
GAZİ ÜNİVERSİTESİ
Gazi
1926

KAMPÜS AĞLARINDA NESNELERİN İNTERNETİ İÇİN GENEL MİMARİ VE PROTOKOL TASARIMI

Erdal ÖZDOĞAN

DOKTORA TEZİ
BİLİŞİM SİSTEMLERİ ANA BİLİM DALI

GAZİ ÜNİVERSİTESİ
BİLİŞİM ENSTİTÜSÜ

HAZİRAN 2020

Erdal ÖZDOĞAN tarafından hazırlanan "KAMPÜS AĞLARINDA NESNELERİN İNTERNETİ İÇİN GENEL MİMARİ VE PROTOKOL TASARIMI" adlı tez çalışması aşağıdaki jüri tarafından OY BİRLİĞİ ile Gazi Üniversitesi Bilişim Sistemleri Ana Bilim Dalında DOKTORA TEZİ olarak kabul edilmiştir.

Danışman: Prof. Dr. O. Ayhan ERDEM

Bilgisayar Mühendisliği Ana Bilim Dalı, Gazi Üniversitesi

Bu tezin, kapsam ve kalite olarak Doktora Tezi olduğunu onaylıyorum. ...……..

Başkan: Prof. Dr. Kemal BIÇAKÇI

Bilgisayar Mühendisliği Ana Bilim Dalı, TOBB – Ekonomi ve Teknoloji Üniversitesi ……..…….

Bu tezin, kapsam ve kalite olarak Doktora Tezi olduğunu onaylıyorum.

Üye: Prof. Dr. Suat ÖZDEMİR

Bilgisayar Mühendisliği Ana Bilim Dalı, Hacettepe Üniversitesi

Bu tezin, kapsam ve kalite olarak Doktora Tezi olduğunu onaylıyorum. ...…….……...

Üye: Doç.Dr. İbrahim Alper DOĞRU

Bilgisayar Mühendisliği Ana Bilim Dalı, Gazi Üniversitesi

Bu tezin, kapsam ve kalite olarak Doktora Tezi olduğunu onaylıyorum. ...……

Üye: Dr. Öğr. Üyesi Ramazan KOCAOĞLU

Bilgisayar Mühendisliği Ana Bilim Dalı, OSTİM Teknik Üniversitesi

Bu tezin, kapsam ve kalite olarak Doktora Tezi olduğunu onaylıyorum. ...……

Tez Savunma Tarihi: 18/06/2020

Jüri tarafından kabul edilen bu tezin Doktora Tezi olması için gerekli şartları yerine getirdiğini onaylıyorum.

……..……

Doç. Dr. Aslıhan TÜFEKCİ

Bilişim Enstitüsü Müdürü

ETİK BEYAN

Gazi Üniversitesi Bilişim Enstitüsü Tez Yazım Kurallarına uygun olarak hazırladığım bu tez çalışmasında;

- Tez içinde sunduğum verileri, bilgileri ve dokümanları akademik ve etik kurallar çerçevesinde elde ettiğimi,
- Tüm bilgi, belge, değerlendirme ve sonuçları bilimsel etik ve ahlak kurallarına uygun olarak sunduğumu,
- Tez çalışmasında yararlandığım eserlerin tümüne uygun atıfta bulunarak kaynak gösterdiğimi,
- Kullanılan verilerde herhangi bir değişiklik yapmadığımı,
- Bu tezde sunduğum çalışmanın özgün olduğunu,

bildirir, aksi bir durumda aleyhime doğabilecek tüm hak kayıplarını kabullendiğimi beyan ederim.

Erdal ÖZDOĞAN

18 /06 / 2020

KAMPÜS AĞLARINDA NESNELERİN İNTERNETİ İÇİN GENEL MİMARİ VE PROTOKOL TASARIMI

(Doktora Tezi)

Erdal ÖZDOĞAN

GAZİ ÜNİVERSİTESİ

BİLİŞİM ENSTİTÜSÜ

Haziran 2020

ÖZET

Bilimsel gelişmeler ve buluşlar tarih boyunca her zaman insanlığın gelişimine katkıda bulunmuş ve yeni teknolojilerin gelişmesine ivme katmıştır. Birçok uzmana göre son altmış yılın en önemli buluşu olan İnternet de, kendisinden önceki teknolojik gelişmelerin bir ürünü ve kendisinden sonrakilerin de tetikleyicisi olmaktadır. İletişim alanındaki yeni gelişen teknolojiler sayesinde, her geçen gün daha fazla sayıda ve türde cihaz İnternet'e bağlanabilmekte, veri üretmekte ve "akıllı" hâle gelmektedir. Araştırmalar, 2020 yılı itibariyle 20 ile 30 milyar arasındaki nesnenin çeşitli teknolojiler ile İnternet'e bağlanacağını öngörmektedir. İnternet'in gelişimine bakıldığında, kullanıcı odaklı olmaktan çıkıp nesne odaklı olmaya doğru gittiğini görmek mümkündür. Nesnelerin İnterneti, bu değerler dizisi doğrultusunda insanı, nesneyi, veriyi ve veri işlemeyi bir araya getiren bir kavram olarak karşımıza çıkmaktadır. Bu tez çalışması kapsamında, akıllı şehirlerin küçük bir ölçeği olarak görülebilen kampüs ağlarında yer alan nesnelerden elde edilen ortama ilişkin verilerin bilgisayar ağlarına taşınabilmesini sağlayan, var olan yaklaşımların karması bir yaklaşım geliştirilmiştir. Bu yaklaşım doğrultusunda bir uygulama protokolü ve protokolün çalışmasına destek olacak bir mimari tasarlanmıştır. Protokolün çalışma prensibinin ve işlevselliğinin incelenmesi amacıyla gerçek bir test ortamında değerlendirmeleri yapılmıştır. Ayrıca, çok sayıda cihazın bulunduğu ortamlarda protokolün davranışının analiz edilebilmesi için benzetim yazılımı ile performans ve işlevsellik değerlendirmesi yapılmıştır. Çalışma kapsamında geliştirilen protokolün, sınırlı sayıda cihazın bulunduğu orta ölçekli ağlarda daha hızlı çalıştığı, daha az bant genişliği tükettiği ve merkezi sunucudan kaynaklanan hata durumlarında iletişimi devam ettirdiği görülmüştür.

Bilim Kodu	:	92407
Anahtar Kelimeler	:	Nesnelerin İnterneti, Protokol Tasarımı, IoT Uygulama Protokolü, Hibrit Uygulama Katmanı Protokolü
Sayfa Adedi	:	96
Danışman	:	Prof. Dr. O. Ayhan ERDEM

GENERAL ARCHITECTURE AND PROTOCOL DESIGN FOR THE INTERNET OF THINGS IN CAMPUS NETWORKS

(Ph. D. Thesis)

Erdal ÖZDOĞAN

GAZİ UNIVERSITY

INSTITUTE OF INFORMATICS

June 2020

ABSTRACT

Scientific research, studies and innovation historically have served development and progress of humanity and accelerated development of new technologies. Internet, noted by experts as the top invention of the last 60 years, is also a product of previous technologies before its time and also is a driver of many that follows. With the newly developing technologies in the field of communications, the number of devices and type of devices that connect to Internet, that output data and become smart are multiplying in size, each day. Research suggests by year 2020, 20-30 billion devices will connect to Internet leveraging a wide range of interoperability technologies. Internet is evolving from user centric to "thing" centric as we look into its progress and underlying trends. Under the light of these concepts, Internet of Things as a technology concept embraces human, devices, data and data analytics under its umbrella. In this thesis study, a new approach, leveraging existing methodologies, is being proposed to incorporate data that is produced via the devices on the campus networks into the computer networks, modeling campus as a smaller scale smart city. In line with this approach, an application protocol and an infrastructure to empower it was developed. In order to assess how the protocol operates and how effective it is, a real case test has been leveraged. Furthermore, to analyze the behavior of the protocol in cases with high number of devices, an assessment was conducted through simulation software for performance and functionality. Protocol proposed within this thesis study is observed to operate faster in medium size networks with limited number of connected devices. It even leads to less bandwidth consumption and sustains communications in case of central server connection problems.

Science Code	:	92407
Key Words	:	Internet of Things, Protocol Design, IoT Application Protocol, Hybrid Application Layer Protocol
Page Number	:	96
Supervisor	:	Prof. Dr. O. Ayhan ERDEM

TEŞEKKÜR

Bu tez çalışması boyunca değerli katkıları ve yardımları ile beni yönlendiren, motive eden tecrübelerini her fırsatta aktaran değerli danışman hocam Prof. Dr. O. Ayhan ERDEM'e, tezin oluşmasında şekillenmesinde değerli desteklerini aldığım Prof. Dr. Suat ÖZDEMİR ve Prof. Dr. Kemal BIÇAKÇI hocalarıma, ayrıca bu zorlu süreçte gösterdikleri sabır ve destekleri için aileme teşekkürü bir borç bilirim.

İÇİNDEKİLER

Sayfa

ÖZET iv

ABSTRACT v

TEŞEKKÜR vi

İÇİNDEKİLER vii

ÇİZELGELERİN LİSTESİ x

ŞEKİLLERİN LİSTESİ xi

SİMGELER VE KISALTMALAR xiv

1. GİRİŞ 1

2. NESNELERİN İNTERNETİ VE TEMEL KAVRAMLAR 11

2.1 Nesnelerin İnterneti 11

2.2 Nesnelerin İnterneti Teknolojileri 13

2.2.1 IoT sisteminde sensörler 15

2.2.2 Aktüatörler (Eyleyiciler) 15

2.2.3 Denetleyiciler 16

2.3 Nesnelerin İnternetinde Kablosuz Teknolojiler 18

2.3.1 Radyo frekans tanımlayıcı (RFID- Radio frequency identification) 19

2.3.2 Kablosuz yerel alan ağları (Wi-Fi) 19

2.3.3 ZigBee 20

2.3.4 Diğer kablosuz teknolojiler 21

2.4 IoT Mimarileri 22

2.4.1 Üç katmanlı mimari 22

2.4.2 Beş katmanlı mimari 23

2.4.3 Referans modelleri 25

Sayfa

2.5 IoT Protokolleri 27
2.6 Uygulama Protokolleri 28
2.6.1 Message queuing telemetry transport -MQTT 29
2.6.2 Constrained application protocol- CoAP 33
2.6.3 Advanced message queuing protocol –AMQP 36
2.6.4 Extensible messaging and presence protocol - XMPP 37
2.6.5 Data distribution service - DDS 37
3. GENEL AMAÇLI IOT UYGULAMA PROTOKOLÜ TASARIMI 39
3.1 Genel Özellikler 39
3.2 Genel Çalışma Mimarisi 41
3.3 Paket Türleri 44
3.4 hIoT Paket Başlığı 45
3.5 hIoT Çalışma Aşamaları 46
3.5.1 Servis kayıt aşaması 47
3.5.2 Servis keşif aşaması 50
3.5.3 Veri transferi aşaması 51
3.5.4 Diğer hIoT paketleri 53
3.6 Mimaride Kullanılan Bileşenler 55
3.6.1 İstemci Bileşeni 55
3.6.2 IoT koordinatör bileşeni 58
3.6.3 IoT Ağ geçidi bileşeni 60
3.7. Performans Değerlendirmeleri 62
3.7.1 Farklı bant genişliklerinde yaşanan gecikmeler 63
3.7.2 Taşınan yük miktarlarında yaşanan gecikmeler 64

Sayfa

3.7.3 Servis keşfi değerlendirmesi 66
3.7.4 Bant genişliği kullanımı değerlendirmesi 67
3.8. Benzetim Modellemesi 68
3.8.1 Düğüm modeli 69
3.8.2 Gecikme 71
3.8.3 Sunucu trafiği karşılaştırma 72
3.8.4 Bant genişliği kullanımı ve işlevsellik karşılaştırmaları 74
4. SONUÇ VE ÖNERİLER 79
KAYNAKLAR 82
EKLER 90
EK-1 IoT Koordinatör Örnek Kodu 91
ÖZGEÇMİŞ 95

ÇİZELGELERİN LİSTESİ

Çizelge **Sayfa**

Çizelge 2.1. WiFi standartları 20

Çizelge 2.2. IoT Dünya forumu referans modeli 26

Çizelge 3.1. Geliştirilen protokolün paket türleri 45

Çizelge 3.2. Bant genişliğine bağlı gecikme değerleri 63

Çizelge 3.3. Taşınan yük miktarlarına bağlı gecikme değerleri 65

Çizelge 3.4. Bant genişliğine bağlı ortalama keşif süreleri 66

Çizelge 3.5. MQTT aşamalarında taşınan yük miktarları 70

ŞEKİLLERİN LİSTESİ

Şekil **Sayfa**

Şekil 2.1. Nesnelerin İnterneti ekosisteminin genel yapısı 14

Şekil 2.2. Mikrodenetleyici olarak kullanılan Arduino 17

Şekil 2.3. Tek kartlı bilgisayar, Raspberry Pi 17

Şekil 2.4. Bant genişliği kullanımı ve kapsama alanına göre kablosuz teknolojiler 18

Şekil 2.5. Zigbee topolojileri ve zigbee rolleri 20

Şekil 2.6. Üç katmanlı IoT mimarisi..... 22

Şekil 2.7. Beş katmanlı IoT mimarisi 24

Şekil 2.8. ITU - IoT referans modeli 25

Şekil 2.9. IoT protokollerinin OSI modelindeki gösterimi 27

Şekil 2.10. MQTT protokolünde bileşenler 30

Şekil 2.11. MQTT paket türleri 32

Şekil 2.12. MQTT başlık yapısı..... 33

Şekil 2.13. CoAP iletişim örneği 35

Şekil 2.14. AMQP protokolü 36

Şekil 2.15. XMPP protokolü..... 37

Şekil 2.16. DDS abone ve yayıncıları..... 38

Şekil 3.1. Sensöre ilişkin alınan veriler 40

Şekil 3.2. Hibrit çalışma modeli 41

Şekil 3.3. Geliştirilen protokole genel bakış..... 42

Şekil 3.4. Geliştirilen protokol mimarisi 43

Şekil 3.5. Protokolde kullanılan aşamalar 44

Şekil 3.6. Paket başlık yapısı 45

Şekil 3.7. Servis kayıt aşaması..... 47

Şekil **Sayfa**

Şekil 3.8. Servis kayıt paket başlık yapısı 48

Şekil 3.9. Kayıt onay paketi başlık bilgisi 48

Şekil 3.10. Kayıt silme süreci 49

Şekil 3.11. Kayıt silme sürecinde paket başlığı 50

Şekil 3.12. Servis keşif aşaması 51

Şekil 3.13. Servis keşif aşamasında kullanılan paketler 51

Şekil 3.14. Doğrudan veri iletişi ve kullanılan paketler 52

Şekil 3.15. Sunucu tabanlı iletişim gösterimi 52

Şekil 3.16. Sunucu tabanlı erişimde kullanılan istek paketleri 53

Şekil 3.17. Sunucu tabanlı iletişimde kullanılan cevap paketleri 53

Şekil 3.18. Kontrol süreci 54

Şekil 3.19. Kontrol sürecinde kullanılan paketler 54

Şekil 3.20. İstemci bileşeni akış diyagramı 56

Şekil 3.21. İstemci bileşeni, sorgu paketi gönderim akış diyagramı 57

Şekil 3.22. İstemci bileşeni veri iletim akış diyagramı 58

Şekil 3.23. Koordinatör bileşeni akış diyagramı 59

Şekil 3.24. IoT Koordinatörü akış diyagramında modüler yapı 60

Şekil 3.25. IoT Ağ geçidi bileşeni, sonlu durum akış diyagramı 61

Şekil 3.26. Deneysel ortam topolojisi 62

Şekil 3.27. Bant genişliğine göre gecikme değerleri grafiği 64

Şekil 3.28. Paket boyutuna göre gecikme değerleri grafiği 65

Şekil 3.29. MQTT trafiği için WireShark bant genişliği grafiği 67

Şekil 3.30. hIoT Protokolü Wireshark bant genişliği tüketim grafiği 68

Şekil 3.31. OPNET benzetim topolojisi 69

Şekil **Sayfa**

Şekil 3.32. Geliştirilen protokoldeki uyarlanan düğüm modeli 70

Şekil 3.33. MQTT aşamalarının OPNET'de tanımlanması 71

Şekil 3.34. hIoT protokolünün aşamalarının OPNET'de tanımlanması 71

Şekil 3.35. MQTT ve hIoT protokolleri Ethernet gecikme değerleri karşılaştırması 72

Şekil 3.36. Sunucu trafiği grafiği 73

Şekil 3.37. Sensör düğümleri ile sunucu UDP trafiklerinin karşılaştırılması 73

Şekil 3.38. Sensör ağında bant genişliği kullanımı 74

Şekil 3.39. Cihaz sayısı arttıkça uçtan uca gerçekleşen iletişim trafiğinin değişimi 75

Şekil 3.40. Sunucu arızasında trafik akışları 76

Şekil 3.41. Sensör düğümlerinde CPU kullanımı 77

SİMGELER VE KISALTMALAR

Bu çalışmada kullanılmış simgeler ve kısaltmalar, açıklamaları ile birlikte aşağıda sunulmuştur.

Simgeler	**Açıklamalar**
bps	Saniye başına bit sayısı
hz	Hertz

Kısaltmalar	**Açıklamalar**
ACK	Onay (Acknowledgement)
AMQP	Gelişmiş Mesaj Kuyruklama Protokolü (Advanced Message Queuing Protocol)
API	Uygulama Programlama Arayüzü (Application Programming Interface)
BLE	Düşük Enerjili Bluetooth (Bluetooth Low Energy)
CAN	Kampüs Alan Ağları (Campus Area Network)
CoAP	Kısıtlı Uygulama Protokolü (Constrained Application Protocol)
DC	Doğrudan İletişim (Direct Communication)
DDS	Veri Dağıtım Sistemi (Data Distribution System)
DNS	Etki Alanı Sistemi (Domain Name System)
DNS-SD	DNS Hizmet Klasörü (DNS Service Directory)
hIoT	Hibrit IoT Protokolü
IEEE	Elektrik Elektronik Mühendisleri Enstitüsü (Institute of Electrical and Electronics Engineers)

Kısaltmalar	Açıklamalar
IETF	Internet Mühendisliği Görev Gücü (Internet Engineering Task Force)
IoT	Nesnelerin İnterneti (Internet of Things)
IoTGW	IoT Ağ Geçidi (IoT Gateway)
IP	Internet Protokolü
IPv4	Internet Protokolü sürüm 4
IPv6	Internet Protokolü sürüm 6
ITU	Uluslararası Telekomünikasyon Birliği (International Telecomunication Union)
LAN	Yerel Alan Ağları (Local Area Network)
LPWAN	Düşük Güçlü Geniş Alan Ağları
M2M	Makinadan Makinaya (Machine to Machine)
M2P	Makineden İnsana (Machine to People)
M2S	Makineden Sunucuya (Machine to Server)
MCU	Mikrodenetleyici
mDNS	Çoklu Yayın DNS (Multicast DNS)
MQTT	Mesaj Kuyruklama Telemtri İletimi (Message Queuing Telemetry Transport)
OSI	Açık Sistem Arabağlantısı (Open System Interconnection)
OMG	Nesne Yönetim Gurubu (Object Management Group)
PAN	Kişisel Alan Ağları (Personal Area Network)
PC	Kişisel Bilgisayar (Personal Computer)
Pub/Sub	Yayın / Abone (Publish / Subscribe)
QoS	Hizmet Kalitesi (Quality of Service)
REST	Temsili Durum İletimi (Representational State Transfer)

Kısaltmalar	Açıklamalar
RFID	Radyo Frekans Tanımlayıcı (Radio Frequency Identification)
SDN	Yazılım Tanımlı Ağ (Software Defined Network)
SPoF	Tek Bir Arıza Noktası (Single Point of Failure)
SQL	Yapılandırılmış Sorgu Dili (Structured Query Language)
TCP	İletim Kontrol Protokolü (Transmission Control Protocol)
UDP	Kullanıcı Datagram Protokolü (User Datagram Protocol)
URI	Evrensel Kaynak Tanımlayıcı (Universal Resource Identifier)
WAN	Geniş Alan Ağları (Wide Area Network)
WiFi	Kablosuz Sadakat (Wireless Fidelity)
XMPP	Genişletilebilir Mesajlaşma ve Varlık Protokolü **(**Extensible Messaging and Presence Protocol)

1. GİRİŞ

Son yıllarda dünyada teknolojik anlamda büyük değişimler yaşanmakta, bu değişimler her insanı, kurumu, organizasyonu, hatta sosyal ve ekonomik yapıyı etkilemektedir. İletişim alanında yaşanan mobil ve kablosuz teknolojilerin gelişimi, her ortamdan İnternet'e erişimi kolaylaştırmıştır. İnternet'in yaygın olarak kullanılmasının ve bant genişliklerinin giderek artmasının bir sonucu olarak fiziksel dünya ile bilgisayar ağları üzerinden iletişim kurmanın yolu açılmıştır. Daha önce bağlı olmayan nesnelerin de çeşitli teknolojiler ile İnternet'e bağlanması sayesinde, İnternet evriminin yeni aşaması olarak adlandırılan Nesnelerin İnterneti'ne (IoT- Internet of Things) geçiş yapıldığı söylenebilir. IoT ile İnternetin kullanım senaryosu, insan merkezli olmaktan çıkmış, cihaz ve nesne merkezli olmaya doğru kaymıştır. Nesnelerin İnterneti, çok çeşitli heterojen kaynaklardan verilerin elde edilmesi, zeki sistemler ve yaklaşımlar ile bu çok çeşitli verilerden anlamlı bilgilerin elde edilmesi ile ilgilidir. İnternet erişim yöntemlerindeki teknolojik gelişmeler, yarı iletken teknolojisindeki ucuzlama, iletişim bant genişliklerinin giderek artması ve her yerden İnternet'e erişebilme gibi nedenler, insanların fiziksel dünya ile iletişim kurmasına ve onlardan veriler elde etmesine imkân sunmaktadır.

İnternet ve ağ kavramı söz konusu olduğunda yalnızca bilgisayarlardan, sunuculardan ve kullanıcılardan değil, sensörler ve Radyo Frekans Tanımlayıcı (RFID-Radio Frequency Identification) etiketler gibi teknolojilerin desteği ile fiziksel dünyadaki nesnelerden de gerçek zamanlı veriler elde edilebilmektedir. İnternet'e bağlı olan nesnelerin sayılarının giderek artmasıyla, önümüzdeki yıllarda IoT'nin tüm sektörlere etki edeceği beklenmektedir.

Bilgi teknolojisinin ve İnternet'in sürekli gelişimi sayesinde Nesnelerin İnterneti küresel olarak ilgi odağı haline gelmekte ve İnternet'ten sonra en önemli teknolojik yeniliklerden biri olarak kabul edilmektedir [1]. Nesnelerin İnterneti hızla yaygınlaşmaya ve gelişmeye devam ederken akıllarda bıraktığı önemli bir soru da, farklı tekniklerle bilgi toplayıp, işleyen ve bunları kullanan ürün ve uygulamaların birbirlerine nasıl ve hangi ortak dille bağlanacağıdır [2]. Önümüzdeki yıllarda giderek artan sayıdaki ve türdeki donanımların doğrudan veya çeşitli teknolojiler aracılığıyla İnternet'e bağlanacağı tahmin edilmekte, bu durum doğal olarak Internet Protokolü (IP) tabanlı çalışan protokollerin geliştirilmesi ihtiyacını doğurmaktadır [3].

