, 11993–12022.
- De Couto, D.S.J., Aguayo, D., Bicket, J., & Morris, R. A. (2003). High-throughput Path Metric for Multi-hop Wireless Routing. In *Proceedings of the 9th Annual International Conference on Mobile Computing and Networking (MobiCom'03)*, San Diego, CA, USA, pp. 134–146.
- De Oliveira, B.T., Alves, R.C.A., & Margi, C.B. (2015). Software-defined Wireless Sensor Networks and Internet of Things standardization synergism. In *Proceedings of the 2015 IEEE Conference on Standards for Communications and Networking (CSCN)*, Tokyo, Japan, pp. 60–65.
- Miguel, M. (2018). SD6WSN: A SDWSN Framework for 6LoWPAN Using Contiki and Node.js Controller. Available online: <https://github.com/marciolm/sd6wsn> (accessed on 1 September 2018). *Sensors* 2018, 18, 3738 23 of 23.
- Dunkels, A., Gronvall, B., & Voigt, T. (2004). Contiki—A lightweight and flexible operating system for tiny networked sensors. In *Proceedings of the 9th Annual IEEE International Conference on Local Computer Networks*, Washington, DC, USA, 16–18 November 2004; pp. 455–462.
- Osterlind, F., Dunkels, A., Eriksson, J., Finne, N., & Voigt, T. (2006). Cross-level sensor network simulation with cooja. In *Proceedings of the 2006 31st IEEE Conference on Local Computer Networks*, Tampa, FL, USA, pp. 641–648.
- Marcio L. F. M., Edgard J., Marcelo E. P., & Manoel C. P. (2018). SDN Architecture for 6LoWPAN Wireless Sensor Networks,1-23.
- URL-3. (2023) <https://github.com/contiki-ng/contiki-ng> Contiki-NG. 24 Eylül 2023 adresinden alınmıştır.
- URL-4. (2023) https://www.george-smart.co.uk/wsn/contiki_cooja_simulator Contiki Cooja Simulator 24 Eylül 2023 adresinden alınmıştır.

- Chalouf M.A. (2022). La gestion et le contrôle intelligents des performances et de la sécurité dans l'IoT, 118-119, London.
- Elsts A., Fafoutis X., Oikonomou G., & Piechocki R. (2017). Adaptive Channel Selection in IEEE 802.15.4 TSCH Networks, 1st Global Internet of Things Summit.
- Belavagi, M. C., & Muniyal, B. (2020). Multiple intrusion detection in RPL based networks. International Journal of Electrical and Computer Engineering (IJECE). doi:10.11591/ijece.v10i1.pp467-476
- Winter, T., Thubert, P., Brandt, A., Hui, J., Kelsey, R., Levis, P., & Alexander, R. (2012). RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks. Internet Engineering Task Force. <https://www.hjp.at/doc/rfc/rfc6550.html>
- Elsts A., Fafoutis X., Piechocki R., & Craddock I. (2017). Adaptive Channel Selection in IEEE 802.15.4 TSCH Networks., https://research-information.bris.ac.uk/ws/portalfiles/portal/110218275/tsch_channels_elsts2017.pdf
- Hermeto, R. T., Gallais, A., & Theoleyre, F. (2017). Scheduling for IEEE802.15.4-TSCH and Slow Channel Hopping MAC in Low Power Industrial Wireless Networks: A Survey. ICube Laboratory, CNRS / University of Strasbourg, Boulevard Sebastien Brant, 67412 Illkirch Cedex, France.
- Du P., & Roussos G. (2012). Adaptive time slotted channel hopping for wireless sensor networks, in 4th Computer Science and Electronic Engineering Conference (CEECE). IEEE, pp. 29–34.
- Li P., Vermeulen T., Liy H., & Pollin S. (2015). An adaptive channel selection scheme for reliable TSCH-based communication, in IEEE ISWCS, pp. 511–515.
- Duquennoy S., Elsts A., Nahas B.A., & Oikonomou G. (2017). TSCH and 6TiSCH for Contiki: Challenges, Design and Evaluation, in IEEE DCOSS.

- Aydın, B., Görmüş, S., Aydın, H., & Külçü, S. (2022). A new routing objective function for ietf 6tisch protocol. 2022 30th Signal Processing and Communications Applications Conference (SIU), IEEE, 2022, 1–4.
- Gomes, P. H., Watteyne, T., & Krishnamachari, B. (2017). MABO-TSCH: Multi-hop And Blacklist-based Optimized Time Synchronized Channel Hopping.
- Amezcuca Valdovinos, I., Figueroa Millán, P. E., Pérez-Díaz, J. A., & Vargas-Rosales, C. (2021). Distributed Channel Ranking Scheduling Function for Dense Industrial 6TiSCH Networks. *Sensors (Basel)*, 21(5), 1593. doi: 10.3390/s21051593. PMCID: PMC7956233, PMID: 33668770.
- Bektaş, M., & Görmüş, S. (2018). An energy efficient IoT protocol for healthcare at home. 26th Signal Processing and Communications Applications Conference (SIU), İzmir. s.1-4.
- Aydın, H. (2018). Ietf 6tisch protokolü için dağıtık kullanıcı kimlik doğrulama mekanizması (Yüksek Lisans Tezi), Karadeniz Teknik Üniversitesi, Fen Bilimleri Enstitüsü, Trabzon.
- Aydın, H., Goermues S., & Jin Y., A. (2018). Distributed User Authentication Mechanism for IETF 6TiSCH Protocol, 2018 IEEE 87th Vehicular Technology Conference (VTC Spring), IEEE.
- Aydın, H., & Bozkurt I. (2020). An Energy Efficient and Lightweight Cryptographic User Authentication Scheme for 6Tisch Networks, 2020 28th Signal Processing and Communications Applications Conference (SIU), IEEE.

ÖZGEÇMİŞ

Erkan DEMİR, Liseyi Trabzon Kanuni Anadolu Lisesi'nde, Lisans eğitimini ise Karadeniz Teknik Üniversitesi Bilgisayar Mühendisliği Bölümü'nü 2014 yılında bitirerek mezun oldu. Ankara'da Bilgisayar Mühendisi olarak özel yazılım şirketlerinde çalıştı, İstanbul'da bir vakıf üniversitesinde bir sene öğretim görevlisi olarak dersler verdi. 2020 yılında Karadeniz Teknik Üniversitesi Bilgisayar Mühendisliği Anabilim Dalı'nda yüksek lisans eğitimine başladı. Trabzon Teknokent'te özel bir yazılım şirketinde Yazılım Mühendisi olarak halen çalışmaktadır.