Ağa bağlı milyarlarca donanımın ya da diğer bir ifade ile nesnenin çok büyük boyutlarda veri üreteceği beklenmektedir [4]. Farklı türde ve yapıda üretilen bu büyük verinin (big data) yapılandırılması, kullanılması ve bilgiye dönüştürülmesi IoT'nin çözmesi gereken en büyük sorunlardan biridir. İnternet ortamında IoT öncesine kadar kullanılan protokollerin bu sorunu karşılamadaki yetersizlikleri, yeni yaklaşımların geliştirilmesini sağlamıştır. IoT ekosisteminde kullanılan donanımların çoğunlukla sınırlı yeteneklere sahip olması, yeni uygulama protokollerinin geliştirilmesine ya da var olan protokollerin IoT gereksinimlerine göre uyarlanmasına neden olmuştur [5]. IoT'nin kişisel kullanımdan, kurumsal kullanıma kadar birçok farklı uygulama alanının olması, yeni yaklaşımların da geliştirilmesi gerektiğini doğuracaktır. Bu nedenle çeşitli kullanım senaryolarını desteklemek amacıyla yeni yöntemler ve araştırmalar yapılması gerekmektedir [6]. Öte yandan, Uluslararası İletişim Birliği'nin (International Telecommunication Union -ITU) çalışma raporunda [7], yeni protokollerin geliştirilmesi gerekliliği ifade edilmiştir. Yine, ağa bağlı cihaz sayılarının ve türlerinin giderek artması, yeni protokollerin ve senaryoların geliştirilmesini tetiklemektedir [8].

Nesnelerin İnterneti uygulamalarında yaygın olarak kullanılan uygulama protokolleri çalışma prensipleri itibariyle incelendiğinde, sunucu tabanlı çalışan protokoller ve doğrudan iletişim yöntemiyle çalışan protokoller (merkezi ve merkezi olmayan) olmak üzere iki kategoride toplanabilir [9]. Hem sunucu tabanlı yöntemlerin, hem de doğrudan iletişimi destekleyen yöntemlerin her ikisinin de çeşitli avantajlı ve dezavantajlı yönleri bulunmaktadır [10]. Sunucu tabanlı çalışan protokoller ele alındığında, sensörlerden elde edilen verilerin transferinde sunucu aktif bir rol oynamaktadır. Ancak veri iletişiminde sunucuya olan sürekli bağımlılık, tek bir hata noktasının (SPoF - Single Point of Failure) oluşmasına neden olmaktadır [11-12]. Ayrıca veri iletiminde merkezi bir sunucunun kullanımı, kaynakların nesnelere dağıtıldığı IoT'nin dağıtık doğasına aykırı olmaktadır. Yine bu yöntemle cihazlara doğrudan erişim sağlanamadığından, arada ek bir cihazın varlığı göreli olarak ek gecikmelere neden olmaktadır [13]. Doğrudan iletişim yönteminin de işlemci tüketiminde üstel artış gibi çeşitli performans sorunları gösterdiği, akademik çalışmalarda ortaya konmaktadır [9]. Yine, özellikle verilerin depolanması ya da buluta gönderilmesi ihtiyacının olduğu ya da daha güçlü donanımlar tarafından hassas karar mekanizmalarının uygulanması gerektiği durumlarda [14] ve nesnelere erişimin kimlik doğrulama ile kısıtlanması gerektiği durumlarda, merkezi olmayan yöntem yetersiz kalmaktadır.

IoT uygulamaları için en uygun uygulama katmanı protokolünün belirlenmesi, bant genişliği kullanımı, paket kayıp olasılığı, uçtan uca gecikme süreleri, kullanılan net genişlik (throughput) ve iletilen paketin boyutu gibi birçok parametreye bağlıdır [11,15]. Ancak bu parametrelerin çoğu, ağın anlık durumuna göre değişkenlik göstermektedir. Bu durumda seçilen uygulama protokolü, ağın o anki durumuna göre ya da parametrelerde değişiklik olduğunda performans kayıplarına neden olabilmektedir. Bu nedenle, merkezi yaklaşım ile doğrudan yaklaşım arasında dinamik geçiş yapabilen, ya da ağın farklı ihtiyaçlarını karşılayabilmek amacıyla, hem doğrudan erişimi hem de sunucu aracılığı ile erişimi destekleyen protokollerin geliştirilmesine ihtiyaç duyulmaktadır. Akademik araştırmalarda [16,17], bu iki yaklaşımı temsil eden protokollerin birlikte çalışabilmesine yönelik çalışmalar yapılmış, ancak mesajların dönüştürülmesinde performans kayıpları yaşandığı belirtilmektedir. Öte yandan, sınırlı kaynaklara sahip IoT cihazlarının bulunduğu ortamda, iki farklı protokolün kullanımı hem verimli bir yaklaşım olmamakta hem de yönetimsel olarak zorluk çıkarmaktadır. Bu nedenle, mevcut IoT uygulama protokollerinin yanı sıra, her iki yöntemi de destekleyen bir protokol geliştirilmesi gerekmektedir.

Bu tez çalışmasında, Nesnelerin İnterneti için kampüs alan ağlarında (CAN- Campus Area Network) kullanılmak amacıyla, hem merkezi sunucu yaklaşımını hem de doğrudan erişim yaklaşımını kullanabilen karma bir uygulama protokolü (hIoT- Hibrit IoT Protokolü) geliştirilmiştir. Geliştirilen uygulama katmanı protokolü, gerçek ortamda sınırlı sayıda cihaz üzerinde test edilmiş ve performans değerlendirmeleri yapılmıştır. Yine çok sayıda nesnenin bulunduğu ortamlarda protokolün çalışmasının değerlendirilmesi amacıyla, OPNET benzetim yazılımı ile bant genişliği kullanımı, gecikme ve cihaz sayısına bağlı kaynak tüketimi gibi yönlerden performans değerlendirmesi yapılmıştır. Performans değerlendirilmelerinin yanı sıra, işlevsellik yönünden yaygın olarak kullanılan protokol ile farklı yönleri ortaya konmuştur.

Bu tez çalışması kapsamında geliştirilen protokol ve protokolün çalışması için gerekli olan mimari tasarlanırken aşağıda belirtilen varsayımlar dikkate alınmıştır.

- Protokol, akıllı şehirlerin küçük bir prototipi olarak algılanabilecek, birden fazla yerel alan ağından (LAN-Local Area Network) oluşan kampüs alan ağları için tasarlanmıştır. Bu nedenle, en sağlıklı çalışması bu ortamlarda olmaktadır.

- Çalışma içerisinde, sensörlerden elde edilen bilgilerin genellikle küçük boyutlarda olduğu gerçeği dikkate alınarak, IP paketinin içerisinde taşınması sırasında paketlerin parçalara (fragmantasyon) ayrılmadığı varsayılmıştır.
- Nesnelerin İnternetinde güvenlik son derece önemli bir konu olmasına rağmen, bu çalışmada güvenlik kapsam dışında tutulmakta ve ortamın güvenli olduğu farz edilmektedir.
- Geliştirilen protokol, Açık Sistem Arabağlantısı (OSI - Open System Interconnection) referans modelinin yedinci katmanında çalışan bir uygulama protokolüdür. Bu nedenle referans modelinin alt katmanlarında kullanılan teknolojilerden ve protokollerden bağımsızdır.

Nesnelerin İnterneti için geliştirilen genel amaçlı hibrit uygulama protokolünün tasarlanmasında sınırlılıklar aşağıda belirtilmektedir:

- Geliştirilen protokolün gerçek ortamda değerlendirilmesi için Arduino ve Raspberry-Pi donanımları kullanılmıştır. Gerçek ortam uygulamaları yapılırken, bu donanımların yetenekleri doğrultusunda değerlendirmeler yapılmıştır. Gerçek ortamda sınırlı sayıda donanım bulunduğundan, değerlendirmeler birden fazla kez yapılmakta ve sonuçlar ortalama değerler üzerinden verilmektedir. Daha fazla sayıda cihazın bulunduğu durumda protokolün davranışının izlenmesi için OPNET benzetim yazılımı kullanılmıştır.
- Geliştirilen protokolün değerlendirmeleri, kablolu Ethernet ağında IPv4 protokolü üzerinden gerçekleştirilmiştir. IPv6 ve kablosuz ortam değerlendirmeleri yapılmamıştır.
- Sensörlerden elde edilen verilerin genellikle küçük boyutlarda olduğu dikkate alınarak, tek bir paket içerisinde taşınan verinin boyutu en fazla 1024 bayt olacak şekilde sınırlandırılmıştır.

Tez çalışmasına konu olan uygulama katmanı protokolleri incelenmiş ve Nesnelerin İnterneti'ndeki kullanım alanları, çalışma prensipleri, işlevsellikleri, güçlü ve zayıf yönleri incelenmiştir. Protokoller çalışma prensiplerine göre merkezi sunucu aracılığı ile iletişim kuranlar ve doğrudan iletişim kuranlar olmak üzere iki kategoride toplanmıştır. Geliştirilen uygulama protokolü her iki yaklaşımı birden sağlayan hibrit bir yapıda geliştirilmiştir.

Çalışmada, kampüs ağlarında kullanılabilecek sensörlerden ne tür verilerin elde edilebileceği incelenmiş ve bu doğrultuda paket yapısı tasarlanmıştır. Protokolün çalışabilmesi için ihtiyaç duyulan bileşenler, diğer uygulama protokolleri ile uyumlu olacak şekilde belirlenmiştir. Geliştirilen protokolün performansı ve işlevselliği hem gerçek ortamda hem de benzetim yazılımı ortamında değerlendirilmiştir.

Son yirmi yılda Nesnelerin İnterneti endüstri, akademi ve standart geliştirilen kuruluşlar için önemli bir araştırma ve geliştirme ortamı haline gelmiştir [19]. Genel bir ifade ile Nesnelerin İnterneti, çeşitli teknolojiler kullanılarak, fiziksel nesnelerin bilgisayar ağlarına bağlanmasıdır. Bu amacı gerçekleştirmek için çok çeşitli teknolojiler ve çözümler geliştirilmektedir. Ancak bağlanan nesne türünün farklılıkları, bağlı nesne sayılarının giderek artması ve IoT'nin doğası gereği heterojen bir yapıda bulunması, çeşitli zorlukları da beraberinde getirmektedir. Yine IoT için genel kabul görmüş bir mimarinin ya da modelin olmayışı, çeşitli alanlarında farklı uygulamaların geliştirilmesine neden olmuştur. Aynı şekilde bu alanda çalışma yapan birçok üretici de, kendi IoT çözümlerini geliştirmektedirler. Değişken ihtiyaçlar ve çok çeşitli kullanım senaryoları, farklı uygulamaların ve bunun bir uzantısı olarak da farklı uygulama protokollerinin geliştirilmesini tetiklemiştir. Bu protokoller ile ilgili olarak birçok bilimsel araştırma yapılmaktadır. Nesnelerin İnternetinde uygulama protokolleri ile ilgili yapılan çalışmalar incelendiğinde, protokollerin performans karşılaştırmaları, protokollerin geliştirilmesine ve birlikte çalışabilmesine yönelik çalışmalar ile protokollerin uygulanmasına ilişkin çalışmalar olmak üzere üç farklı grupta toplandığı görülmektedir. Bu bölümde, IoT Uygulama Katmanı Protokolleri üzerine yapılan çalışmalar, tezin ele aldığı problem durumu çerçevesinde incelenmiştir.

IoT uygulama protokolleri ile ilgili yapılan akademik çalışmalar incelendiğinde, protokollerinin performanslarının değerlendirilmesi konularında yapılan çalışmaların daha yoğun olduğu görülmektedir. Akademik çalışmalarda çoğunlukla, Advanced Message Queuing Protocol (AMQP), Constrained Application Protocol (CoAP), Message Queuing Telemetry Transport (MQTT) ve Extensible Messaging and Presence Protocol (XMPP) gibi en yaygın kullanılan IoT protokolleri karşılaştırılmıştır. MQTT protokolü ile CoAP protokolünün, noktadan noktaya gecikme değerleri ile bant genişliği kullanımı yönünden karşılaştırıldığı bir çalışmada [18], MQTT iletileri, düşük paket kaybı oranlarında CoAP iletilerinden daha düşük gecikmeye sahip olduğu; yüksek kayıp oranlarında ise CoAP iletilerinden daha yüksek gecikmeye sahip olduğu ifade edilmiştir. Laboratuvar ortamında

hücresel ağlar üzerinden, iletim zamanlarını değerlendirmek ve optimizasyon potansiyellerini analiz etmek için taşıma mekanizmaları bakımından CoAP, MQTT ve OPC Birleşik Mimari (OPC UA) gibi Makineden Makineye (M2M-Machine-to-Machine) protokollerin karşılaştırıldığı çalışmada [19], yüksek yük taşıma özelliğine rağmen OPC UA protokolünün daha başarılı olduğu ifade edilmiştir. Enerji verimliliği ve kaynak tüketimi (işlemci ve bellek) yönünden performans karşılaştırmalarının yapıldığı Mun ve diğerlerinin yaptığı bir çalışmada, 1024 bayt değerinden yüksek veri taşıması yapıldığında CoAP protokolünün enerji tüketiminin arttığı ifade edilmektedir. Aynı çalışmada, bir uygulama MQTT protokolünü kullanıyorsa, performans ve enerji verimliliği için paket boyutunun en fazla 1,5 KB olması gerektiği ifade edilmektedir [20]. Çeşitli M2M etkileşimleri gerektiren bir uygulamanın, IoT cihazları için ilke tabanlı bir erişim kontrol çerçevesi tasarlama ve uygulama bağlamında değerlendirildiği bir çalışmada [21], protokoller işlemci ve bellek kullanımı yönünden incelenmektedir. Çalışmada MQTT protokolünün CoAP'a göre işlemciyi daha fazla kullandığı, ancak daha az bellek tükettiği ifade edilmektedir. AMQP, CoAP ve MQTT protokollerinin mesaj boyutu ve paket kaybı metrikleri açısından performans değerlendirmelerinin yapıldığı deneysel bir çalışmada [22], CoAP protokolünün en iyi sonuçlar verdiği ifade edilmektedir. Performans karşılaştırmaları kapsamında yapılan araştırmalardan biri de, sensör mesajlarının gecikmesinin ve net genişlik değerlerinin web performansı açısından incelendiği çalışmadır [23]. İlgili çalışmada MQTT protokolünün çok çeşitli IoT Web uygulamaları için en uygun mesajlaşma protokolü olduğu belirtilmektedir. Luzuriaga ve diğerlerinin yaptığı çalışmada [24], IoT uygulama katmanı protokolleri bant genişliği kullanımı, enerji verimliliği, uçtan uca gecikme, mesajların iletim zamanlamasındaki değişim ve güvenlik parametreleri yönünden değerlendirilmiştir. Güvenlik açısından AMQP protokolünün başarılı olduğu, enerji verimliliği açısından ise MQTT'nin daha yüksek performans gösterdiği belirtilmektedir. CoAP ve MQTT uygulama protokollerinin karşılaştırıldığı, aynı boyuttaki verilerin taşındığı diğer bir çalışmada [25] MQTT'ye oranla CoAP protokolünün daha yüksek performans ortaya koyduğu ifade edilmektedir. Yoğun paket akışının gerçekleştiği ortamlar ile ilgili yapılan diğer bir karşılaştırmada ise [26], MQTT protokolünün daha yüksek net genişlik değerine ve daha küçük gecikme değerine sahip olduğu belirtilmektedir.

Performans karşılaştırmalarına yönelik yapılan diğer çalışmalarda [9,27–30] çeşitli değerlendirme ölçütleri ve performans metrikleri kullanılmıştır. Tüm bu çalışmalardan görüleceği üzere, en uygun uygulama katmanı protokolü, IoT'nin uygulama alanına, ağın

kullanılan bant genişliği ya da gecikme gibi anlık durumuna ve çeşitli gereksinimlere göre değişkenlik göstermektedir. Kimi durumlarda doğrudan erişim yapabilen CoAP gibi protokoller başarılı iken, kimi durumda ise MQTT gibi merkezi sunucu yaklaşımı sergileyen protokoller başarılı olmaktadır.

Nesnelerin İnternetinde kullanılan uygulama protokollerinde hâlen değişiklikler ve güncellemeler yapılmaktadır. Aynı şekilde bu protokollerde geliştirmelere yönelik olarak çeşitli akademik çalışmalar da yapılmaktadır. Choi ve diğerleri tarafından yapılan bir çalışmada [31], CoAP ve MQTT protokollerinin özellikle sensör sayısının fazla olduğu ağ ortamlarında ölçekleme konusunda yetersiz olduğuna değinilmektedir. İlgili çalışmada, CoAP protokolünün çalışma mekanizması üzerinde geliştirme yapılmakta ve CoAP istemcilerinin bir küme oluşturarak şekilde gruplandırılması sağlanmaktadır. Her bir istemcinin bağımsız olarak CoAP iletileri göndermesi yerine, bir temsilci aracılığıyla iletilerin gönderilmesi ve daha az bant genişliği tüketimi sağlanmaktadır. Uygulama protokolü tasarımına ilişkin yapılan bir çalışmada ise [32], yazarlar tarafından önerilen User Datagram Protocol (UDP) tabanlı protokol, diğer IoT protokolleri ile karşılaştırılmaktadır. Geliştirilen protokolün, paket kayıplarının önceden kestirilemeyen ağ ortamlarında daha düşük enerji kullanımı ve daha az bant genişliği tüketimi avantajları sergilediği belirtilmektedir.

Nesnelerin İnternetinde hemen her gün daha fazla sayıda ve türde cihaz ağa bağlanmakta ve daha karmaşık ağ yapıları ortaya çıkmaktadır. Ağa bağlı olan her cihazın gereksinimlerinin farklılıklar göstermesi, karma yaklaşımların ve protokollerin birlikte çalışabilirliği konularında çalışmaların yapılmasına neden olmuştur. Özellikle servis keşfine yönelik yapılan ve MQTT ile CoAP protokolünün birlikte çalışabilirliğinin uygulandığı bir çalışmada[17], protokollerin noktadan noktaya sağlıklı iletişim sağlayamadıkları, gönderilen iletilerin dönüşümü nedeniyle gecikmeler yaşandığı ve sürecin karmaşık olduğu belirtilmektedir. Aynı ortamda birden fazla IoT uygulama protokolünün bulunması senaryolarını desteklemek amacıyla Bellavista ve diğerleri tarafından yapılan çalışmada [16] CoAP ile MQTT protokolünün birlikte çalışabilmesini sağlamak amacıyla bir yapı geliştirilmiştir. Çalışmada, trafik ve cihaz yoğunluğunun yüksek olduğu ağ ortamlarında bu mimarinin yüksek ölçeklenebilirlik sunduğu ifade edilmektedir. Çeşitli IoT uygulama protokollerinin aynı ortamda birlikte çalışabilirliği üzerine yapılan ve ara katman yazılımının geliştirildiği bir tez çalışmasında [33] yüksek oranda performans kayıpları yaşanmadan IoT

uygulama protokollerin birlikte çalışabilirliği ortaya konmaktadır. C. Lee ve diğerleri, CoAP ve MQTT protokollerinden gelen paketlerin birbirlerine dönüşümü için, yazılım tanımlı ağa (SDN - Software Defined Network) dayalı karma bir IoT iletişim çerçevesi önermektedir. Paketler aynı protokollere aitse, orijinal iletişim senaryoları çalışmakta ve trafik SDN tarafından göz ardı edilmektedir. Aksi durumda, SDN bu paketleri SDN denetleyicisine göndermekte ve denetleyici bu paketleri dönüştürmektedir [34]. P. Desai ve diğerleri, her protokol için ayrı bir arayüze sahip olan, çok protokollü bir aracı sunucu kullanarak XMPP, CoAP ve MQTT gibi mesajlaşma protokolleri arasında çeviri sağlayan bir ağ geçidi ve semantik web özellikli merkezi sunuculu bir IoT çözümü önermektedir [35]. A. Al-Fuqaha ve diğerleri, MQTT protokolünü geliştirerek, akıllı aracı sunucu (broker) kuyruk yönetimi ve trafik analizi tekniklerinin karışımından yararlanarak zengin hizmet kalitesi (QoS – Quality of Service) özelliklerini destekleyen, genel amaçlı bir IoT protokolü geliştirmişlerdir. İlgili çalışmada hibrit mimari, protokolün makineden sunucuya (M2S) iletişim için aracıyı kullanırken, doğrudan iletişim için noktaya yayın iletişimini kullanmaktadır [36].

Mevcut IoT protokolleri üzerine yapılan çalışmalarda, bu protokollerin birbirlerine göre farklı alanlarda üstünlükler sağlayabildiği görülmektedir. Nesnelerin İnternetinde belirli bir protokolün diğerine göre her alanda bir üstünlüğü bulunmamaktadır. Uygulamaların ihtiyaçları, ortamdaki ağın yapısı, güvenlik ihtiyacı, ölçeklenebilirlik, bant genişliği kullanımına göre farklı protokoller kullanılabilir [5]. Bu amaç doğrultusunda, aynı ortamda farklı çalışma mantıklarını sergileyen protokollerin birlikte kullanımına da ihtiyaç duyulmuştur. Ancak, birlikte çalışabilirlik performans kayıpları ve gecikme gibi beraberinde çeşitli sorunları da doğurmaktadır. Bu nedenle çok sayıda ve türde cihaz içeren IoT sistemlerinin karmaşık yapısı ve gittikçe artan oranda bir büyüme sergilemesi, yeni yaklaşımların ve yeni protokollerin geliştirilmesi ihtiyacını doğurmaktadır. Bu tez çalışması kapsamında, var olan yaklaşımların karması bir yaklaşım sergileyen hibrit bir uygulama protokolü geliştirilmiş ve kampüs ağlarında IoT uygulamalarında kullanılmak amacıyla önerilmiştir.

Bu tez çalışması dört bölümden oluşmaktadır.

Birinci bölümde, çalışmaya konu olan problem durumu ve tanımı, yapılan çalışmanın amacı, literatür özeti, tez çalışmasını özgün kılan yönler, Nesnelerin İnterneti alanına yapması hedeflenen katkılar ve tezin organizasyonu hakkında bilgiler yer almaktadır.

İkinci Bölümde, Nesnelerin İnterneti kavramı ele alınmakta, yaygın olarak kullanılan uygulama katmanı protokolleri incelenmekte ve başarım değerlendirmelerine değinilmektedir. Ayrıca Nesnelerin İnternetinde kabul görmüş mimariler ele alınmakta ve kampüs ağları için bir mimari önerilmektedir.

Üçüncü Bölümde, Nesnelerin İnterneti için genel amaçlı hibrit bir uygulama protokolü önerilmekte ve bu protokolün çalışma prensibi ayrıntıları ile ele alınmaktadır. Yine bu bölümde geliştirilen protokolün benzetim modellemesi sonuçları ve yaygın kullanılan uygulama protokolleri ile karşılaştırmaları incelenmektedir. Geliştirilen uygulama protokolünün fiziksel ortamdaki uygulaması ele alınmakta ve gerçek ortamda protokolün çalışmasına yönelik örnekler verilmektedir. Bu bölümde ayrıca, protokolün benzetim ortamındaki davranışları incelenmektedir.

Dördüncü bölümde, tez çalışması sonucunda elde edilen çıktılar genel itibariyle özetlenmekte, literatüre katkıları tartışılmaktadır. Ayrıca, bu çalışmanın devamı olarak yapılabilecek araştırmalara kaynak olabilecek öneriler sunulmaktadır.

2. NESNELERİN İNTERNETİ VE TEMEL KAVRAMLAR

İnternet'in kısa süredeki gelişimi, insanların günlük hayatını, çalışma şeklini, öğrenme stillerini ve hatta oyun oynama şekillerini belirgin bir şekilde değiştirmiştir. İnternet'in gelişim evreleri incelendiğinde, ilk kullanımının insanlar arasında bilgi paylaşımı yapmak olduğu söylenebilir [37]. Ancak sonraki evrelerde, İnternet teknolojilerin gelişmesi ve onlara erişimin daha kolay hale gelmesi sayesinde, birçok cihazın İnternet'e bağlanması sağlanmıştır. İnternet kullanıcıları genellikle, cep telefonu, bilgisayar ve taşınabilir cihazlar gibi birden fazla donanım ile İnternet'e bağlanmaktadır. İnsanlar tarafından kullanılan donanımların yanı sıra, çok çeşitli amaçlar doğrultusunda İnternet'e bağlanan, uzaktan yönetilmesi ya da izlenmesi gereken donanımlar da bulunmaktadır. 2020 yılında yeryüzünde yaşayan insan sayısı yaklaşık 7 milyar iken, aynı yıl itibari ile internete bağlı cihaz sayısının 30 milyar olması beklenmektedir [38]. Bağlı cihazların sayısının giderek artması, İnternet'in eskiden olduğu gibi insan merkezli olmaktan çıkıp, cihaz merkezli, daha doğru bir ifade ile nesne merkezli olmaya doğru kaydığını göstermektedir. Radyo Frekans Tanımlayıcı teknolojisi ile başlayan ve sensör teknolojileri ile devam eden, nesneleri internete bağlayabilme yeteneği, "Nesnelerin İnterneti" kavramının doğmasına neden olmuştur.

Bu bölümde Nesnelerin İnterneti kavramının ne olduğu, Nesnelerin İnternetinde kullanılan teknolojilere, protokollere ve mimarilere değinilmektedir. Yine bu bölümde, tez çalışmasına kaynak teşkil eden en yaygın kullanılan IoT uygulama katmanı protokollerinin çalışma prensipleri, paket yapıları ve iletişim modelleri detaylı olarak ele alınmıştır.

2.1 Nesnelerin İnterneti

Günümüzde her geçen gün daha fazla sayıda cihaz İnternet'e bağlanmaktadır. Geçtiğimiz yıllarda bağlı nesne sayısı dünya nüfusunu aşmış ve 2020 yılı itibariyle 26 - 50 milyar nesnenin internete bağlanacağı öngörülmektedir [5]. Yeni teknolojilerin kullanılmaya başlanmasıyla, fiziksel dünya da artık İnternet'e bağlanabilmektedir. Daha önce İnternete bağlı olmayan nesnelerin de çeşitli teknolojiler ile İnternet'e bağlanması sayesinde, İnternet evriminin dördüncü aşaması olarak adlandırılabilen Nesnelerin İnternetine geçiş yapıldığı söylenebilir [37,39]. İnternet'in kullanım senaryosu, insan merkezli olmaktan çıkmış, cihaz ve nesne merkezli olmaya doğru kaymıştır. Bu anlamda IoT, yeni bir İnternet değil, mevcut İnternet'in bir uzantısıdır.

IoT, birbiri ile bağlantılı olan nesnelerin eskiye oranla daha değerli ve anlamlı olmasını sağlamak amacıyla insanları, verileri, nesneleri ve veri işlemeyi bir araya getirmektedir. IoT'nin bu dört önemli unsuru arasındaki etkileşim sayesinde, makineler arası iletişim (M2M - Machine to Machine), makine – insan iletişimi (M2P - Machine to People) daha anlamlı ve verimli hale gelmektedir [38]. Bu sayede, ölçüm araçları, sensörler, güvenlik cihazları, sağlık cihazları ve daha birçok farklı türdeki cihazların ağ üzerinden birbirleri ile iletişime geçmesi sağlanmaktadır. Günlük hayatta karşılaşılan nesneler, her geçen gün daha çok ağ üzerinden adreslenebilir, tanımlanabilir, kontrol edilebilir cihazlar haline gelmekte ve birbirleri ile iletişime geçebilmektedirler.

Yarı iletken teknolojisinde yaşanan gelişmeler ve maliyetlerin ucuzlaması, iletişim bant genişliklerinin artması, erişim alternatiflerinin artması, hemen her yerden İnternete erişim olanağı, veri kaynaklarının çok çeşitlilik göstermesi ve sosyal medya gibi yeni veri kaynaklarının varlığı IoT kavramının dünyada yer edinmesine neden olmuştur. IoT, sensörlerden, makinelerden, eyleyicilerden (aktüatör) ve diğer fiziksel nesneler tarafından toplanan verileri elde etmek için bağlı cihazların ve sistemin kullanımını ifade eder. İnternet ve ağ kavramı söz konusu olduğunda yalnızca bilgisayarlardan, sunuculardan ve kullanıcılardan değil, sensörler ve RFID etiketler gibi teknolojilerin desteği ile fiziksel dünyadaki nesnelerden de gerçek zamanlı veriler elde edilebilir. Bu anlamda sensörler, aktüatörler ve etiketler (tag) fiziksel dünyanın dijital arayüzleridir. Kavramsal olarak Nesnelerin İnterneti yerel alan ağları, geniş alan ağları, kablolu veya çeşitli kablosuz teknolojiler aracılığıyla İnternet üzerinden adreslenebilen, tanımlanabilen, okunabilen ve kontrol edilebilen günlük nesnelerin oluşturduğu ağlar olarak ifade edilebilir.

Nesnelerin İnterneti kavramının ilk olarak Kevin Ashton tarafından 1999 yılında yaptığı bir sunumda kullanıldığı ifade edilse de [40], akademik olarak üzerinde fikir birliğine varılmış açık bir tanımı yoktur. Vermesan ve diğerlerine göre, basit olarak dijital dünya ile fiziksel dünyanın etkileşimi olarak ifade edilmektedir [4].

Elektrik Elektronik Mühendisleri Enstitüsü (IEEE- The Institute of Electrical and Electronics Engineers) tarafından Mart 2014'te yayınlanan Nesnelerin İnterneti hakkındaki özel raporunda "Nesnelerin İnterneti" ifadesi "*İnternete bağlı olan, her biri sensörlerle gömülü olan bir öğe ağı*" olarak tanımlamıştır [41].

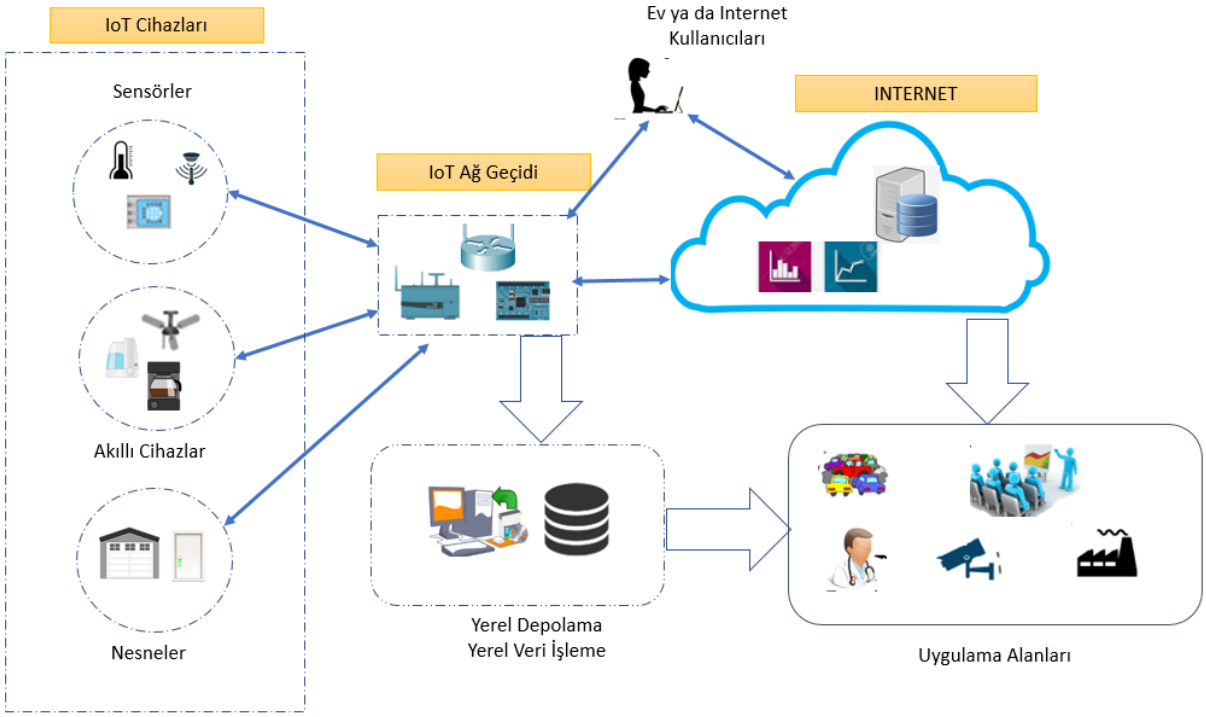
İnternet Mühendisliği Görev Gücü (IETF- The Internet Engineering Task Force) tanımlamasına göre Nesnelerin İnterneti, nesnelerin üretici, operatör ve/veya diğer bağlı cihazlarla veri alışverişi yapmalarını sağlamak için elektronik, yazılım, sensör ve bağlanabilirliğe sahip fiziksel nesnelerin ağıdır [42].

Uluslararası Telekomünikasyon Birliği (ITU) tanımlamasına göre ise IoT, bilgi toplumu için, mevcut ve gelişen birlikte çalışabilir bilgi ve iletişim teknolojilerini temel alan (fiziksel ve sanal) nesneleri birbirine bağlayarak gelişmiş hizmetler sunan küresel bir altyapıdır [43].

Bu tanımlamalara göre Nesnelerin İnterneti çok farklı türde, farklı çalışma prensiplerinde sensörler, yazılımlar, ağ bağlantıları ve veri iletişimi barındırdığından heterojen yapıda bir ekosistem olduğu söylenebilir. Bu çalışma kapsamında Nesnelerin İnterneti, "*elektronik nesneler, yazılımlar, sensörler, aktüatörler ve bu nesnelerin veri toplamasına, değiştirmesine olanak tanıyan fiziksel bağlantıların, araçların, yapıların ve diğer öğelerin ağı* [44]" tanımlaması esas alınacaktır.

2.2 Nesnelerin İnterneti Teknolojileri

Nesnelerin İnterneti, ağlar üzerinden bilgi aktarabilen işlemciler veya gömülü sensörler aracılığıyla bilgi işlem gücüne sahip nesnelerden oluşur. IoT gereksinimlerini karşılayan birçok farklı cihaz türü mevcuttur. Bununla birlikte, çoğu IoT cihazı işlevleri gerçekleştirmek için radyo frekans tanımlayıcıları, sensörler, kontrolörler ve aktüatörler (eyleyiciler) kullanmaktadırlar [45]. Birçok IoT çözümünde, bu donanımların birlikte kullanımının yanı sıra, bulut tabanlı destekler de bulunmaktadır. Şekil 2.1'de IoT çözümlerinin genel yapısının bir örneği bulunmaktadır.



Şekil 2.1. Nesnelerin İnterneti ekosisteminin genel yapısı

Çoğu IoT çözümlerinde, IoT cihazları, IoT Ağ Geçidi ve İnternet Bulut Hizmetleri bulunmaktadır [44]. Sensörler tarafından desteklenen ortamlar, giyilebilir teknolojiler, nesneler veya akıllı cihazlar Nesnelerin İnterneti'nde "*IoT Cihazları*" olarak tanımlanmaktadır. Nesneler, akıllı nesneler, internete bağlı nesneler, düğümler, cihazlar, IoT cihazları, akıllı cihazlar gibi terimler "nesneleri" ifade etmek için genellikle birbirinin yerine kullanılmaktadır. IoT cihazları, "*IoT Ağ Geçidi*" olarak tanımlanan donanımlar aracılığıyla ağ bağlantısına ya da internet bağlantısına sahip olmaktadırlar. IoT Ağ Geçidi, bu cihazlardan elde edilen verileri işlemek ya da değerlendirmek üzere yerel bilgi işlem cihazlarına ya da İnternet bulutuna gönderebilmektedir. Kullanıcılar, yerel ağdan ya da İnternet üzerinden bu IoT cihazları ile etkileşim kurabilmektedir. Yerel kaynaklardan ya da İnternet bulutundan elde edilen anlamlandırılmış bilgiler, IoT çözümlerinin kaynağını oluşturmaktadır. Şekil 2.1'de gösterilen ve birbirleri ile etkileşim halinde bulunan donanımlar, teknolojiler, veriler ve kullanıcılar "*IoT Ekosistemi*" olarak adlandırılan bir yapıyı oluşturmaktadır.

Bu bölümde, IoT ekosisteminde yaygın olarak kullanılan donanımlar ve teknolojiler ele alınmaktadır.

2.2.1 IoT sisteminde sensörler

Günümüzde sensörler taşınabilir donanımlar, akıllı cep telefonları, ev içinde kullanılan elektronik donanımlar ve giyilebilir teknolojiler örnek olmak üzere hayatımızın hemen her noktasında kullanılmaktadır [46]. Üretim sektöründe, fabrikalarda, montaj hatlarında çeşitli parametreleri algılamak için de yine sensörler kullanılmaktadır. Sensör, tanımlanmış fiziksel kimyasal veya biyolojik miktarları algılayabilen veya ölçebilen ve buna ilişkin nicel veriler üreten belirli bir cihaz kategorisini ifade etmektedir. Diğer bir ifade ile fiziksel dünyadan bir tür bilgiyi algılayıp çevresel bir özelliği ölçmek için kullanılabilen, fiziksel dünyanın dijital arayüzü olarak ifade edilebilen cihazlardır. Bu özellikler ışık, nem, hareket, basınç, sıcaklık veya başka herhangi bir çevresel durum olabilir. IoT sisteminde sensörler, fiziksel dünya olaylarının teknoloji yardımıyla sinyallerle taşınan ve işlenebilmesi için uygun bir dijital formata dönüştürülen veri kaynaklarıdır [47].

Temel olarak sensörler, aktif ve pasif olmak üzere iki kategoride incelenebilir. Pasif sensörler, çalışabilmesi için ek bir güce ihtiyaç duymayan türler iken, aktif sensörler çalışabilmesi için güce ihtiyaç duymaktadırlar. Yine sensörlerin sınıflandırma yöntemlerinden biri de algılama ve ölçüm için kullanılan yönteme göre belirlenmesidir. Bu yönteme göre sensörler mekanik, kimyasal, termal, gaz, kızılötesi sensörler gibi sınıflandırılabilir. Ayrıca, sensör tarafından üretilen sinyal türüne göre bir sınıflandırma yapılacak olursa, analog ve dijital sensörler olarak da kategorize etmek mümkündür [38].

Genel olarak sensörler, çoğunlukla hesaplama kapasitesi sınırlı olan ya da bilgi işlem hesaplama kaynakları yönünden yetersiz olan cihazlardır ve verinin kaynağı olarak görev yaparlar. Bu nedenle IoT ekosisteminde sensörler, kendilerinden elde edilen verilerin bilgisayar ağlarında kullanılabilir bir forma dönüştürülebilmesi için denetleyiciler ile birlikte kullanılmaktadırlar.

2.2.2 Aktüatörler (Eyleyiciler)

Temel olarak aktüatör, enerjiyi harekete geçirmeyi sağlayan bir mekanizmadır. IoT ekosisteminde geniş kullanım alanı bulan eyleyiciler, genel olarak bir elektrik sinyalini alıp, bu sinyal doğrultusunda hareket başlatan ya da fiziksel uyarı sunan bileşenler ya da donanımlar olarak ifade edilmektedir. Bu fiziksel hareket ya da çıkış, ledler aracılığıyla bir kullanıcıya bilgi

sağlayabildiği gibi, başka bir cihazı çalıştırmak üzere tetikleyebilen herhangi bir mekanizma da olabilir [48].

Genel olarak endüstride, elektrikli, hidrolik, termal ve pnömatik olmak üzere dört tip aktüatör kullanılmaktadır. Elektrikli türdeki aktüatörler, elektrik enerjisini mekanik işlemlere dönüştüren bir motorla çalışmaktadırlar. Hidrolik olanlarda ise mekanik hareketi gerçekleştirmek için sıvı basıncı kullanılmaktadır. Pnömatik türlerde ise hareketi sağlamak için basınçlı hava kullanırken, termal türlerde sıcaklıktan faydalanılır. Aktüatörün hareketinin gerçekleştirilmesine neyin sebep olduğuna bakılmaksızın, bir aktüatörün temel işlevi denetleyiciden sinyal almak ve bu sinyale dayanarak belirlenmiş bir eylemi gerçekleştirmektir [38].

2.2.3 Denetleyiciler

IoT ekosistemindeki sensör gibi donanımların esas görevi ortama ilişkin veri toplamaktır. Toplanan bu verilerin işlenmesi, depolanması, daha güçlü konumlarda bulunan bilgi işlem cihazlarına gönderilmesi ve bu verilere ağ ya da internet üzerinden erişilebilmesi, IoT'nin sunduğu avantajlardandır. Bu amaç doğrultusunda, kullanıcıların programlamasına olanak tanıyan küçük, ucuz ve esnek bilgisayar donanımları, diğer bir ifade ile mikro denetleyiciler (MCU) kullanılmaktadır. Basit olarak ifade etmek gerekirse, bir mikrodenetleyici işlemci, bellek, saat sayacı, giriş/çıkış birimlerini tek bir bütünleşmiş kartta sağlayan donanımlardır [49]. IoT cihazları kullanım ve özelliklere göre büyük farklılıklar gösterir, ancak temel olarak kullanılan donanımlar mikrodenetleyiciler tarafından desteklenmekte olan donanımlardır [50].

RFID ya da sensörler aracılığıyla pasif cihazlardan elde edilen verilerin ağ ortamında kullanılabilmesi için de denetleyiciler kullanılmaktadır. Denetleyiciler çeşitli sensörlerden aldıkları bilgileri IP tabanlı olarak internet ortamına sunmaktan sorumlu olan cihazlardır. Denetleyiciler, karar verme mekanizmasında sahip güçlü cihazlar olabildiği gibi, yerel ağdaki ya da internetteki güçlü bir bilgisayara verinin ulaştırılmasından sorumlu olan ara cihazlar da olabilir [51].

IoT uygulamalarında prototip kullanımında genel olarak, kullanımı kolay, genişletilebilir donanım desteği sunan Arduino, Raspberry-PI ve NodeMCU gibi mikrodenetleyiciler kullanılır[52–54].



Şekil 2.2. Mikrodenetleyici olarak kullanılan Arduino

Şekil 2.2'de gösterilen Arduino, basit bir mikrodenetleyici kartına dayanan açık kaynaklı fiziksel bir bilgi işlem platformudur. Kolay kullanımı, yazılım geliştirme ortamının bulunması, birlikte çalışabilecek donanım sayısının fazlalığı ve kablolu-kablosuz iletişim modüllerinin bağlanabilmesi, IoT uygulamalarında yaygın kullanılmasına neden olmaktadır.

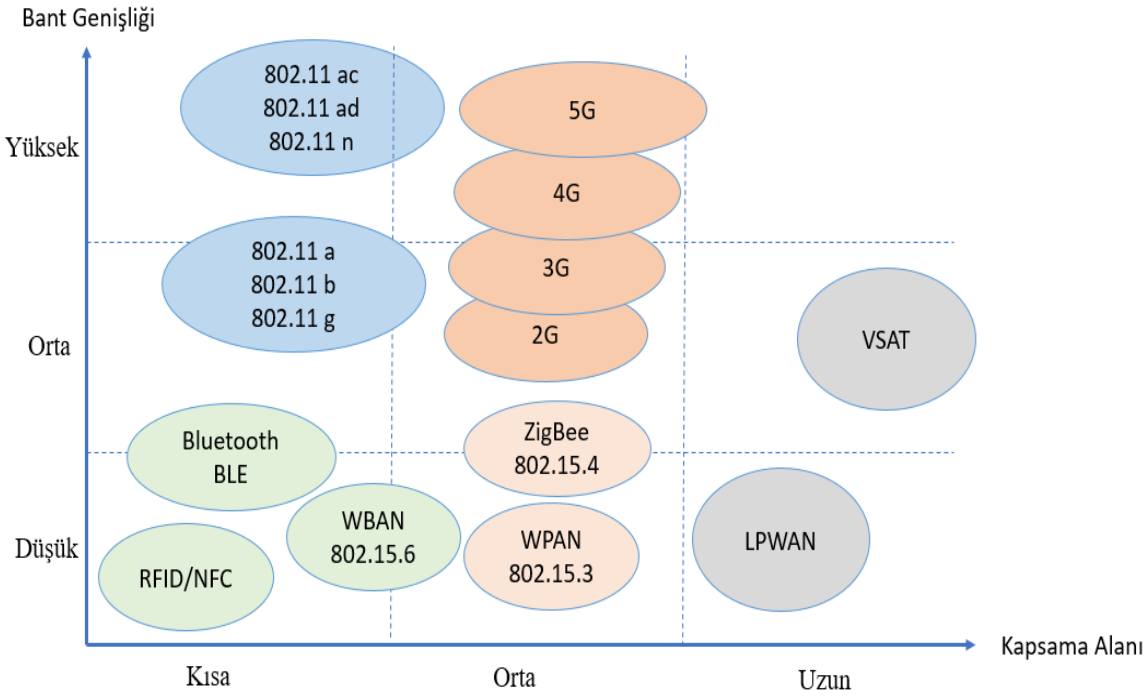


Şekil 2.3. Tek kartlı bilgisayar, Raspberry Pi

Şekil 2.3'de gösterilen Raspberry-PI, düşük maliyetli, küçük boyutlu, üzerinde işletim sistemi çalıştırabilen, fare/klavye gibi ek donanımların takılabildiği bir bilgisayardır. Bu donanım üzerinde Python programlama yapabilme yeteneği sayesinde, Nesnelerin İnternetinde uygulama geliştiriciler için yaygın kullanılan bir donanım olmaktadır.

2.3 Nesnelerin İnternetinde Kablosuz Teknolojiler

Son yıllarda hemen her alanda olduğu gibi, iletişim teknolojileri alanında da gelişmeler yaşanmakta ve bu doğrultuda Nesnelerin İnternetinin çok daha fazla yaygınlaşmasını sağlayan yeni kablosuz teknolojiler de ortaya çıkmaktadır. Önceki bölümde de belirtildiği gibi, Nesnelerin İnterneti yeni bir İnternet değil, mevcut internetin bir uzantısıdır. Bu nedenle, hem daha önceden kullanılan WiFi gibi birçok kablosuz iletişim teknolojisi, hem de LoraWan, ZigBee gibi yeni teknolojiler de IoT ekosisteminde geniş ve önemli bir yer kaplamaktadır.



Şekil 2.4. Bant genişliği kullanımı ve kapsama alanına göre kablosuz teknolojiler

IoT ekosisteminde kullanılan cihazlar, geleneksel bilgi işlem donanımlarının konumlandırıldığı yerler haricinde çok farklı konumlarda kullanılmaktadır. Hemen her gün çok farklı türde ve sayıda cihaz IoT'nin sunduğu uygulamalarda kullanılmaktadır. Bu yeni cihazların çeşitliliğini ve farklı bağlantı ihtiyacını karşılamak için, birçok yeni kablosuz teknoloji ve protokol geliştirilmiştir. Kablosuz protokollerin bazıları kısa mesafeli ağlar için, bazıları orta, bazıları ise daha uzun mesafeli ağlar için tasarlanmıştır. Şekil 2.4, farklı protokolleri destekledikleri mesafeler ve bant genişliği kullanımına göre kategorize eden bir grafiği göstermektedir.

2.3.1 Radyo frekans tanımlayıcı (RFID- Radio frequency identification)

IoT'nin temel sağlayıcılarından biri, son 20 yılda geliştirilen düşük güçlü kablosuz teknolojilerdir. Bu teknolojiler, fiziksel bağlantı ya da kurulum olmadan iletişim kurulmasını sağlamaktadır. Günlük hayatta kullanılan nesnelerin ağlara bağlanma tekniklerinden biri, radyo frekansı tanımlama (RFID) teknolojisidir. RFID, nesnelere entegre edilen bir etiket sayesinde nesneye ait bilgiler taşıyan ve bu sayede nesnenin hareketlerinin izlenebilmesini, analiz edilebilmesini ve yönetilebilmesini radyo frekansları ile sağlayan, otomatik nesne tanımlama ve takip teknolojisidir [55]. Bu özellikleri sayesinde, IoT ekosisteminde geniş kullanım alanları bulmuştur. Günlük hayatta kullanılan nesnelerde RFID etiketler kullanıldığında, bilgisayarlar tarafından tanımlanabileceğinden, IoT uygulamalarının ilk evrelerinde bir ön koşul olarak görülmektedir. En başarılı uygulamalar arasında tedarik zinciri yönetimi, üretim süreci kontrolü ve nesne izleme yönetimi bulunmaktadır [56].

2.3.2 Kablosuz yerel alan ağları (Wi-Fi)

Wi-Fi (Wireless Fidelity), IEEE 802.11 standart ailesine dayanan, cihazların yerel alan ağı (LAN) ve İnternet erişimi için radyo dalgalarını kullanan yüksek hızlı bir kablosuz ağ teknolojileri ailesidir. WiFi, küresel ölçekte her yerde bulunan kablosuz ağlardan biridir. Geniş kapsama alanı yeteneği, düşük güç tüketimi ve yongalarının etkileyici bir şekilde küçük ve düşük maliyetli olması nedeniyle, gelecekte IoT uygulamalarında anahtar rol alacağı düşünülmektedir [57].

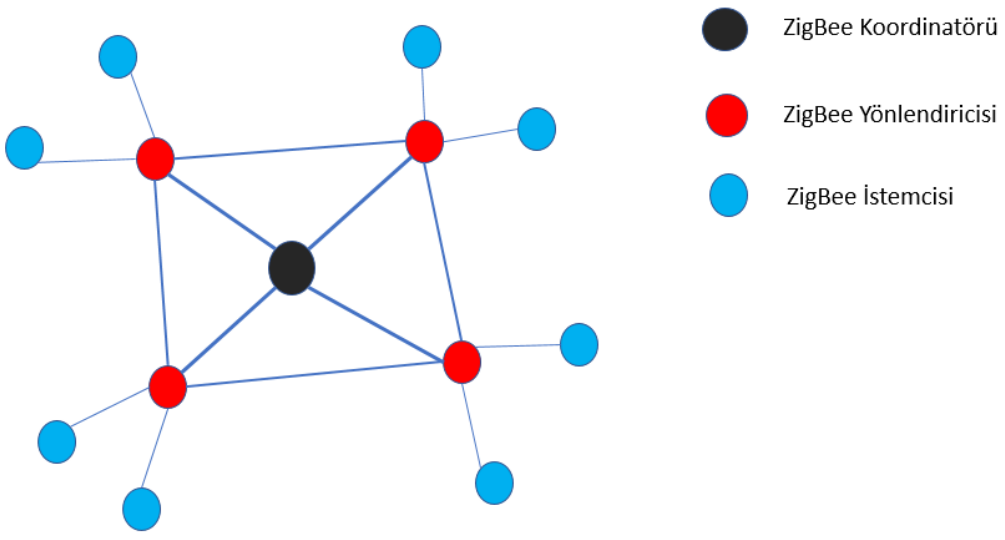
WiFi kablosuz bağlantısı, diğer kablosuz bağlantıların çoğuna oranla daha yüksek hız ve daha geniş kapsama alanı sunmaktadır [58]. WiFi kablosuz standartlar ailesi, birçok farklı bant genişliği sunan alt teknolojilere sahiptir. Çizelge 2.1’de 802.11 ailesinin alt standartları gösterilmiştir.

Çizelge 2.1. WiFi standartları

Standart	Frekans	Bant Genişliği
802.11a	5 GHz	54 Mb/s
802.11b	2.4 GHz	11 Mb/s
802.11g	2.4GHz	54 Mb/s
802.11n	2.4/5GHz	300Mb/s
802.11ac	5 GHz	1,35 Gb/s
802.11ad	2,4/5 GHz	7 Gb/s

2.3.3 ZigBee

ZigBee, kablosuz kontrol ve izleme uygulamalarında pille çalışan cihazlar için geliştirilen düşük maliyetli, düşük güç tüketen, kablosuz bir ağ standardıdır [59]. Özellikle ev otomasyonları gibi kişisel alan ağları (PAN- Personal Area Network) uygulamalarında kullanılan bu standart, Bluetooth veya Wi-Fi gibi diğer kablosuz kişisel alan ağlarından daha basit ve daha ucuz olacak şekilde ağ kurmak için tasarlanmıştır. 250 Kb/s aktarım hızı sunan ZigBee, çok ek yük gerektirmeyen temel M2M IoT uygulamalarında kullanılabilmektedir [60, 61].



Şekil 2.5. Zigbee topolojileri ve zigbee rolleri

Şekil 2.5'de gösterildiği gibi bu standartta, ZigBee Koordinatörü, ZigBee Yönlendiricisi ve ZigBee İstemcisi rolleri bulunmaktadır [60]. ZigBee istemcilerinin yönetimi ve ağın oluşturulması, koordinatör olan donanımın sorumluluğundadır. Yönlendiriciler ise, istemciler ile koordinatör arasında bir köprü görevi görerek, ZigBee ağının genişletilmesini sağlamaktadır. Koordinatör, sekiz adede kadar istemci ya da yönlendirici ile iletişim kurabilir. ZigBee ağlarında yıldız topolojisi ya da örgü (mesh) topolojisi kurulabilir. Bu nedenle kablosuz algılayıcı ağ uygulamalarında da geniş kullanım alanı bulmuştur [62].

2.3.4 Diğer kablosuz teknolojiler

Nesnelerin İnternetinde, yukarıda adı geçen teknolojilerin yanı sıra, Düşük Enerjili Bluetooth ve Düşük Güçlü Alan Ağları gibi çok farklı türde kablosuz teknoloji kullanılmakta ve her geçen gün yeni teknolojiler eklenmektedir.

Düşük Enerjili Bluetooth: Bluetooth, kişisel alan ağları gibi kısa mesafelerde veri iletişimi için kullanılan, neredeyse tüm güncel mobil cihazlarda desteklenen kablosuz bir protokoldür. Bluetooth endüstriyel, bilimsel ve tıbbi (ISM) radyo frekans bandında çalışır. Zaman içerisinde birçok Bluetooth geliştirmesi yapılmıştır. Nesnelerin İnterneti uygulamalarında ise daha çok kullanılan sürümü, Düşük Enerjili Bluetooth (BLE- Bluetooth Low Energy) geliştirmesidir. Milisaniyeler düzeyinde çok yüksek bir bağlantı hızına ve ZigBee teknolojisine oranla 1 Mb/s gibi yüksek bir veri taşıma kapasitesine sahiptir. Bu bluetooth sürümünün IoT ekosisteminde kullanılmasının en önemli sebeplerinden biri, bağlantı kurulduktan sonra, bağlantı tekrar kuruluncaya kadar "uyku moduna" geçebilme yeteneğidir. Bu önemli avantaj, pil ile beslenen IoT sensör düğümlerinin pil ömrünün uzatılmasında önemli rol oynamaktadır. Düşük güç gerektiren, ev aletleri ve giyilebilir cihazlar gibi donanımların bağlantılarının kurulması, izlenmesi ve bu donanımlardan bilgi paylaşımının yapılması uygulamalarında kullanılmaktadır [62].

Düşük Güçlü Geniş Alan Ağları (LPWAN): Sensörler, aktüatörler ve denetleyiciler gibi düşük hızlı cihazlarda, uzun mesafelerdeki iletişimleri desteklemek üzere tasarlanmış bir kablosuz iletişim ağıdır. Sinyal alanı, görüş alanına bağlı olarak 5 ile 40 km arasında değişkenlik göstermektedir. Baz istasyonları 10 yıl pil ömrüne sahiptir ve binlerce cihazı destekleyebilmektedir. LPWAN protokollerine örnek olarak, LoRaWAN ve NarrowBand IoT verilebilir [48].

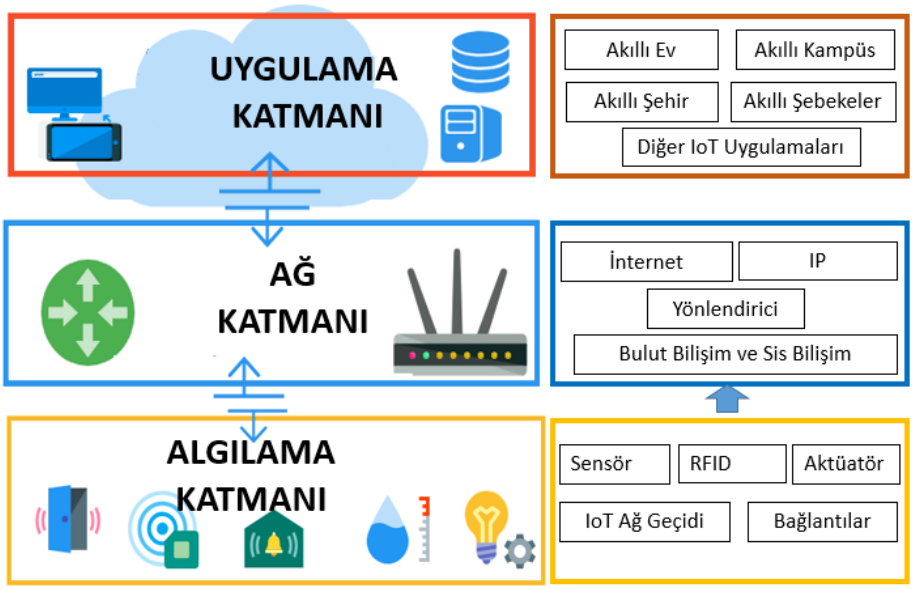
2.4 IoT Mimarileri

Mimari, ana kavramsal öğelerin, bir hedef sistemin gerçek öğelerinin, birbirleriyle nasıl ilişkilendiklerinin ve tasarım ilkelerinin tanımlanması anlamına gelmektedir. IoT sisteminin heterojen yapısı nedeniyle çok farklı sayıda ve türde uygulamalar, yaklaşımlar, teknolojiler ve protokoller geliştirilmiştir. Bu farklı IoT sistemleri arasında etkin iletişimi sağlamak, tutarlı, güvenli ve yaygın olarak tanınan teknolojilerin ve standartların kullanılmasını gerektirmektedir. Bu amaç doğrultusunda çeşitli kuruluşlar tarafından mimariler geliştirilmiştir. Bu mimariler 2.4.3 başlığı altında, referans modeller olarak anlatılmaktadır.

IoT için mimari konusunda yapılan akademik çalışmalarda, üzerinde fikir birliğine varılmış tek bir mimari bulunmamaktadır. Farklı araştırmacılar tarafından farklı mimariler önerilmiştir ancak genellikle üç veya beş katmanlı mimariler esas alınmıştır.

2.4.1 Üç katmanlı mimari

Üç katmanlı IoT mimarisinde, Algılama Katmanı, Ağ Katmanı ve Uygulama Katmanı bulunmaktadır.



Şekil 2.6. Üç katmanlı IoT mimarisi

Şekil 2.6'da gösterildiği gibi birinci katmanda, sensörler ve aktüatörler gibi fiziksel donanımlar bulunmaktadır. İkinci katman ise, sensör veri toplama sistemlerinden, ağ ve internet bağlantısı sağlayan sistemlerden oluşur. 3.Katmanda, veriler bir uygulamaya gönderilir ya da bu veriler bir uygulamayı geliştirmek için kullanılır [48].

Sensörler, bir depoda bulunan su seviyesinin tespiti, ortamın sıcaklığı gibi ölçtükleri ortamdan ya da bağlı bulunduğu nesneden gelen verileri toplayıp yararlı verilere dönüştürür. Çoğu uygulamada elde edilen veriler zamana duyarlı verilerdir ve hemen işlenmesi, bilgiye dönüştürülmesi gerekmektedir. Bu katmanda kullanılan denetleyici gibi donanımların desteği ile daha üst katmanlara çıkmadan, elde edilen verinin bilgiye dönüşmesi işlemi yapılabilir. Elde edilen veri, ağ bağlantısı gerektirmeyecek şekilde bir aktüatörü tetiklemesi gereken bir durum söz konusu ise, M2M iletişimi yine bu katmanda gerçekleşecektir. Ancak bulut ortamına gönderilmesi gerektiği durumlarda veya daha yüksek hesaplama kapasitesine sahip donanımlarla veri analizi yapılması gerektiği durumlarda ya da verinin ağda bulunan bir konumda depolanması ihtiyacı durumunda, ağ bağlantısının sağlanması gerekmektedir.

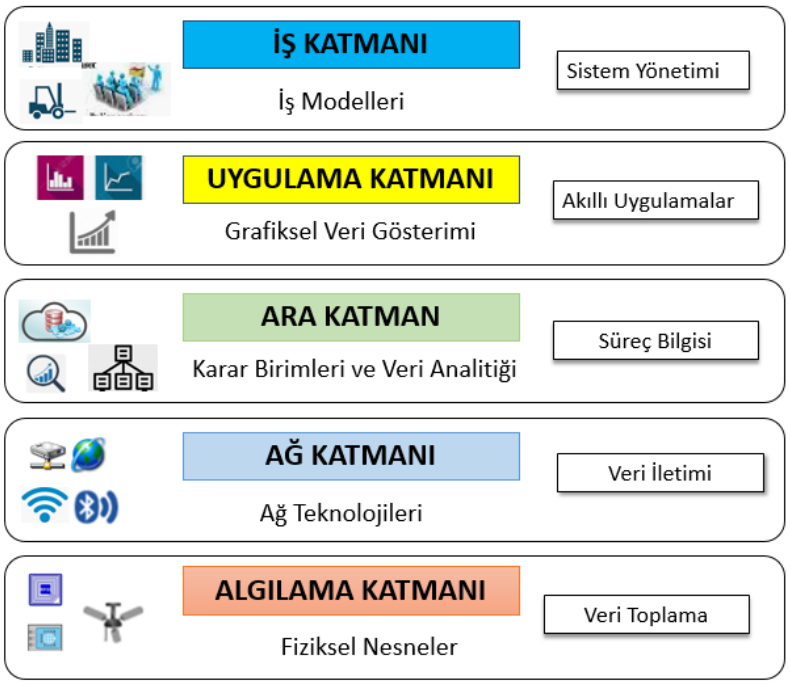
Sensörlerden ya da aktüatörlerden elde edilen veriler bazı durumlarda ham, işlenmesi gereken veriler olabilir. Bu verilerin, IP paketleri içerisinde kapsüllenmesi ve işlenmek üzere ağ üzerinden transfer edilmesi gerekmektedir. IoT ekosisteminde kullanılan denetleyiciler bu amaç doğrultusunda kullanılan ve IoT verileri için Ağ Geçidi olarak çalışan donanımlardır. Ağa aktarılan veriler daha sonra, kablolu ya da kablosuz teknolojiler yardımıyla yönlendiriciler kullanılarak İnternet ortamına aktarılabilir.

Ağ ortamına aktarılan veriler, geliştiriciler tarafından yazılan uygulamaları besleyen veriler olmaktadır. Bu veriler toplandıktan, temizlenip derlendikten ve incelendikten sonra analiz edilmekte ve yeni ürün veya hizmetlere kaynak teşkil etmektedir. Akıllı şehir, akıllı kampüs ya da akıllı ev uygulamaları bu katmanda çalışan IoT hizmetleridir.

2.4.2 Beş katmanlı mimari

Bu IoT mimarisinde, işlem adımları daha ayrıntılı ele alınmaktadır. Beş katmanlı mimari konusunda çeşitli akademik öneriler geliştirilmiştir [63–65]. Bu başlık altında, Wu ve diğerleri tarafından önerilen model anlatılmaktadır. Şekil 2.7'de gösterilen bu model, Algılama Katmanı,

Ağ Katmanı, Ara Katman, Uygulama Katmanı ve İş Katmanı olarak adlandırılan katmanlardan oluşmaktadır [66].



Şekil 2.7. Beş katmanlı IoT mimarisi

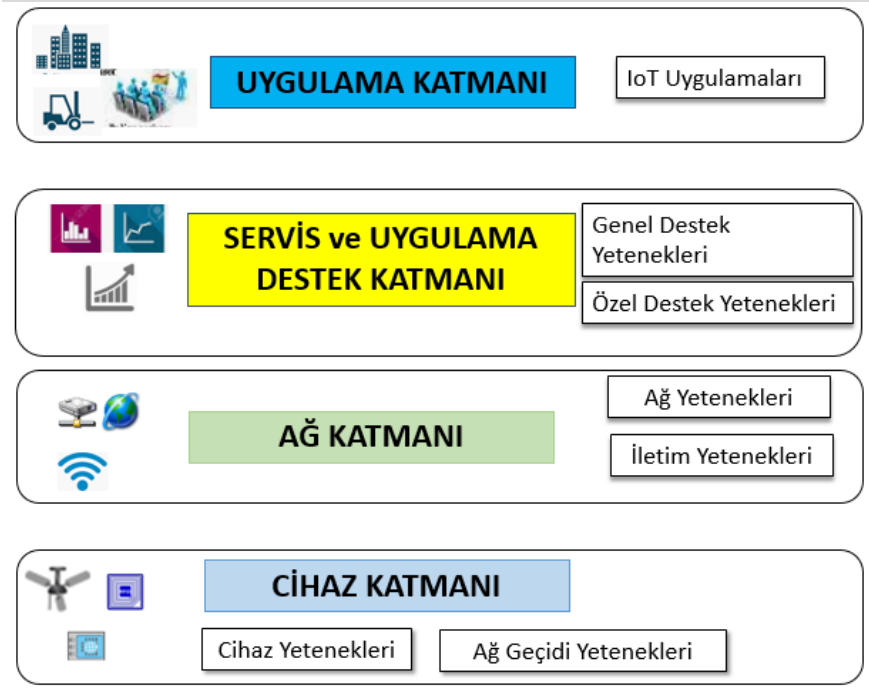
Bu mimaride algılama katmanı, sensörler ve fiziksel nesneleri içeren "fiziksel katman" olarak görülmektedir. 3 katmanlı mimaride olduğu gibi, verilerin sensörlerden elde edilmesi süreci algılama katmanında gerçekleşmektedir. 3 katmanlı mimariden farklı olarak, IoT Ağ Geçidi donanımları bu katmanda bulunmamaktadır. Ağ katmanı, fiziksel ortamdan alınan verilerin, kablolu – kablosuz teknolojiler ya da bluetooth teknolojileri ile aktarılması süreçlerini kapsamaktadır. Verilerin Internet Protokolü paketleri aracılığı ile taşınması süreci yine bu aşamada gerçekleşmektedir.

Ağ ortamında aktarılan veriler işlenmek ve analiz edilmek amacıyla, karar destek mekanizmalarının bulunduğu ara katmana geçmektedir. Ara katmanda, veriler üzerinde veri analizi işlemleri yapılmakta ve karar verici uygulamalara kaynak olacak anlamlı bilgiler sunulmaktadır. Uygulama katmanında, elde edilen verilerin uygulamalar aracılığı ile grafiksel olarak gösterimi gibi, daha çok uygulamaların kullanıldığı grafik kullanıcı arayüzü işlemleri yapılmaktadır. Bu katmanda artık bir uygulama geliştirilmiş ve ihtiyaç duyulan sektörde

kullanılmak üzere kullanılabilir hâle gelmiştir. İş Katmanında ise, IoT'nin sunduğu avantajdan yararlanılması süreci temsil edilmektedir.

2.4.3 Referans modelleri

IoT uygulamalarında çeşitli sistemler, uygulamalar, dağıtık ve heterojen yapılar, teknolojiler ve protokoller yer almaktadır. Sektörde bulunan kurumlar kendi çözümlerini geliştirmektedirler. Ancak bu çözümlerin de bir ağ ortamında birlikte çalışacağı öngörülmekte ve bu nedenle standart belirleyen kurumlarca referans modeller geliştirmeye yönelik girişimler başlatılmıştır. Bu girişimler birlikte çalışabilirliği kolaylaştırmayı, geliştirmeyi basitleştirmeyi ve uygulamayı kolaylaştırmayı amaçlamaktadır [3].



Şekil 2.8. ITU - IoT referans modeli

Uluslararası Telekomünikasyon Birliği tarafından hazırlanan IoT Referans Modelinde, Şekil 2.8'de gösterildiği gibi, temel olarak dört katman bulunmaktadır [67]. Uygulama katmanı, Nesnelerin İnterneti uygulamalarını içermektedir. Servis ve Uygulama Destek Katmanı, genel destek yetenekleri ve özel destek yetenekleri olmak üzere iki grupta incelenmektedir. Genel destek yetenekleri, veri işleme veya veri depolama gibi farklı IoT uygulamaları tarafından kullanılabilen ortak özelliklerdir. Özel destek yetenekleri, çeşitli uygulamaların

gereksinimlerini karşılayan özel yeteneklerdir. Aslında bu katman, farklı IoT uygulamalarına farklı destek fonksiyonları sunabilmek amacıyla çeşitli ayrıntılı yetenek gruplarından da oluşabilir.

Ağ katmanında yer alan ağ yetenekleri, erişim ve iletim kaynağı kontrol işlevleri, hareketlilik yönetimi veya kimlik doğrulaması, yetkilendirme ve hesap yönetimi gibi ağ bağlantısının ilgili kontrol işlevlerini sağlar. İletim yetenekleri, IoT hizmetinin ve uygulamaya özgü olan veri bilgilerinin taşınması yeteneğini ifade etmektedir. Ayrıca IoT ile ilgili kontrol ve yönetim verilerinin taşınması için, bağlantı sağlama yeteneğini de içermektedir.

Cihaz katmanında, cihaz yetenekleri ve ağ geçidi yetenekleri yer almaktadır. Cihaz yetenekleri, cihazın doğrudan veri toplayabilmesi ve ağ geçidinden bağımsız olarak bilgiyi doğrudan ağa aktarabilme yeteneğini ifade etmektedir. Bu noktada diğer modellerden farklılık göstermektedir. Ancak doğrudan olmayan yöntemlerle, yani ağ geçidi aracılığıyla veri toplayıp gönderebilme yeteneğinin olması da yine bu cihaz yetenekleri arasında yer almaktadır.

Nesnelerin İnterneti için tanımlanan diğer bir referans modeli de, 28 farklı üye tarafından oluşturulan bir toplulukça benimsenen, IoT Dünya Forumu Referans Modelidir. Bu modelde, IoT sistemi için yedi düzeyden oluşan bir yapı önerilmektedir. Bu modelin katmanları, Çizelge 2.2'de gösterilmiştir.

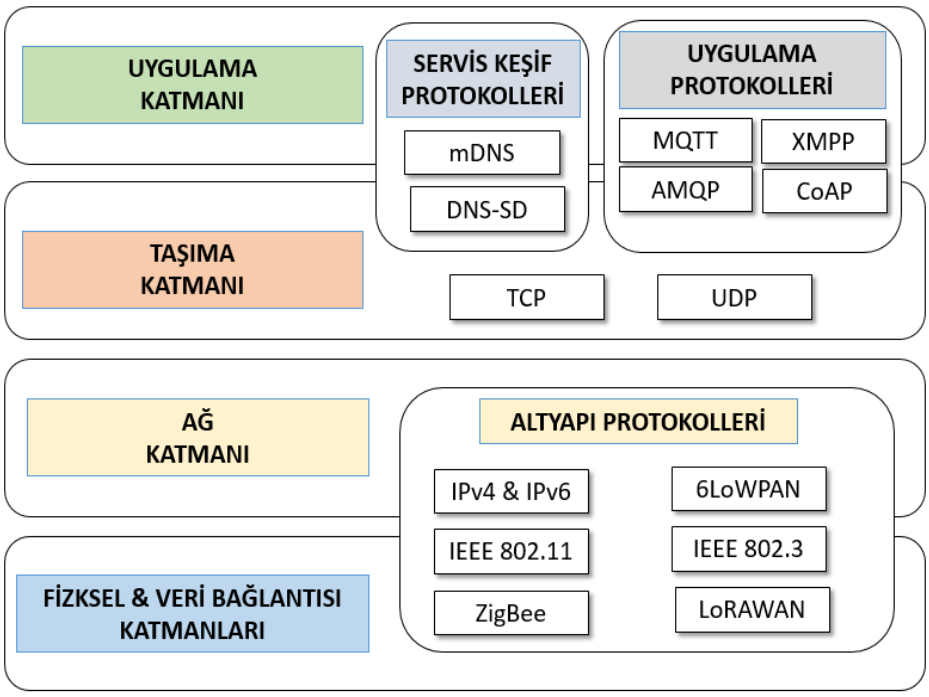
Çizelge 2.2. IoT Dünya forumu referans modeli

Düzey	**Düzey Adı**	**Açıklama**
7	İşbirliği ve Süreçler	İnsanlar ve iş süreci arasında gereken iletişimi ve işbirliğini içerecek şekilde birden fazla uygulamayı karşılama.
6	Uygulama	Cihaz verilerinin niteliğine ve iş gereksinimlerine dayalı bilgi yorumlama.
5	Veri Soyutlama	Verileri ve veri depolamasını uygulama geliştirmeyi sağlayacak şekilde oluşturma.
4	Veri Toplama	Hareket halindeki verinin, bekleyen durumdaki verilere dönüştürülmesidir. Ayrıca verinin üst katmanlarca kullanılabilecek şekilde sunulması.
3	Sis Bilişim	Verileri, depolama ve daha üst düzey işlemler için uygun bilgilere dönüştürme.
2	Bağlanabilirlik	3.Düzeydeki cihazlar ve ağ arasında, ağlar arasında ve ağ ve veri işleme arasında güvenilir ve zamanında veri iletiminden sorumludur.
1	Fiziksel Cihazlar ve Denetleyiciler	Bilgi gönderip alan çok çeşitli uç nokta cihazları içerir.

IoT referans modelinin amacı, ortak terminoloji sağlamak ve birleşik bir IoT endüstrisi için bilginin nasıl aktığını ve işlendiğini netleştirmeye yardımcı olmaktır.

2.5 IoT Protokolleri

Herhangi bir cihazın internete bağlanabileceği ve kendini diğer nesnelere tanımlayabildiği bir iletişim sistemi olarak tanımlanabilen Nesnelerin İnterneti, akıllı evler veya giyilebilir cihazlar gibi kişisel kullanımdan, kentsel çevre izlemeye, sağlık hizmetlerine, endüstriyel otomasyona ve acil durumlara kadar çok farklı alanlarda uygulanabilmektedir [8]. Bu uygulamalarda kullanılan donanımlar, genellikle düşük bellek ve düşük işlem gücü gibi kısıtlı ve sınırlı kaynaklara sahip cihazlardır. Standart TCP/IP kümesinin protokolleri bu donanımlar için uygun olmadığından[68], çeşitli protokoller geliştirilmiş ve geliştirilmeye devam edilmektedir [21].



Şekil 2.9. IoT protokollerinin OSI modelindeki gösterimi

Nesnelerin İnternetine konu olan protokoller daha önce sözü edilen mimariler açısından incelendiğinde, altyapı protokolleri, servis keşif protokolleri ve uygulama protokolleri olmak üzere üç grup altında toplamak mümkün olmaktadır. Bu protokollerin OSI katmanlarındaki yerleşimi Şekil 2.9'da gösterilmiştir.

IoT uygulamaları mevcut IP ağı üzerinde çalışabilmelidir. Bu doğrultuda IoT uygulamaları, standart IP ağlarında kullanılan Ethernet gibi teknolojileri ya da altyapı protokollerini kullanabilmektedir. Kablolu Ethernet (IEEE 802.3) ya da WiFi (IEEE 802.11 ailesi), hâlen yerel alan ağlarında çok sık kullanılan birer standarttır. Bu nedenle Nesnelerin İnternetinin LAN uygulamalarında, dolayısıyla IPv4 ya da IPv6 ağlarında kullanılabilmektedir. Arduino ve Raspberry-PI gibi IoT denetleyicilerinin Ethernet desteği sunması sayesinde, birçok IoT uygulama prototiplerinde ya da küçük çaplı uygulamalarda ve akademik çalışmalarda kullanılmaktadır.

IPv4 yetersizliklerini çözmek amacıyla geliştirilen IPv6 iletim protokolü de Nesnelerin İnternetinde kullanım alanı bulmaktadır. 6LoWPAN, IEEE 802.15.4 standardı ile tanımlanan düşük güçlü kablosuz iletişimi kullanarak, kısıtlı kaynaklara sahip cihazlar için IPv6 bağlantısını sağlamak üzere geliştirilmiştir [69].

IoT'nin sunduğu vizyonun gerçekleştirilmesinde kullanılan diğer bir protokol grubu da, servis keşif mekanizmalarını sunan protokollerdir. IoT ekosistemlerinde çok sayıda hizmet sunma potansiyeli, servis keşfini herhangi bir IoT platformunun temel gereksinimi haline getirmiştir [70]. Servis keşfi ya da bu servisi yapan cihazın keşfedilmesi, Nesnelerin İnternetinde üzerinde hâlen akademik çalışmalar yapılan ve geliştirilen bir alandır. Bu keşif süreci, bir görevi yerine getirmek için yararlı cihazlar bulma problemidir ve IoT için büyük bir zorluk oluşturmaktadır. Servis keşfi, tüketicilerin insan müdahalesi olmadan diğer hizmetlerin varlığını bilmesini sağlamak için tasarlanmıştır [71,72].

Servis Keşif sürecindeki temel zorluk, IoT ihtiyaçlarını karşılamak için benimsenen mimariler ve yeni teknolojiler açısından IoT standardizasyonunun olmamasıdır. Bu nedenle, hem IoT ortamının hem de IoT hizmetlerinin gereksinimlerini karşılamak için etkili bir keşif mekanizması gereklidir [73]. Günümüzde, İnternet için en geniş keşif mimarisi, çoklu yayın DNS (mDNS) ve DNS Servis Dizini (DNS-SD) uzantıları aracılığıyla hizmetlerin türüne ve özelliklerine göre sorgusunu ve keşfini sunan Etki Alanı Adı Sistemleridir (DNS)[74].

2.6 Uygulama Protokolleri

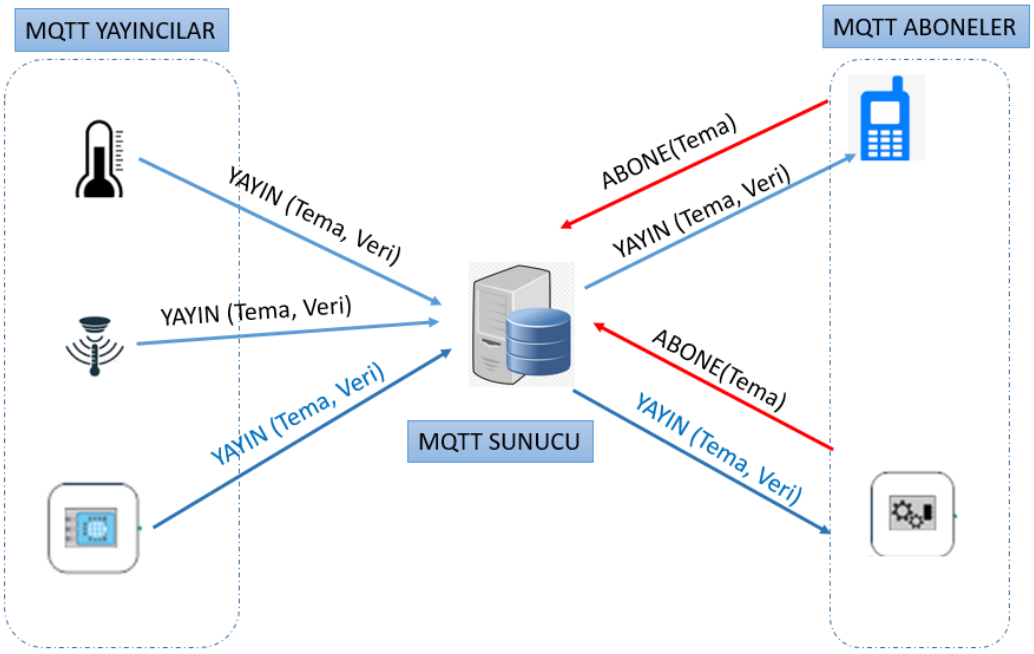
Uygulama katmanı, bilgisayar ağları için OSI modelinin yedinci ve son katmanı olup, etkili iletişimin sağlanması amacıyla bir uygulama programı için hizmetler sağlar [75]. Uygulama

katmanı, kullanıcıların ihtiyaç duyduğu hizmetleri sağlayan bir katmandır. Bu katmanda çalışan protokoller, yerel ağlardaki ve Internet'teki ağ geçitleri arasındaki bilgileri işleyen, kullanıcıların ya da uygulamaların ihtiyaç duyduğu verilerin ve bilgilerin sağlandığı protokollerdir.

İnternetin kullanım senaryoları incelendiğinde, HTTP protokolünün uygulama katmanı protokolü olarak çok geniş ölçüde kullanıldığı görünmektedir. Ancak bu protokol, Nesnelerin İnterneti uygulamaları için uygun olmadığından, IoT gereksinimlerini sağlıklı olarak karşılayacak protokoller geliştirilmiş ya da mevcut bazı protokoller bu doğrultuda güncellenmiştir. Bu bölümde, en yaygın kullanılan IoT Uygulama Protokolleri ele alınmıştır.

2.6.1 Message queuing telemetry transport - MQTT

Bu açık standart protokol, özellikle uzak konumlarda yer alan, kısıtlı bant genişliği bulunan konumlar arasında veri transferi sağlamak amacıyla geliştirilmiş, iletişimde veri bütünlüğü ve güvenliği sağlayan, düşük ek yüklü hafif bir uygulama katmanı protokolüdür. IoT ekosisteminde, makine-makine (M2M) iletişiminde yayın – abone mimarisinde çalışan ve iletim katmanında TCP protokolüne dayanan bu protokol, International Business Machine (IBM) tarafından tasarlanmıştır. Özellikle çok sayıda cihazdan bilgi alınması gerektiği ve bu bilgilerin diğer alıcılara aktarılması gerektiği senaryolarda yaygın olarak kullanılır. MQTT protokolünün yapısında genel olarak istemci ve aracı sunucudan (broker) oluşan iki rol bulunmaktadır. İstemci rolü ise yayıncı (publisher) ya da abone (subscriber) olabilir [76]. İstemci rolünde bulunan abone ile yayıncı arasındaki iletişimin koordinesi, sunucunun sorumluluğundadır. Veri içeren tüm iletişim, “tema (topic)” olarak isimlendirilen bir dizi karakter içermektedir [77]. Yayıncı, gönderdiği her mesajı bir tema ile işaretler ve sunucuya gönderir. Yayıncı rolündeki istemci, tema ile işaretlenmiş ve veri içeren bir ileti gönderdiğinde, aracı rolündeki sunucu bu iletiyi alır ve bu temaya abone olan istemcilere bu mesajı yayınlar. Şekil 2.10’da MQTT sunucusu, abone ve yayıncı rollerindeki cihazlar ve bu cihazlar arasındaki iletişim gösterilmiştir.



Şekil 2.10. MQTT protokolünde bileşenler

Tema kavramı genellikle sunulan hizmet ve konum ile ilişkili olup, ad ve düzey ile belirtilir. Farklı düzeyde bulunan temalar hiyerarşik bir yapıda birbirlerinden "/ " karakteri ile ayrılır. Örneğin, bir su deposunda bulunan bir sensör tarafından sunulan, su seviyesine ilişkin tema aşağıdaki gibi olabilir:

Tesis1/Depo2/SuSeviyesi

Aynı konumda bulunan diğer bir sensör tarafından sunulan, su sıcaklığı bilgisine ait tema bilgisi için aşağıdaki gibi bir dizi kullanılabilir:

Tesis1/Depo2/Sicaklik

Yayıncı rolündeki her bir sensör tarafından gönderilen ve veri içeren her mesaj, ilgili tema bilgisini içermektedir. Gönderilen mesajlar sunucu tarafından alınır ve bu temalara abone olan istemcilere yayınlanır.

Ayrıca tema isimlerinde "+" ve "#" olmak üzere joker karakterler de kullanılabilir. Böylece bir istemci birden çok temaya / düzeye abone olabilir. Örneğin, yukarıdaki örnekte, Tesis1'deki

Depo2'den sunulan tüm temalara abone olunabilir. Bu durumda MQTT istemcisi, aşağıdaki gibi bir tema gösterimine abone olmalıdır:

Tesis1/Depo2/#

Yine Tesis1'de yer alan tüm konumlara ilişkin "sıcaklık" bilgisine abone olabilmek için aşağıdaki gibi bir tema kullanılabilir:

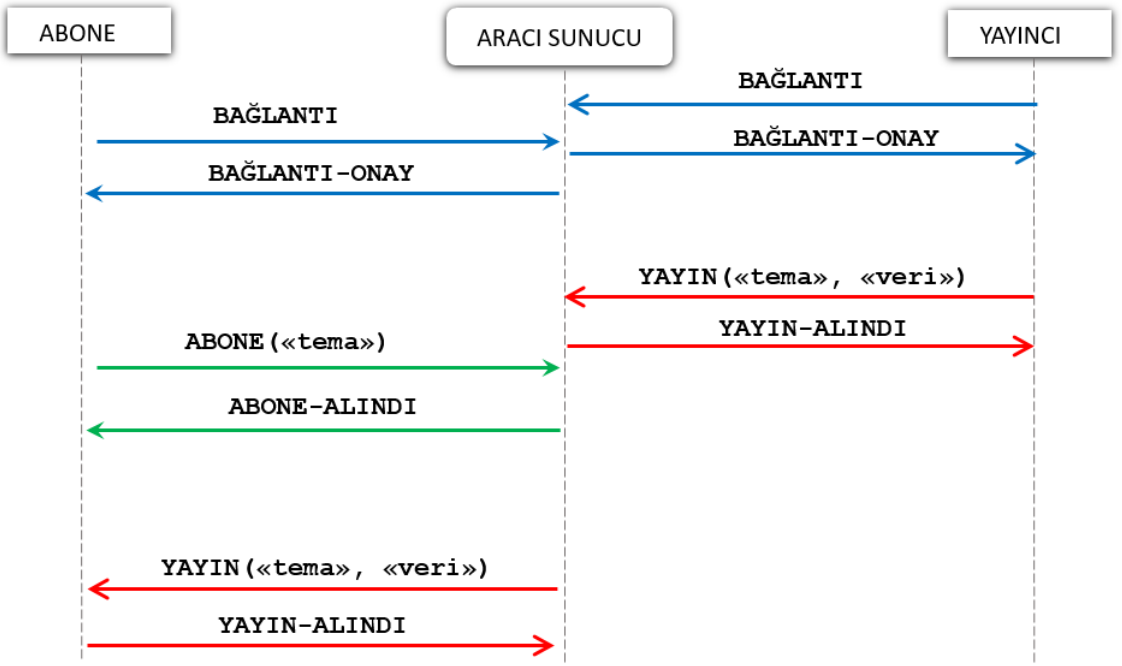
Tesis1/#/Sicaklik

Yayın abone mantığındaki iletişim, birden fazla cihaz ile M2M iletişimin gerçekleşmesi ihtiyacı olan IoT ekosistemlerinde yaygın olarak kullanılan bir yapıdır. MQTT'de verilerin alıcıya ulaşıp ulaşmadığının garantisi ise üç farklı hizmet kalitesi (QoS- Quality of Service) ile sağlanmaktadır [78]. QoS düzeyleri, mesaj iletimini garanti etme yöntemidir ve aracı sunucu ile istemci arasındaki iletişimin güvenilirliğini ifade eder.

- QoS 1 (Sadece bir kez): Bu hizmet seviyesinde, gönderilen mesajın alındığına ilişkin bir geri bildirimde bulunulmaz. Bu nedenle, alıcı tarafından mesajın alınıp alınmadığı bilinemeyeceğinden, en güvensiz hizmet kalitesi seviyesi olarak görülür. Bu düzeyde gönderilen mesaj, gönderici tarafından tekrar gönderilme amacıyla saklanmaz, gönderimin arından hemen silinmektedir.
- QoS 2 (En az bir kez) : Bu düzeydeki hizmet seviyesinde, mesajı yayınlayan gönderici, mesajın bir kopyasını ileti kuyruğunda bekletir ve alıcıdan alındı onayının gelmesini bekler. Gönderici, bu geri bildirimi aldığında, mesajı gönderme kuyruğundan siler. Aksi durumda mesaj tekrar iletilmek üzere gönderilir. Ancak, mesaj hedefe ulaşsa bile, alıcı tarafından gönderilen onay bildirimi göndericiye ulaşmadığı takdirde, gönderici mesajın ulaşmadığını varsayıp, bu mesajı tekrar gönderecektir. Bu durumda aynı mesajın çoklu kopyaları hedefe gidebilir.
- QoS 3 (Tam olarak bir kez): Bu hizmet düzeyi, gönderen ile alıcı arasında dört mesajlık bir sıraya sahip olan, ana mesajın gönderildiğini ve onayın alındığını doğrulamak için bir tür el sıkışma olan, en yüksek hizmet seviyesidir. El sıkışma tamamlandığında, hem gönderen hem de alıcı, iletinin tam olarak bir kez gönderildiğinden emin olmaktadır.

IoT uygulamalarında, ağın güvenilirliği, kullanılabilir bant genişliği ve kullanılan uygulamaların gereksinimleri farklılık göstereceğinden, bu hizmet düzeyleri farklı amaçlar doğrultusunda kullanılabilir. Az sayıda paket gönderileceğinden, güvenilir ağlarda, düşük bant genişliği bulunan ağlarda ya da paket kayıplarının göz ardı edilebileceği uygulamalarda QoS 0 düzeyinin kullanılması uygun olacaktır. Aynı şekilde, hangi düzeyin kullanılacağı, IoT ortamın gereksinimlerine bağlı olarak değişkenlik göstereceğinden, verimli bir iletişim için doğru seviyenin kullanılması sağlanmalıdır.

MQTT protokolünün sunduğu iletişimin sağlıklı bir şekilde gerçekleşmesi için, farklı amaçlar doğrultusunda kullanılan on dört paket türü bulunmaktadır. MQTT QoS1 düzeyinde kullanılan temel paketler Şekil 2.11'de gösterilmiştir.



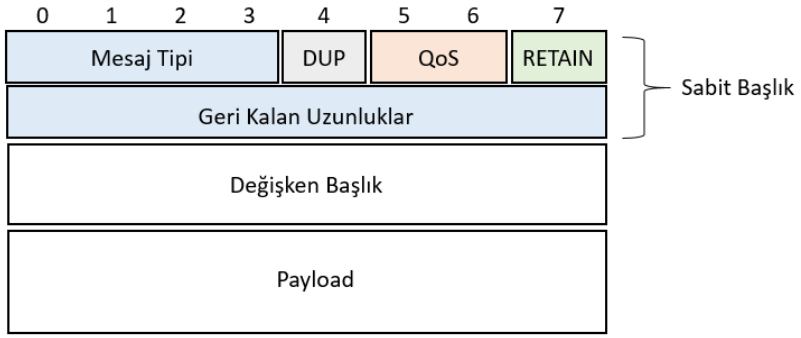
Şekil 2.11. MQTT paket türleri

Şekil 2.11'de görüleceği üzere, Abone ya da Yayıncı olan MQTT istemciler, iletileri gönderebilmek ya da alabilmek için, öncelikli olarak sunucu ile bağlantı kurmak zorundadırlar. Bu amaç doğrultusunda, istemci tarafından " Bağlantı (Connect)" paketi gönderilir ve Sunucu tarafından da "Bağlantı Onay (Connect-Ack)" ile bağlantı isteği onaylanır. Bağlantı paketinde, isteğe bağlı olarak kullanıcı adı ve parola doğrulaması da yapılabilir.

MQTT yapısında veri iletimi ve alındı bildirimi için, “Yayın (Publish)” ve “Yayın-Alındı (PubRec)” paketleri kullanılır. Yayın paketi, herhangi bir tema ile etiketlenmiş ve içerisinde veri barındıran bir iletidir. Bu pakete cevap olarak verilen paket ise, yayıncı tarafından gönderilen mesajın alındığına ilişkin, QoS 2 düzeyinde kullanılan geri bildirim paketidir. Ayrıca QoS0 dışında bir yapı kullanıldığı durumda, paketin benzersiz olmasını tanımlayan bir “Paket Tanımlayıcı” bulunur.

Abonelik işlemi için kullanılan diğer bir paket çifti de, “Abone (Subscribe)” ve “Abone-Alındı (SubRec)” paketleridir. Abone paketi, herhangi bir temaya abone olmak isteyen istemciler tarafından, aracı sunucuya gönderilen paket türüdür. Sunucu tarafında verilen cevap ise, abonelik işleminin gerçekleştiğini belirten “Abone-Alındı” mesajıdır.

MQTT protokolünün en önemli avantajlarından biri de, küçük boyutlu paket başlığı kullanmasıdır. Şekil 2.12’de MQTT paketinin başlık yapısı verilmektedir [79].



Şekil 2.12. MQTT başlık yapısı

MQTT Paket Türü alanı (mesaj tipi), paketin hangi tür olduğunu tanımlamaktadır. Bayraklar ise, QoS düzeyi, çift alınan mesajın algılanması gibi, paketin türüne göre değişen değerler almaktadır. İkinci bayt ve sonraki alanlar, Abone, Yayın ve bu paketler ile ilgili diğer mesaj türlerinde, “Tema” ve “Veri” alanlarını içermektedir. Yine bu alan MQTT bağlantı paketlerinde kimlik doğrulama amacıyla kullanılmaktadır.

2.6.2 Constrained application protocol- CoAP

CoAP, UDP protokolü üzerinden temsili durum aktarımı (REST – Representational State Transfer) tabanlı olarak çalışan, uç noktalar arasında mesaj transferini sağlayan bir

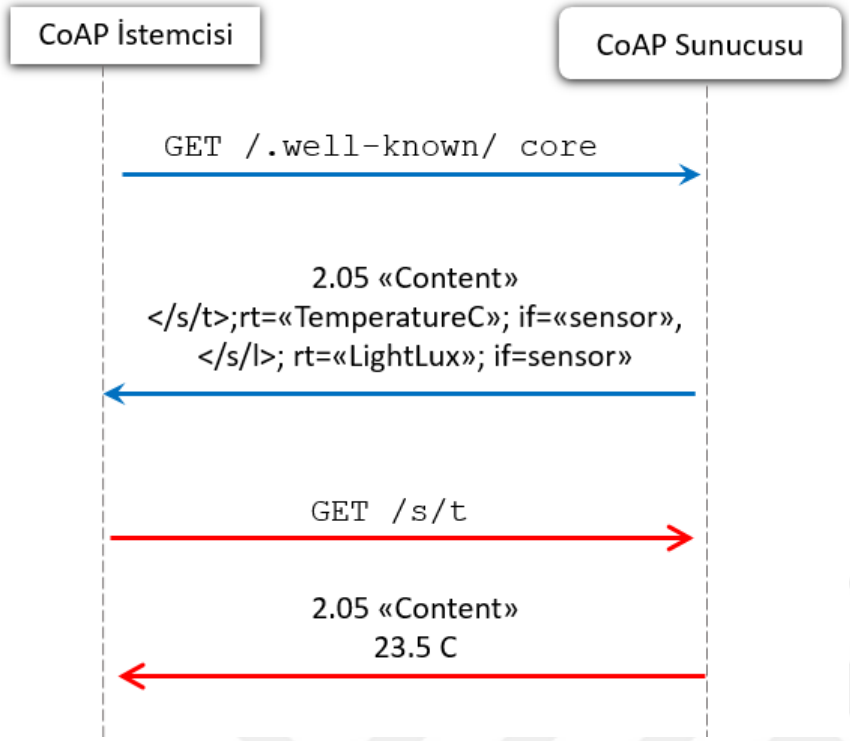
protokolüdür. Kısıtlı düğümler arasında ve kısıtlı ağ ortamlarında kullanılmak amacıyla, IETF çalışma grubu CoRE tarafından tasarlanan, istek – cevap mantığında çalışan özelleştirilmiş bir web taşıma protokolüdür [25]. Genel olarak, M2M iletişimde kullanılmak amacıyla tasarlanan bu protokolde, istemci ve sunucu yapısında iki rol bulunmaktadır. Belirli bir hizmeti sunan ve CoAP çalıştıran her nesne, bir CoAP sunucusu olarak görev yapmaktadır. CoAP, HTTP ile aynı ilkeleri kullanır, ancak kısıtlı cihazlarda çalıştırılabilmesi için çok daha hafiftir [80]. CoAP istemciler, sunucu üzerinde verilen hizmete erişmek için http yapısındakine benzer, GET ve POST gibi istek paketleri göndermekte ve sunucu tarafından bu mesajlara cevap paketleri ile karşılık verilmektedir.

MQTT yapısında kullanılan tema kavramı yerine burada, Evrensel Kaynak Tanımlayıcı (URI - Universal Resource Identifier) adresleri kullanılmaktadır. Bu adres yapısı, web iletişiminde kullanılan adreslere benzerdir. Örnek bir URI adres tanımlaması aşağıda verilmiştir:

GET coap://192.168.1.1: 5555 /depo1/suSeviyesi

CoAP sunucusu gelen bir isteği işler ve bir yanıt kodu ile beraber veri içeren bir yanıtı geri gönderir. HTTP'den farklı olarak CoAP, bu iletişimi UDP aktarım protokolü üzerinden eşzamansız olarak yapmaktadır.

CoAP protokolünün önemli avantajlarından biri, servis keşif mekanizmasını içinde barındırmasıdır. M2M iletişimlerde önemli olan keşif mekanizması, RFC 6690'da açıklandığı gibi CoRE Link Formatı kullanılarak desteklenmektedir. Bir sunucu tarafından barındırılan kaynaklar, "/.well-known/core" olarak bilinen URI adresinde saklanır. Kullanılabilir kaynakların listesi sunucudan alındığında, istemci belirli bir kaynağın değerini almak için istekler göndermektedir. Şekil 2.13'de CoAP protokolünün çalışma şekli örnek olarak gösterilmiştir.



Şekil 2.13. CoAP iletişim örneği

Şekil 2.13'de istemci tarafından kaynak keşfine dair bir istek gönderilmiş, sunucu tarafından da sıcaklık (/s/t) ve ışık miktarını (/s/l) belirten parametreler ile cevap verilmiştir. Böylece istemci, hangi tür kaynakların bulunduğunu algılayabilmekte ve GET istek mesajları ile erişmek istediği kaynağa ilişkin veri talep edebilmektedir.

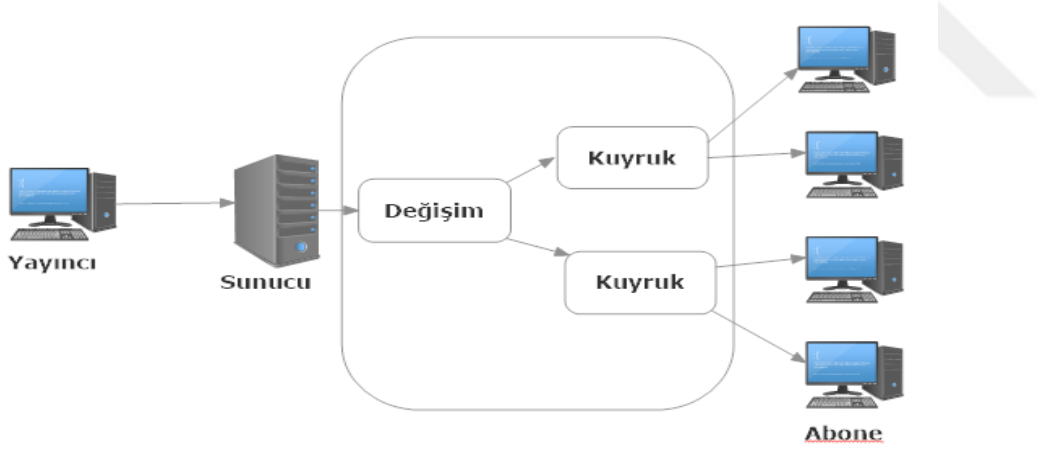
CoAP, trafik iletiminin garantisi için, "onaylanabilir" ve "onaylanamaz" olmak üzere iki ana çalışma şekli de sunmaktadır. Bir paketin bir onay beklemeden tek yönlü olarak gönderildiği durumlar "onaylanamaz" modu olarak bilinir ve bu nedenle başarılı teslimat garantisi sunmamaktadır. Paketin alındığına dair geri bildirim gönderilmesi, dolayısıyla iletinin alınıp alınmadığının anlaşıldığı durum ise "onaylanabilir" çalışma modu olarak anılmaktadır [81].

CoAP protokolünün getirdiği özelliklerden biri de, kaynaktaki değer değişimine ilişkin istemcileri bilgilendirebilme yeteneğidir. Bu özellik MQTT yapısındaki abonelik işlemine benzetilebilir. Buna göre, belirli bir kaynağı izlemek isteyen bir istemci, izleme seçeneği (observe) ile işaretlenmiş bir GET mesajını CoAP sunucusuna gönderir. CoAP sunucusu, kaynak durumunda bir değişiklik olduğunda, istemciye bir bildirim gönderir. Ayrıca, kaynağın durumu değişmezse, ancak son bildirimden bu yana geçen süre kaynağın maksimum süre

değerini aşarsa, sunucu tarafından istemciye tekrar bir bildirim gönderilir. Bu yapı sayesinde, bir kaynağın gözlemlenebilmesi için sürekli olarak istek mesajları göndererek, gereksiz ağ trafiği oluşturmanın önüne geçilmektedir.

2.6.3 Advanced message queuing protocol –AMQP

Mesajlaşma temeline dayanan bu protokol, farklı sağlayıcılardan gelen uygulama verilerinin birlikte çalışabilmesini sağlamak amacıyla geliştirilmiş TCP temelli bir protokoldür. AMQP, mesajları kaybetmemeye odaklanmıştır. Güvenli mesajlaşma sağlanabilmesi için çeşitli mesajlaşma uygulamalarını ve iletişimlerini desteklemek üzere tasarlanmış olup, veri formatı MQTT'de olduğu gibi ikililer şeklindedir [82]. Ayrıca, uç noktalar her mesajın alındığını kabul etmelidir. Geliştirilen API sayesinde, AMQP veri formuna uygun mesajları oluşturup yorumlayabilen herhangi bir uygulama, diğer uyumlu uygulamalar ile haberleşebilmektedir [34].

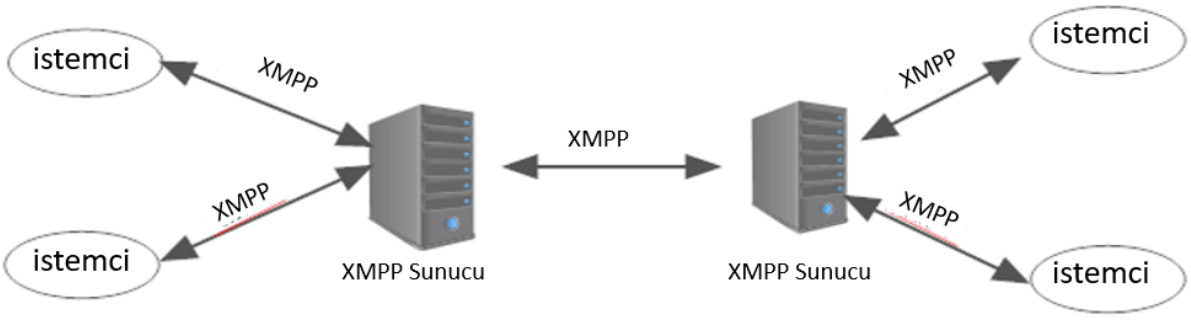


Şekil 2.14. AMQP Protokolü

AMQP'nin yayın/abone mimarisinde *Değişim kuyruğu* ve *Mesaj Kuyruğu* olmak üzere iki bileşeni bulunmaktadır (Şekil 2.14). Değişim kuyruğu, mesajları uygun sıraya yönlendirmeden sorumludur. Mesaj kuyruğu, aboneye iletilinceye kadar mesajları saklar. AMQP farklı platformlar arasında uyumluluğu sağlayacak şekilde tasarlandığından yüksek ölçeklenebilirliğe sahiptir. Ancak kısıtlı kaynaklara sahip cihazlar için ve gerçek zamanlı uygulamalar için uygun değildir.

2.6.4 Extensible messaging and presence protocol - XMPP

Anlık mesajlaşma protokolü olarak geliştirilen, XML biçimini kullanan TCP tabanlı bir protokoldür. IoT yapısında, birbirleri ile iletişim kurması gereken fiziksel nesneleri adreslemede kullanılmaktadır. Sunucu – istemci mantığı ile çalışan bu protokol, uç birimler arasındaki haberleşmede kullanılmaktadır [74].



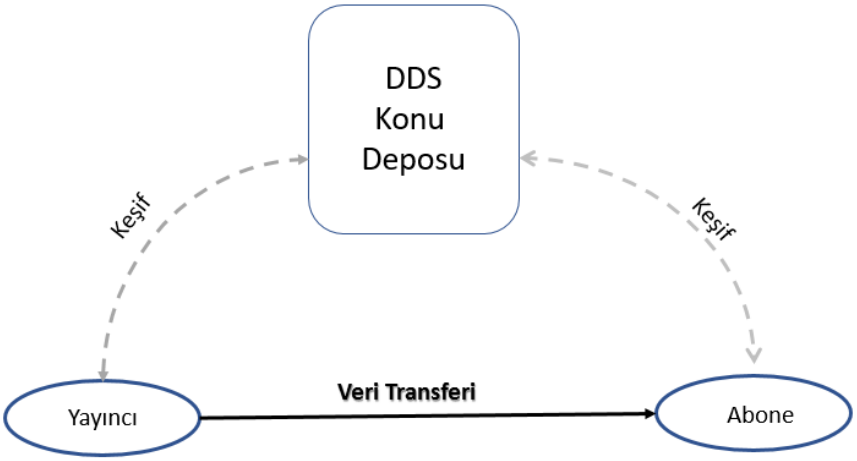
Şekil 2.15. XMPP protokolü

XMPP basit ve esnek bir yapıda olup, heterojen projelerde ve uygulamalarda kullanılabilir. Ancak yüksek bant genişliği tüketimi, yüksek işlemci kullanımı bu protokolün zayıf noktalarındandır.

2.6.5 Data distribution service - DDS

DDS, yayın-abone etkileşim modelinde, Nesne Yönetimi Grubu (Object Management Group - OMG) tarafından tasarlanmış bir uygulama katmanı standardıdır. MQTT ve AMQP protokollerinin aksine DDS merkezi olmayan bir yapıda çalışmakta, yani aracı sunucu yapısı kullanılmamaktadır. DDS yapısında yayıncılar ve aboneler veri yolu üzerinden eşler olarak iletişim kurabilir, ilgi duyulan konuya göre asenkron veri iletişimini mümkün kılmaktadırlar. Aracı bir sunucunun kullanılmaması, MQTT ve AMQP protokollerinde yaşanan tek bir hata noktası (SPoF) oluşumunu önlediğinden, iletişimin bu sebeple kesilme olasılığı düşmekte ve sistem daha güvenilir hale gelmektedir [11]. DDS, veriyi, komutları ya da olayları iletim protokolü olarak UDP aracılığıyla göndermekte ve almaktadır. Ayrıca gerçek zamanlı çoktan çoğa (many-to-many) bağlanabilirliği ve otomatik keşfi desteklemektedir. Bu sistemin sunduğu

avantajlardan biri de, otomatik geçiş yapabilme yeteneğidir. İlk yayıncı herhangi bir sebepten dolayı başarısız olursa, DDS otomatik olarak diğer bir yayıncıyı devreye alabilmektedir [4].



Şekil 2.16. DDS abone ve yayıncıları

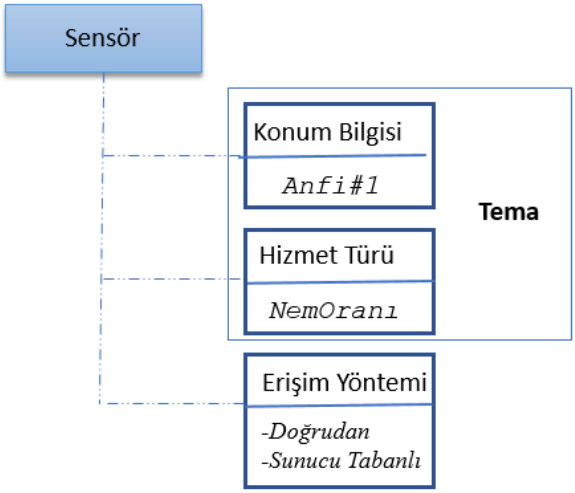
IoT sistemlerinde kullanılan cihazlar, veriyi geleneksel bilişim altyapısının talep ettiğinden farklı şekilde talep etmektedir. Genellikle cihaz talepleri mikro saniyeler düzeyinde çok daha hızlı veya gerçek zamanlı ya da yarı gerçek zamanlı olabilmektedir. Yine cihazlar diğer cihazlar ile çok karmaşık yollarla iletişim kurabilmektedir. Bu nedenle TCP'nin basit ve güvenilir olan noktadan-noktaya yaklaşımı çok kısıtlayıcı olabilmektedir. Ancak, DDS ayrıntılı hizmet kalitesi (QoS) kontrolü, çoklu noktaya yayın, yapılandırılabilir güvenilirlik ve yaygın yedeklilik sunmaktadır. DDS, tam olarak hangi verilerin nereye gittiğini filtrelemek ve seçmek için güçlü yöntemler sunmaktadır.

3. GENEL AMAÇLI IOT UYGULAMA PROTOKOLÜ TASARIMI

Nesnelerin İnternetinin etkili ve verimli olarak uygulanmasında, uygulama protokolleri önemli rol oynamaktadır. MQTT, XMPP, AMQP gibi merkezi sunucu yaklaşımı ile çalışan protokollerin yanı sıra, DDS ve CoAP gibi merkezi olmayan yaklaşımlar ile çalışan protokoller de IoT ekosisteminde yaygın olarak kullanılmaktadır. Nesnelerin İnternetinin doğası gereği, çeşitli türde nesnelerin kurulu olduğu ortamlarda, hem doğrudan erişim gerektiren senaryolar hem de sunucu tabanlı yaklaşımlar kullanılmaktadır. Bu bölümde, her iki yaklaşımı birden kullanabilen, genel amaçlı hibrit bir protokol tasarlanmaktadır.

3.1 Genel Özellikler

Nesnelerin İnterneti son birkaç on yılda ivme kazanarak internetin bir uzantısı olarak endüstriye girmektedir. Çeşitli cihazların farklı teknolojiler kullanarak ağa bağlanması sayesinde, daha akıllı nesnelerin internet ortamında yer alması, beraberinde yeni zorlukları da getirmiştir. Bağlı olan bu çeşitli nesnelerin birlikte çalışabilirliği, geleneksel bilişim teknolojileri cihazlarından farklı konumlarda bulunan nesnelerin ve bu nesnelerden türetilen verilerin güvenliğinin sağlanması, uygulamalarda kullanılan protokollerde yaşanan performans sorunlarının giderilmesi ve kısıtlı kaynaklara sahip IoT cihazlarının ihtiyaçlarını karşılayabilmesi gibi konular IoT'de çözülmesi gereken zorluklardandır. Akademik araştırmalarda bu problem durumlarını ele alan çalışmalar yapılmaktadır. Ancak bu tez çalışmasında bu sorunların tamamının çözülmesine odaklanılmamıştır. Çalışmada, bazı sınırlılıklar bulunmakta ve çeşitli varsayımlar dikkate alınmıştır. Öncelikli olarak, Nesnelerin İnternetindeki güvenlik konuları dikkate alınmamış, IoT ekosisteminin kampüs alan ağları gibi ölçekte kapalı bir yapıda olduğu ve iletişimin güvenli olduğu varsayılmıştır. Bu nedenle, IoT güvenlik konusu kapsam dışına alınmaktadır. Bu çalışma kapsamında, IoT sistemlerinde nesnelerin izlenmesini sağlayan sensörlere ilişkin Şekil 3.1'de gösterilen bir yapı tasarlanmıştır.



Şekil 3.1. Sensöre ilişkin alınan veriler

Buna göre, MQTT protokolünde kullanılan "tema" kavramına ve CoAP protokolünde kullanılan URI yapısına paralellik göstermesi ve bu protokollerden gelen verileri algılayabilmesi amacıyla, temel olarak sensörün hizmet verdiği konum ve verdiği hizmet türü dikkate alınmaktadır. Ayrıca, protokolün hibrit yapısı gereği, bu donanıma doğrudan ya da sunucu üzerinden erişeceğini belirten parametre kullanılmaktadır. Sensörlerin veritabanına kayıt ettirilmesi esnasında bu bilgiler kullanılmaktadır.

Tez çalışması kapsamında, genellikle küçük veriler üreten sensör gibi donanımlardan elde edilen bilgiler IP paketleri aracılığıyla taşınmaktadır. Bu nedenle veri iletimi esnasında, IP paketlerinin bölünmelere uğramadığı, tek bir paket halinde taşınabilir boyutta olduğu varsayılmaktadır.

Nesnelerin İnterneti uygulamalarında yer alan donanımlar genellikle sınırlı olanaklara sahip cihazlar olduklarından, yüksek işlemci gereksinimi, yüksek bellek kullanımı ve yüksek bant genişliği kullanımı gerektiren uygulamalarda, ihtiyacı karşılamada eksik kalmaktadırlar. Bu nedenle diğer IoT uygulama katmanı protokollerinde olduğu gibi, sınırlı sahip donanımlarda sağlıklı olarak uygulanabilecek esneklikte ve bu cihazlarda kolaylıkla çalışacak şekilde geliştirilmiştir.

Geliştirilen protokolün çalışmasına destek olmak üzere, MQTT yapısındakine benzer bir şekilde, koordinatör, istemci ve IoT Ağ Geçidi (IoTGW) olmak üzere üç rol (bileşenler) bulunmaktadır.

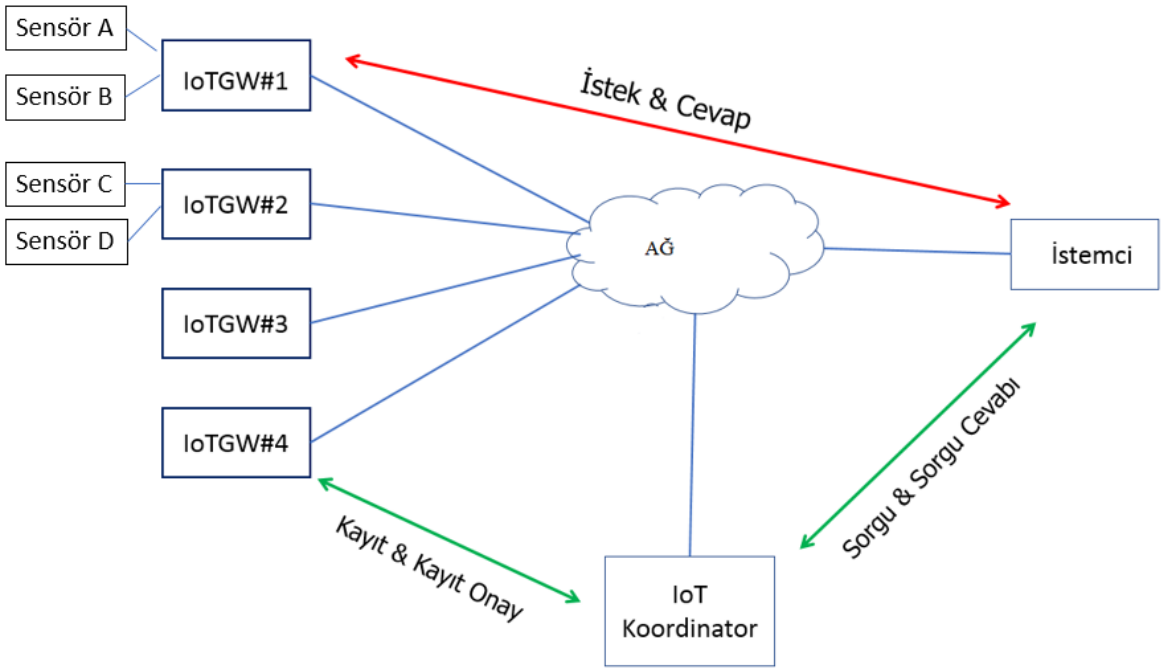


Şekil 3.2. Hibrit çalışma modeli

Şekil 3.2'de gösterildiği üzere, istemcinin hem sunucu üzerinden, hem de doğrudan bir şekilde IoT Ağ Geçidi cihazına erişimi mümkün olmaktadır. Bu özellik, protokolün hibrit olarak adlandırılmasının ana gerekçesini teşkil etmektedir. Şekil 3.1'de gösterilen sensör erişim türüne göre, bu iki yaklaşımdan herhangi biri kullanılabilmektedir. Sensörlerden sağlanan (sıcaklık, nem, ışık şiddeti gibi) hizmetler IoT Ağ Geçidi aracılığıyla sunulur. Bu hizmetlerin her biri, hibrit uygulama protokolünde bir "tema" ile ilişkilendirilmektedir.

3.2 Genel Çalışma Mimarisi

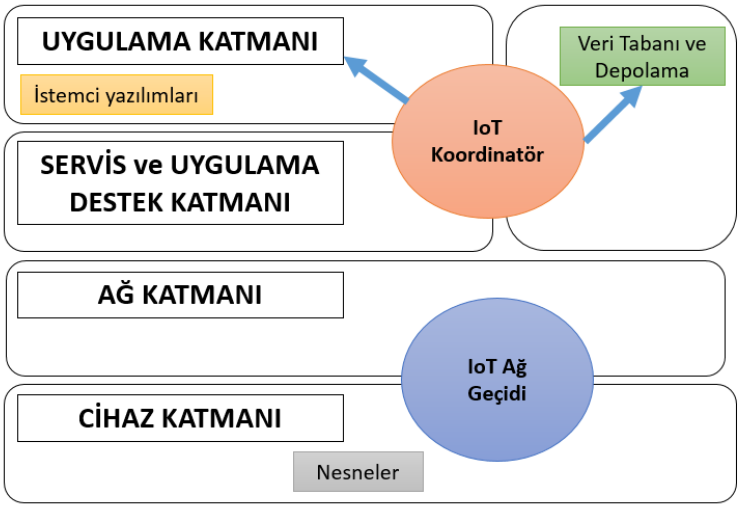
Geliştirilen protokolün çalışma prensibinin anlaşılabilmesi için, yapıda kullanılan bileşenlerin görevlerinin doğru bir şekilde tanımlanması ve iletişim paketlerinin ayrıntılı olarak ele alınması gerekmektedir.



Şekil 3.3. Geliştirilen protokole genel bakış

Şekil 3.3'de protokolde belirtilen üç bileşen arasındaki paket akışları gösterilmektedir. IoT Ağ Geçidi, verinin kaynağı olan ve sensörlerden elde edilen verilerin ağ ortamına sunulması görevini üstlenmektedir. İstemci bileşeni, veriye ulaşmak isteyen kullanıcıyı, donanımı ya da uygulamayı temsil etmektedir. Koordinatör bileşeni ise, sunucu tabanlı iletişimde veri transferi için aracı sunucu görevi görmektedir. Aynı şekilde, ağda IoT Ağ Geçidi bileşenler aracılığıyla sunulan hizmetlerin kayıtları yine koordinatör bileşende tutulmaktadır. Bu bileşenlerin işlevsellikleri, çalışma prensipleri ve görevleri *3.6 Mimaride Kullanılan Bileşenler* başlığı altında detaylı bir biçimde incelenmektedir.

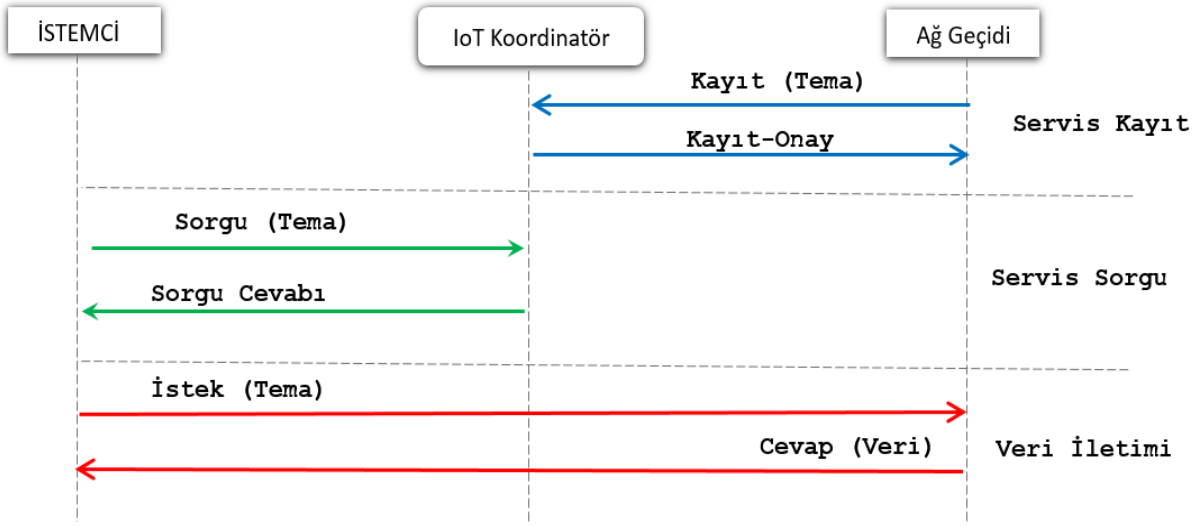
Şekil 3.4'te katmanlı mimarisi gösterilen genel amaçlı hibrit uygulama protokolü geliştirilirken, ITU-IoT Modeli referans alınmıştır.



Şekil 3.4. Geliştirilen protokol mimarisi

Geliştirilen protokoldeki IoT Ağ Geçidi cihazı, fiziksel katman olarak da anılan, cihaz katmanı ile Ağ Katmanı arasında görev yapmaktadır. Nesnelerden elde edilen verilerin ağ ortamına sunulmasından sorumludur. IoT koordinatörü servis ve uygulama destek katmanı ile uygulama katmanı arasında görev yapmaktadır. Depolama yapabilme yeteneği sayesinde veritabanı uygulamalarına ya da kullanıcı uygulamalarına veri aktarabilmektedir.

Mimaride kullanılan aşamalar ve her aşamadaki mesaj türleri Şekil 3.5'de gösterilmektedir. Genel olarak, bu aşamalar Servis Kayıt, Servis Keşif ve Veri Transferi aşamalarıdır. Bu aşamalar ve her aşamada gerçekleşen olaylar "*3.5 hIoT Çalışma Aşamaları*" başlığı altında ayrıntılı olarak ele alınmaktadır.



Şekil 3.5. Protokolde kullanılan aşamalar

Fiziksel katman ile Ağ katmanı arasında görev yapan IoT Ağ Geçidi bileşeni, Servis Kayıt Sürecinde, kendisine doğrudan bağlı olan nesnelerden sunulan hizmetlerin koordinatöre kayıt ettirilmesi görevini yapmaktadır. Kullanıcıların ya da uygulamaların erişmek istedikleri hizmetlere nasıl ulaşacağına ilişkin yaptıkları sorgulamalar ve koordinatör tarafından hizmete erişim bilgilerini içeren cevap, mimarinin servis sorgu aşamasını oluşturmaktadır. Sunucu bu aşamada erişim yöntemini, yani doğrudan erişim ya da sunucu üzerinden erişim cevabını verebilmektedir. İstemci ile hizmeti sunan Ağ Geçidi arasındaki doğrudan ya da sunucu aracılığı ile gerçekleşen veri alış verişi, veri transferi ya da veri iletimi aşamasını belirtmektedir.

3.3 Paket Türleri

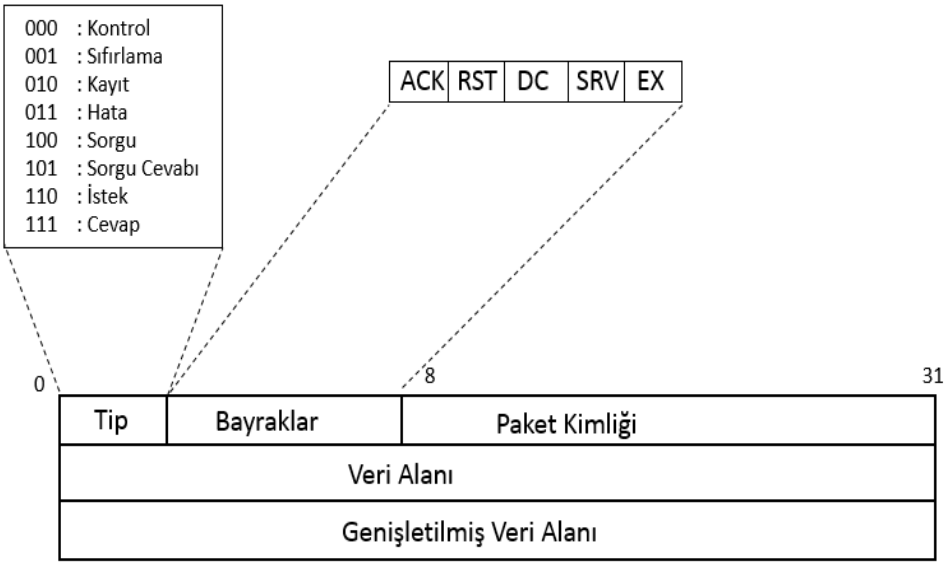
Geliştirilen modelde, her aşamada gerçekleşen ve amacı farklı olan iletişim türleri bulunmaktadır. Bu iletişim türleri, hibrit uygulama protokolünün paket türleri ile belirlenmektedir. Her paket türü, iletişim sürecindeki birbirinden farklı olan, sorgu, keşif ve iletim gibi adımlara karşılık gelmektedir. Protokolde, bu adımları karşılamak amacıyla sekiz paket türü bulunaktadır. Paketlerin isimleri ve görevleri Çizelge 3.1'de belirtilmektedir.

Çizelge 3.1. Geliştirilen protokolün paket türleri

Paket Türü	Değer (Onluk – İkilik)	Görevler
Kontrol Paketi	0 - 000	IoT Koordinatörün ve IoT Ağ Geçidinin erişilebilir olup olmadığını doğrulamak amacıyla kullanılmaktadır.
Sıfırlama Paketi	1 - 001	IoT Ağ Geçidi üzerinden verilen hizmetin erişim türümünü sıfırlamak amacıyla kullanılmaktadır.
Kayıt Paketi	2 - 010	IoT Ağ Geçidi aracılığıyla sunulan hizmetlerin, IoT Koordinatör sunucusuna kayıt sürecinde ve bu kayıt işlemine verilen cevapta kullanılmaktadır.
Hata Paketi	3 - 011	İletişimde yaşanan çeşitli hata durumlarında gönderilen ve hata kodu içeren pakettir.
Sorgu Paketi	4 - 100	Veriye erişmek isteyen istemcinin servis keşfi sürecinde kullandığı, hizmetin erişim bilgilerinin istendiği pakettir.
Sorgu Cevabı Paketi	5 - 101	Servis keşfi paketine koordinatör tarafından verilen cevaptır.
İstek Paketi	6 - 110	İstemci tarafından IoT Ağ Geçidinde sunulan hizmete ilişkin bilgi isteme paketidir.
Cevap Paketi	7 - 111	IoT Ağ Geçidi tarafından istek paketine cevap olarak gönderilen istemci tarafından istenen veriyi içermektedir.

3.4 hIoT Paket Başlığı

Paket başlığı, hibrit uygulama protokolünde kullanılan paketlerin türlerinin belirlendiği, iletişimin gerçekleşmesi için bayrakları, adreslemeleri ve ek yükü (payload) barındıran bir yapıdır. Geliştirilen protokolün başlık bilgisi, Şekil 3.6'da gösterilmektedir [5].



Şekil 3.6. Paket başlık yapısı

Hibrit Uygulama Protokolünde kullanılan başlık bilgisi sabit olan 4 bayttan (32-bit) ve veri alanlarından oluşmaktadır. Paket başlığındaki ilk alan 3-bitten oluşan "Tip" alanıdır. Bu alanda bulunan farklı bit dizilişleri paket türünü gösteren değerlerden oluşmaktadır. Paket başlığında yer alan "Bayraklar" alanı, toplamda 5-bitten oluşmaktadır. Bu bayraklar iletişimin gerçekleşmesi için gerekli olan, her biri tek bitten oluşan bilgileri saklamaktadır. Her paketin benzersiz olmasını sağlayan "*Paket Kimliği*" alanı 24 bitten oluşmaktadır.

Paket başlığında yer "*Veri Alanı*", protokolde veri taşınması sürecinde kullanılan 32 baytlık bir alandır. "*Genişletilmiş Veri Alanı*" ihtiyaç durumunda daha büyük boyutlarda verileri taşımak amacıyla geliştirilmiştir. Bu alan 1024 bayt boyutuna kadar ek yük taşıyabilmekte olan ve "EX" bitinin (bayrağının) durumuna göre kullanılıp kullanılmayacağı kararlaştırılan seçimlik bir alandır.

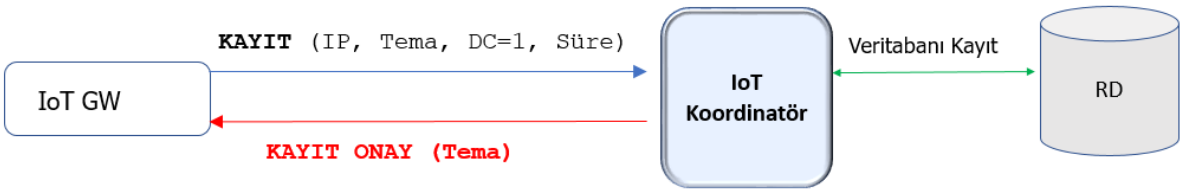
5 bitlik bayraklar alanının ilk biti ACK bayrağıdır. Bu bir bitlik alan, paketlerin cevaplanması sürecince kullanılan onaylama paketlerini belirlemede kullanılmaktadır. İkinci bayrak, IoT Ağ Geçidi üzerinden sunulan hizmetin erişim türünü değiştirmek amacıyla kullanılan sıfırlama paketinde kullanılmaktadır. Bu alan "1" olarak işaretlenmişse, paketi alan Ağ Geçidi donanımı, hizmet için sunulan doğrudan erişim özelliğini kaybetmektedir. Bu paketi alan IoT Ağ Geçidi donanımı, koordinatör haricindeki herhangi bir cihazdan gönderilen hiçbir paketi dikkate almamaktadır. Bu durumda, ilgili hizmete erişebilmek için sunucu üzerinden iletişim kurmak zorunlu olmaktadır. Erişim yöntemi DC bayrağı ile belirlenmektedir. Bu bayrak set edilmişse (değer "1" ise), sunulan hizmete doğrudan erişim sağlanabilmektedir. Bayraklar içerisinde yer alan SRV biti, trafiklerin sunucudan kaynaklanan paketleri olduğunu göstermektedir. Diğer bir ifade ile sunucunun ürettiği her pakette SRV biti "1" olarak işaretlenmektedir. Paket içerisinde yer alan ek veri taşıma alanının (Genişletilmiş Veri Alanı) kullanılıp (EX="1") kullanılmayacağı (EX="0") EX bitine göre belirlenmektedir.

3.5 hIoT Çalışma Aşamaları

Tasarlanan mimaride hibrit uygulama protokolünün, Servis Kayıt, Servis Keşif ve Veri Transferi aşamaları bulunmaktadır. Bu üç çalışma aşamaları ve paketlerin kullanım yeri Şekil 3.5'te özetle belirtilmiştir. Bu başlıkta, aşamaların her biri detaylı olarak ele alınmaktadır.

3.5.1 Servis kayıt aşaması

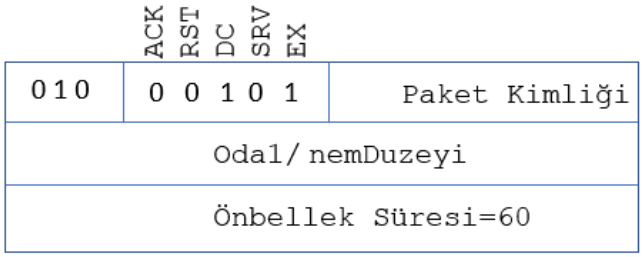
Tez çalışması kapsamında geliştirilen mimarideki aşamalardan ilki servis kayıt sürecinin yaşandığı servis kayıt aşamasıdır. Bu fazın amacı, IoT Ağ Geçitleri aracılığıyla verilen hizmetlerin her birisinin tema bilgileri ile eşleştirilerek, erişim parametrelerinin tutulduğu kaynak dizini (RD) oluşturmaktır. Servisler genellikle sensörler tarafından sunulan hizmetlere dayanmaktadır. IoT sistemlerinde hizmetlerin ya da sensörlerin kayıt edilmesi süreci, otomatik, yarı-otomatik ve elle kayıt olmak üzere üç farklı şekilde yapılmaktadır. Bazı IoT uygulamalarında kayıt ettirilecek hizmetlerin sayısı az olabilmektedir. Bu gibi durumlarda elle ya da yarı-otomatik yöntemlerle kayıt ettirilmesi mümkün olmaktadır. Oysa çoğu IoT uygulamalarında sensör ya da sensörden sağlanan hizmet sayısı çok fazla ve çeşitli olabilmektedir. Böylesi bir durumda, gereksiz iş yükü oluşturmamak adına otomatik kayıt yöntemleri kullanılması daha uygun olmaktadır[83]. Bu tez çalışması kapsamında geliştirilen protokolde, sensörler aracılığıyla sunulan servislerin IoT Ağ Geçidine yarı-otomatik yöntemlerle kaydedilmesini mümkün kılan bir yaklaşım geliştirilmiştir. Her biri bir tema ile ilişkilendirilen ve sensör üzerinden alınan ancak IoT Ağ Geçidi üzerinden sağlanan çeşitli hizmetler, konum parametresi ve erişim yöntemi ile eşleştirilerek IoT koordinatöre gönderilmektedir. IoT Koordinatör, elde ettiği bu verileri kaynak dizin oluşturmak amacıyla kullanabilmektedir. Ayrıca, daha büyük ölçekli yapılara destek sunabilmek amacıyla, bu veri tabanını harici bir sunucuda da tutmak mümkün olabilmektedir.



Şekil 3.7. Servis kayıt aşaması

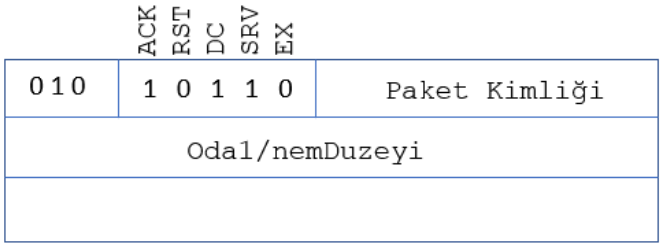
Servis kayıt aşamasında (Şekil 3.7), IoT Ağ Geçidi üzerinden sunulan hizmete ilişkin, Internet Protokol (IP) erişim bilgisi, doğrudan hizmete erişim yetkisinin (DC-Direct Communication) sunulup sunulmayacağı gibi bilgilerin yanı sıra, bu hizmete ilişkin gönderilen verinin ne kadar süre içerisinde zaman aşımına uğrayacağı (önbellekte tutulacağı) bilgileri de aynı şekilde IoT Koordinatöre iletilmektedir. Kayıt paketini alan IoT Koordinatör, önbellekleme zamanını, tema bilgisini ve doğrudan erişim bilgileri ile beraber paketi gönderen IoT Ağ Geçidinin IP adresini

kaynak dizinine kaydetmektedir. Kayıt işleminin ardından IoTGW bileşenine, ilgili tema bilgisi ile beraber onay bilgisi gönderilmektedir.



Şekil 3.8. Servis kayıt paket başlık yapısı

Şekil 3.8'de herhangi bir IoTGW cihazı tarafından sağlanan "Oda1/nemDuzeyi" tema bilgisine ilişkin örnek Kayıt paketinin başlık yapısı ve içeriği görüntülenmektedir. İlk bitlerdeki "010" dizilişi, bu paketin bir Kayıt Paketi olduğunu göstermektedir. DC bayrağının "1" olarak işaretlenmiş olması, IoTGW tarafından bu tema bilgisine doğrudan erişim sağlanabileceği ifade edilmektedir. Ek yük alanında "Oda1" konumunda "Nem düzeyi" bilgisi için verilen yaşam süresinin 60 saniye olduğu belirtildiğinden, EX bayrağı set edilmiş olarak görüntülenmektedir.



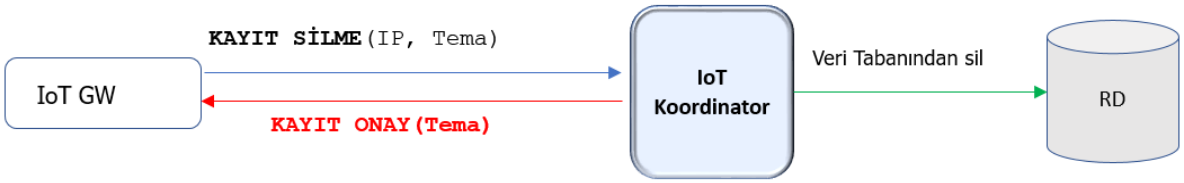
Şekil 3.9. Kayıt onay paketi başlık bilgisi

Şekil 3.8'de gösterilen Kayıt Paketine verilen cevap, kayıt onay paketidir. Bu paketin başlık yapısı ve örnek içeriği Şekil 3.9'da gösterilmektedir. Bu örnekte belirtilen başlık yapısında, "010" bilgisi ve "ACK" bayrağının "1" olarak set edilmesi bunun bir kayıt onay paketi olduğunu göstermektedir. Yine "SRV" paketinin "1" olarak işaretlenmesi, bu paketin sunucu tarafından gönderildiğini göstermektedir.

Bu örnekte, "*Oda1/nemDuzeyi*" tema bilgisi, doğrudan erişimi desteklemekte ve bu tema bilgisi tarafından gönderilecek olan veriler 60 saniye boyunca IoT Koordinatör bileşenin ön belleğinde saklanacağı ifade edilmektedir. Bu önbellekleme süresi, sunucu tabanlı gerçekleşen

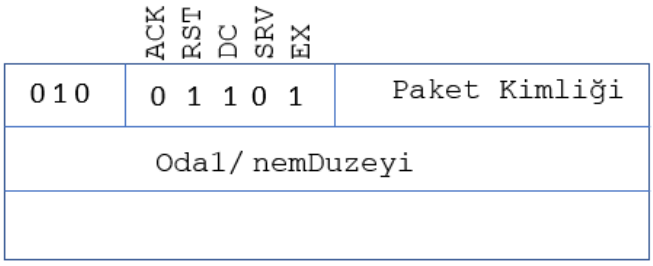
iletişimlerde, aynı tema bilgisi için gelen veri taleplerinin tekrar IoT Ağ Geçidi bileşenine gönderilmeden, doğrudan Koordinatör ön belleğinden cevaplanabileceği ve istemciye gönderilebileceği süreyi göstermektedir.

Bu mimaride, servis kayıt aşamasında IoT Ağ Geçidi cihazları tarafından gönderilen tema, erişim yöntemi, önbellekleme süresi gibi bilgiler, IoT Koordinatörü tarafından sağlanan yerel dizinde saklanmaktadır. Bu yapı, SPOF (tek bir hata noktası) olarak bilinen bir riskin doğmasına neden olmaktadır. Bu nedenle, potansiyel olarak yerel dizinin hata verdiği durumlarda servis keşif sürecinde yapılan sorgulara cevap verilememesi ihtimali ortaya çıkmaktadır. Bu sorunun aşılabilmesi adına, hizmeti sunan IoTGW bileşenleri üzerinde, "Otomatik Kendi Kendine Kayıt" yeteneği bulunmaktadır. Belirli bir zaman diliminde, IoTGW tarafından her bir temaya ilişkin gönderilen kayıt paketine onay paketleri ile cevap sağlanamadığı durumlarda, IoTGW bileşeni bu keşif iletilerine kendisi cevap vermektedir.



Şekil 3.10. Kayıt silme süreci

Çeşitli nedenlerde IoTGW donanımı tarafından sağlanan hizmetin devre dışı kaldığı ya da hizmetin artık sunulmayacağı durumlarda, temaya ilişkin bilgilerin IoT Koordinatör yerel dizininden silinmesi gerekebilmektedir. Böyle bir durumda, hizmeti sunan Ağ Geçidi bileşeni, Şekil 3.10'da gösterildiği gibi kayıt silme sürecini başlatmaktadır. Kayıt silme işlemi, IoTGW tarafından gönderilen RST bayrağı "1" olarak işaretlenmiş bir Kayıt Paketi ile mümkün olmaktadır. Bu paketin başlık bilgisi ve örnek içeriği Şekil 3.11'de gösterilmektedir.



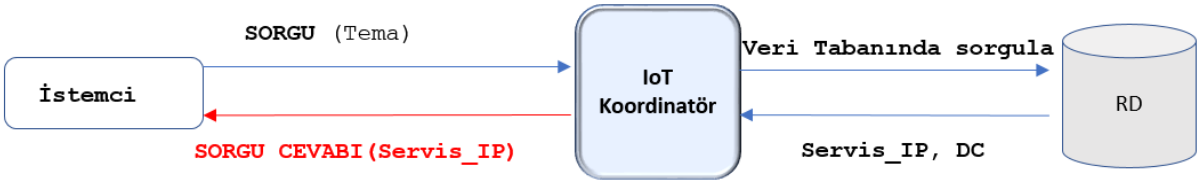
Şekil 3.11. Kayıt silme sürecinde paket başlığı

Kayıt silme özel paketini alan IoT Koordinatör, pakette belirtilen temaya ilişkin bilgileri yerel dizinden kaldırır ve talebi gönderen IoTGW bileşenine onay paketi ile geri bildirimde bulunmaktadır.

3.5.2 Servis keşif aşaması

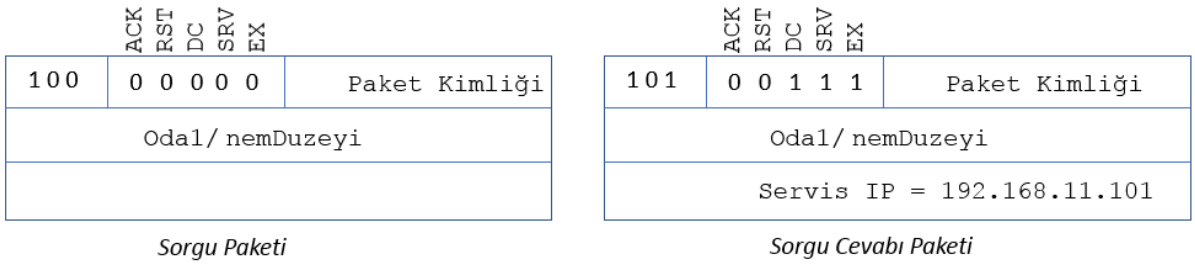
Nesnelerin İnterneti uygulamalarında, ortama ilişkin hizmetler genellikle sensörler gibi cihazlar tarafından sağlanmaktadır [84]. Her geçen gün daha fazla sayıda cihaz ağa bağlanmaktadır. Bağlanan bu cihazların keşfedilmesi IoT için büyük bir zorluk oluşturmaktadır [71,85]. Servis Keşfi, istemcilerin insan müdahalesi olmadan diğer hizmetlerin varlığını bilmesini sağlamak amacıyla tasarlanmaktadır. Sensörler tarafından sağlanan hizmetlerin keşfedilebilmesi için etkin yöntemlerin kullanılması gerekmektedir. Bu tez çalışmasında geliştirilen uygulama protokolünün yeteneklerinden biri de harici bir protokole gereksinim olmadan servis keşfi yapabilmesidir. IoT yapısında mDNS ya da DNS-SD gibi uygulama katmanı harici protokoller ile servis keşfi çalışmaları yapılmaktadır. Ayrıca CoAP protokolünün de bünyesinde servis keşfi mekanizmaları URI yapısı ile sağlanmaktadır. Ancak MQTT protokolünde, servislerin keşfedilmesine yönelik herhangi bir özellik bulunmamaktadır [86].

Hibrit Uygulama Protokolü kapsamında geliştirilen mimaride, ulaşılmak istenen hizmet bilgisine ait sorgular istemci tarafından “tema” belirtilerek “Sorgu” paketleri aracılığıyla IoT Koordinatör bileşenine gönderilmektedir. Şekil 3.12’de bu süreç gösterilmektedir.



Şekil 3.12. Servis keşif aşaması

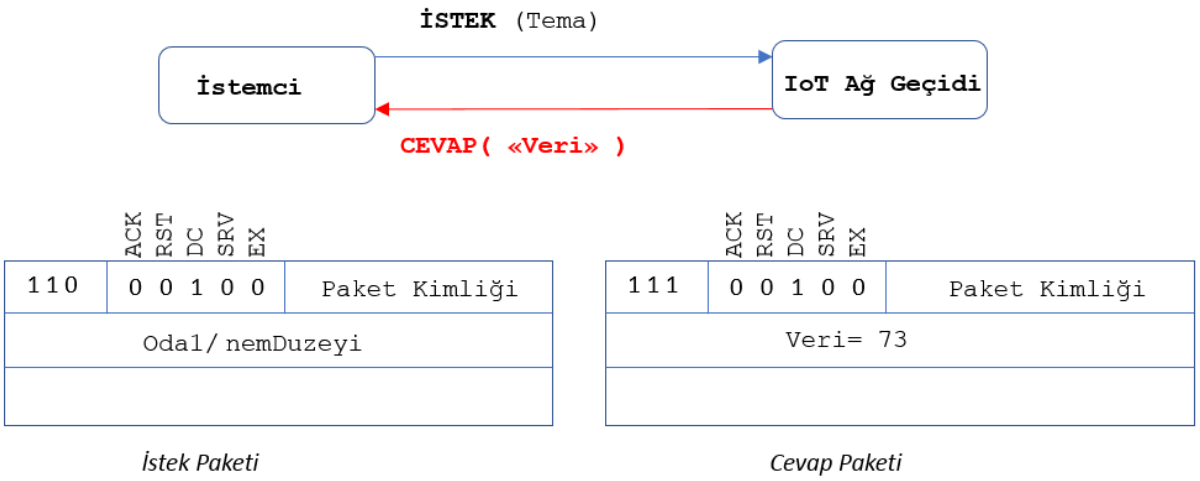
İstemcinin erişmek istediği tema bilgisine ait sorguda yer alan bilgi, IoT Koordinatör bileşenin yerel dizininde araştırılır. Bu hizmeti sağlayan IoTGW bileşenine ait IP adresi ve erişim yöntemi (DC) bilgileri veri tabanından alındıktan sonra, erişim yöntemine göre istemciye farklı bilgiler gönderilmektedir. Temaya ilişkin desteklenen iletişim yöntemi doğrudan erişimi sunuyorsa (DC=1), IoTGW bileşeninin IP adresi bilgisi istemciye gönderilmektedir. Aksi durumda, sunucu aracılığı ile iletişim modeli desteklendiğinden, IoT Koordinatör bileşeninin kendi IP adresi istemciye sorgu paketi içerisinde iletilmektedir. Sorgu paketi ve bu pakete verilen örnek Sorgu Cevabı paketinin yapısı Şekil 3.13'te gösterilmektedir.



Şekil 3.13. Servis keşif aşamasında kullanılan paketler

3.5.3 Veri transferi aşaması

Çalışma kapsamında önerilen mimarinin üçüncü aşaması, verilerin doğrudan IoTGW donanımından ya da sunucu üzerinden iletildiği aşamadır. İstemci bileşeni, servis keşif aşamasında sorguladığı tema bilgisine ilişkin kaynaklara nasıl ulaşacağı bilgisine sahip olmaktadır. Veri transferi aşamasında, elde edilen erişim bilgilerine göre, "tema" belirtilerek veri talebinde bulunulmaktadır. Desteklenen yöntem doğrudan erişim yöntemi olduğu durumda, iletişim istemci ile IoT Ağ Geçidi arasında gerçekleşmektedir. Bu iletişime ilişkin grafiksel gösterim ve paket yapısı Şekil 3.14'te gösterilmektedir.



Şekil 3.14. Doğrudan veri iletişi ve kullanılan paketler

Doğrudan erişim yönteminde, istemci tarafından gönderilen İstek paketlerine cevap olarak, IoT Ağ Geçidi tarafından Cevap paketi gönderilmektedir. Cevap paketinin içeriğinde, istemcinin öğrenmek istediği bilgiler bulunmaktadır. Şekil 3.14'te belirtilen örnekte, Oda1 konumunda yer alan Nem Düzeyi bilgisi istenmiş ve cevap olarak "73" değeri gönderildiği görüntülenmektedir.

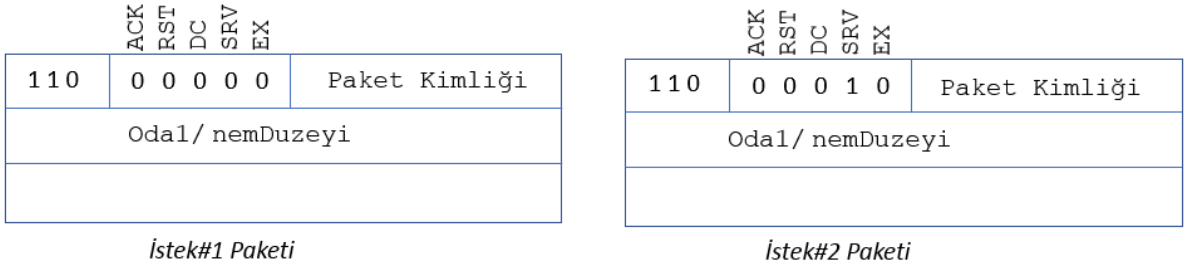
Doğrudan erişim desteği sağlanmadığı durumda, istemciye IoT Koordinatör bileşeninin kendi IP adresi cevap olarak gönderilmektedir. Bu durumda istemcinin talep ettiği temaya ilişkin bilgi koordinatör bileşen üzerinden sağlanmaktadır. Bu iletişimin grafiksel gösterimi Şekil 3.15'te belirtilmektedir.



Şekil 3.15. Sunucu tabanlı iletişim gösterimi

Bu yaklaşımda, istemcinin IoTGW bileşenine doğrudan erişimi istenmemektedir. Bu nedenle, belirlenen tema için IoTGW donanımı sadece IoT Koordinatör tarafından gönderilen istek paketlerine karşılık vermektedir. Bu durum, istemcinin istek taleplerinin IoT Koordinatör bileşenine gelmesi, ardından koordinatör tarafından bu isteğin IoT Ağ Geçidine gönderilmesi anlamına gelmektedir. Koordinatörün görevi burada aracı sunucu olarak görev almaktadır.

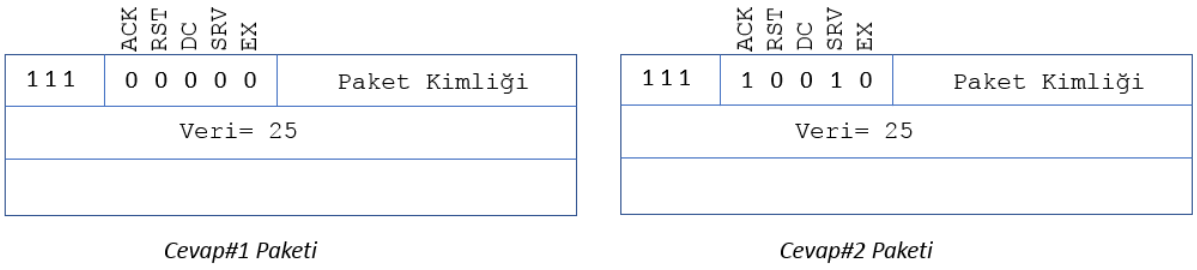
Belirli bir hizmete sınırlı erişimin sunulması istendiği durumlarda, ya da tüm erişimin analiz edilip toplanması gerektiği durumlarda kullanılması uygun olmaktadır. Ancak, verinin kaynağı ile hedefi arasında bir ek donanımın daha bulunması, gecikmelere yol açmaktadır. Bu nedenle, bu yöntemin zamana duyarlı hizmetlerde kullanılması ek gecikmelere neden olmaktadır. Bu erişim yönteminin paket başlık yapısı Şekil 3.16'da gösterilmektedir.



Şekil 3.16. Sunucu tabanlı erişimde kullanılan istek paketleri

IoT Ağ Geçidi tarafından gönderilen cevap paketi koordinatör bileşenine gelmektedir. Bu cevap bilgisi, daha sonra veri talebini yapan istemciye gönderilmesinin arından, önbellekleme süresi boyunca yerel dizinde tutulur. Bu süre içerisinde aynı tema için gönderilen tüm talepler, IoTGW bileşenine tekrar bir istek talebi olmadan doğrudan doğruya bu veritabanından verilmektedir.

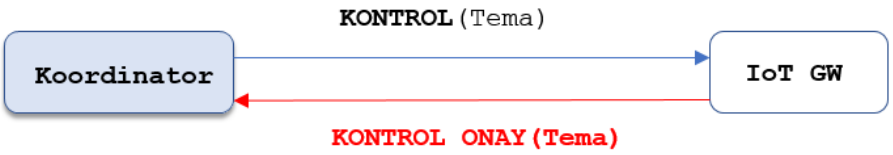
Bu iletişime ait, Ağ Geçidi bileşeninden ve sunucudan gönderilen cevaplar sırasıyla Şekil 3.17'de gösterilmektedir.



Şekil 3.17. Sunucu tabanlı iletişimde kullanılan cevap paketleri

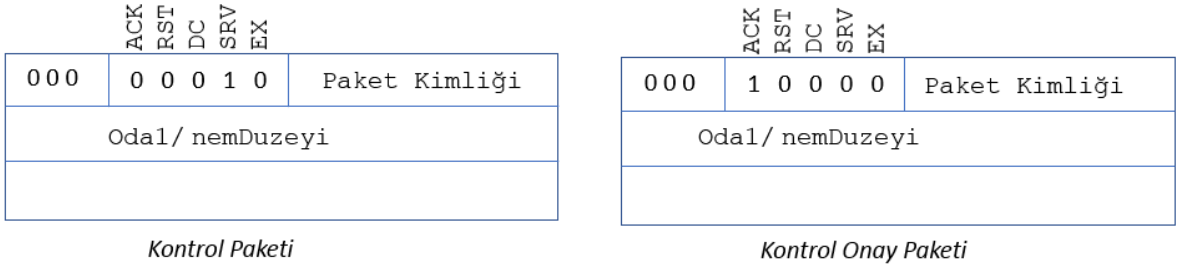
3.5.4 Diğer hIoT paketleri

Geliştirilen uygulama protokolünde ana fonksiyonları gerçekleştiren sorgulama, keşif ve kayıt paketlerinin yanı sıra, diğer yardımcı görevlerde kullanılan, Kontrol Paketi, Hata Paketi ve Sıfırlama paketi olmak üzere üç paket türü daha bulunmaktadır.



Şekil 3.18. Kontrol süreci

Kontrol Paketleri, IoT GW bileşeni üzerinde yer alan bir hizmete ait tema bilgisinin kullanımda olup olmadığını kontrol etmek için kullanılmaktadır (Şekil 3.18). IoT Koordinatör tarafından "tema" bilgisine göre yapılan kontrol ve buna IoTGW tarafından gönderilen onay mesajları ve örnek kullanımı Şekil 3.19'da gösterilmektedir.



Şekil 3.19. Kontrol sürecinde kullanılan paketler

IoT Koordinatör tarafından gönderilen kontrol paketlerine belirli bir süre içerisinde kontrol onay paketleri ile cevap verilmediği durumda, sorgulanan temaya ilişkin hizmetin geçersiz olduğu kabul edilir ve IoT Koordinatörün yerel dizininden kaldırılmaktadır. Öte yandan, IoT Ağ Geçitleri de koordinatör bileşeninin erişilebilir olup olmadığını doğrulamak için tema bilgisi boş olan kontrol paketlerini kullanmaktadır. Bu kontrol mesajına koordinatör tarafından cevap verilmediği durumda, IoTGW bileşenleri doğrudan erişim moduna geçiş yapmaktadırlar.

IoT GW cihazının hangi erişim yöntemini (DC) sunduğu, temanın kayıt sürecinde koordinatör bileşenine bildirilmektedir. Ancak ağın o anki bant genişliği kullanımı, yoğunluk ya da IoTGW üzerinde bulunan yük miktarına göre doğrudan erişim metodunun değiştirilmesi gerekebilir. Bu durumda erişim metodunun değiştirilmesi için *Sıfırlama* paketi gönderilmektedir [5].

Geliştirilen mimaride kullanılan diğer bir yardımcı paket türü de hata paketidir. İletişimde, sorgulanan temanın bulunmaması, koordinatör yerel dizinine kayıt yapılamaması gibi çeşitli sebeplerle yaşanan hata durumlarında, hata kodu ile birlikte gönderilen paketlerdir.

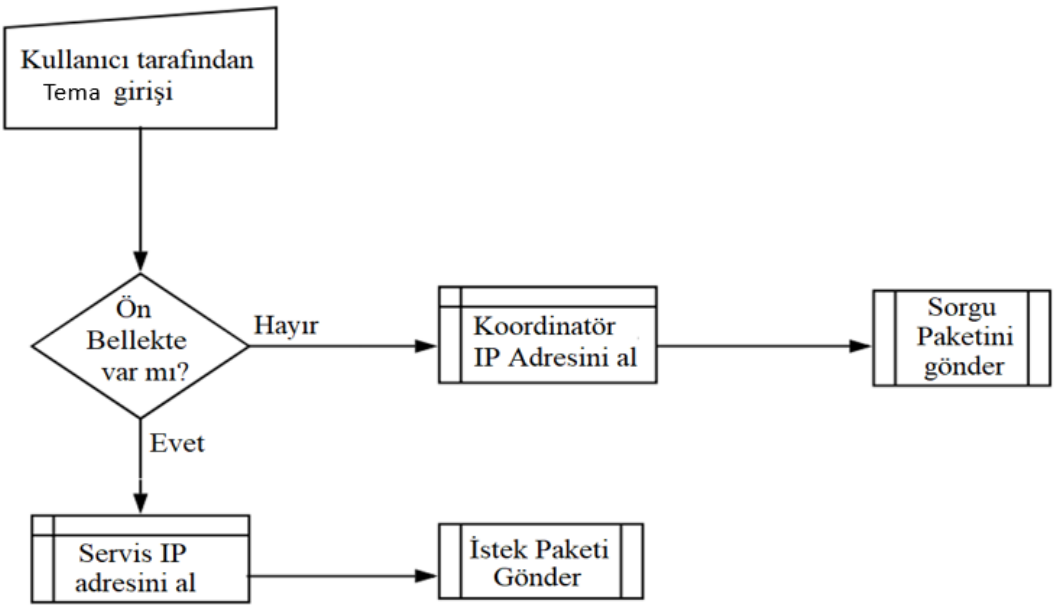
3.6 Mimaride Kullanılan Bileşenler

IoT uygulama protokollerinin çalışmasında her donanımın ya da protokolü kullanan her cihazın rolü aynı değildir. MQTT yapısında, Abone, Yayıncı ve Aracı Sunucu rolleri; CoAP yapısında İstemci ve Sunucu rolleri, AMQP mimarisinde de Yayıncı, Sunucu ve Abone bulunmaktadır. Bu tez çalışması kapsamında geliştirilen mimaride ise, IoT Koordinatörü, IoT Ağ Geçidi ve İstemciden oluşan üç rol bulunmaktadır. Çalışma kapsamında bu rollerdeki cihazlar "bileşenler" olarak adlandırılmıştır.

3.6.1 İstemci Bileşeni

IoT uygulamalarının önemli kullanım senaryolarından ikisi M2M ve M2P bağlantı türleridir [39]. Bu her iki yaklaşımda da insan ya da makine, veri isteyen bir taraf bulunmaktadır. Bu çalışma kapsamında "İstemci" kavramı her iki bağlantı türü içinde kullanılabilen, veri talebinde bulunan uygulamayı, yazılımı ya da donanımı ifade etmektedir. IoT uygulama protokolleri açısında ele alındığında ise, MQTT ya da AMQP mimarilerindeki "abone" rolüne karşılık gelmektedir. Ancak bu iki protokoldeki abone ile bu çalışmadaki "istemci" arasında temel bir farklılık bulunmaktadır. Adı geçen iki protokolde de, veriyi başlatan taraf yayıncı bileşeni iken, bu tez kapsamında geliştirilen protokolde, veri trafiğini başlatan taraf, CoAP protokolündekine benzer olarak "*istemci*"dir. Geliştirilen modelde, -İstek/Cevap iletişim türüne uygun olarak- veriye ihtiyaç duyulduğu an veri talebi istemci tarafından başlatılmaktadır.

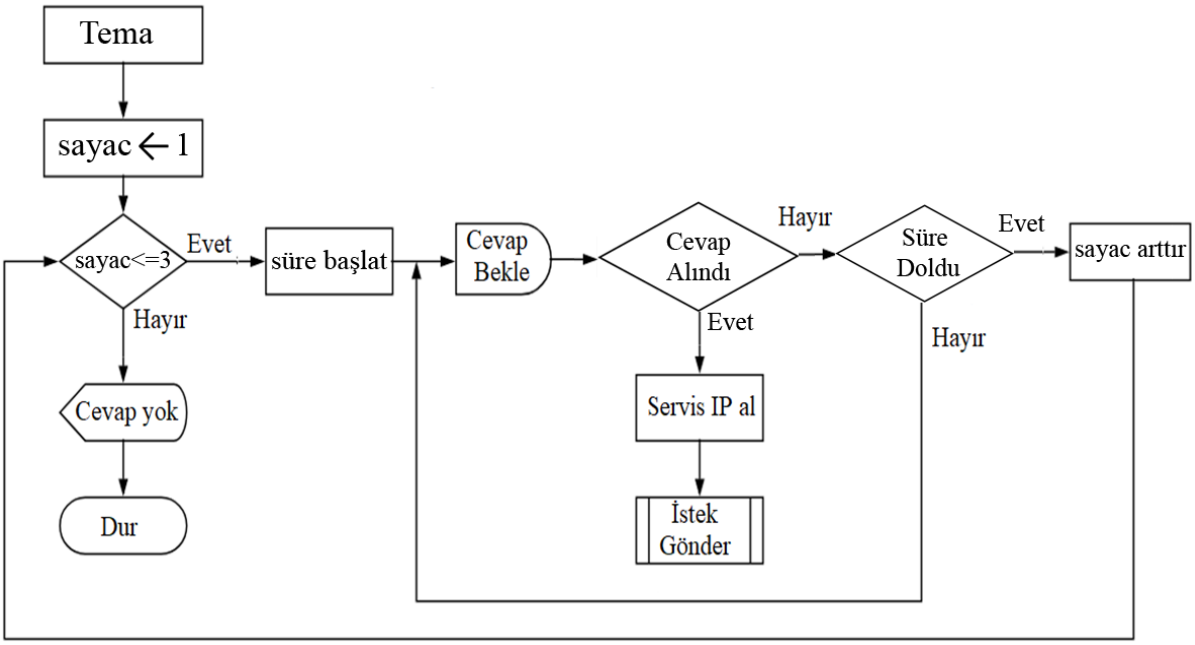
İstemci "tema" parametresini kullanarak erişmek istediği hizmete ait sorgulama yapmaktadır. IoT Koordinatörü tarafından cevap olarak gönderilen Sorgu Cevabı paketinde, ulaşılmak istenen hizmetin erişim parametreleri yer almaktadır. İstemci gönderilen bu bilgileri sorguladığı "tema" ile eşleştirip önbelleğe atar. Bu ön bellekte yer alan bilgi sayesinde, sonradan aynı hizmete ait veriye ihtiyaç duyulduğuna, erişim bilgileri zaten bilindiğinden, tekrar bir keşif sorgulaması yapılmadan doğrudan veri transferi safhasına geçilmektedir. Şekil 3.20'de istemci bileşeni için akış şeması tasarımı gösterilmektedir [5].



Şekil 3.20. İstemci bileşeni akış diyagramı

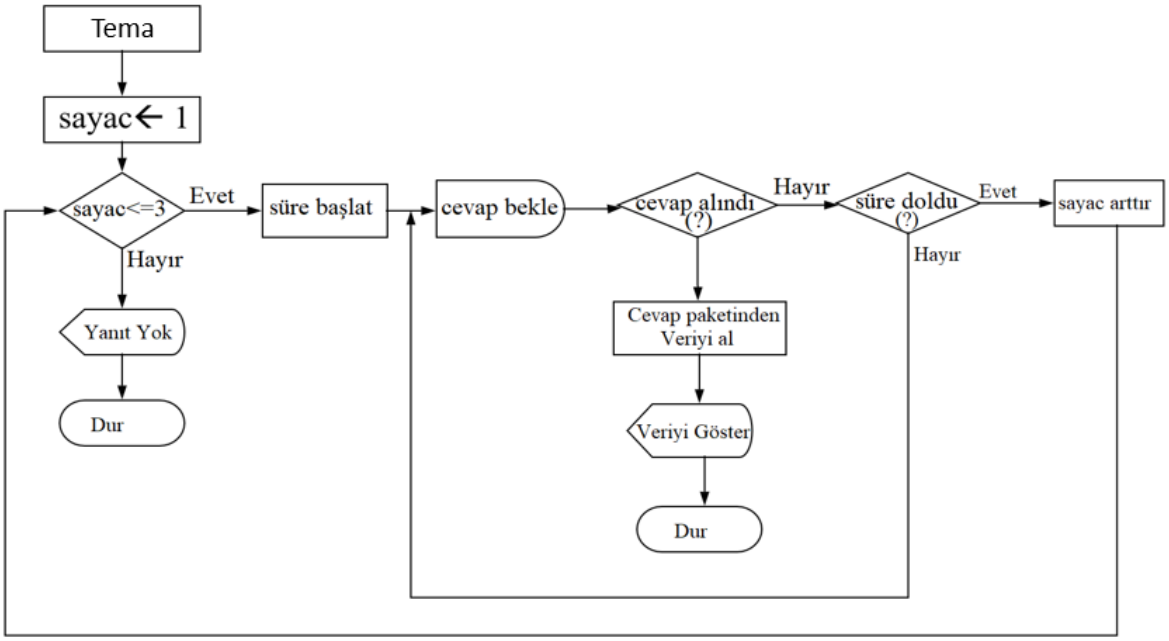
Başlangıç durumunda istemci için genel olarak iki durum bulunmaktadır. İhtiyaç duyulan "tema" bilgisine ilişkin erişim bilgileri ön bellekte yer alıyorsa, tekrar sorgu yapılmasına gerek kalmayacağından, doğrudan veri talebinin yapılacağı "istek durumuna" geçmektedir. Aksi durumda, hizmetin sağlanacağı aşamaya geçmek için "sorgu durumuna" geçmektedir. Bu aşamada, IoT Koordinatör bileşenine, aranılan servise nasıl erişeceğini soran "sorgu paketi" gönderilmektedir. Daha önceden temaya ilişkin bir sorgulama yapılmışsa, bu sorgulamaya ait kayıtlar istemcinin ön belleğinde yer almaktadır. Bu durumda, hizmetin sağlandığı cihazın, daha doğru bir ifade ile belirli bir *temanın* bulunduğu cihazın IP adresi, tekrar sorgulama yapılmadan gecikmesiz olarak doğrudan alınmaktadır. Artık servisi sunan cihazın (IoTGW veya IoT Koordinatör) erişim bilgileri bilindiği için, istek paketi gönderilebilir durumuna geçilmektedir.

Sorgu durumu (akış diyagramında "sorgu paketi gönder" olarak belirtilmektedir), erişim bilgilerinin IoT Koordinatörden istenmesi aşamasıdır ve hibrit uygulama protokolünün "Servis Keşfi" aşamasına denk gelmektedir. Bu aşamada istemcinin sergilediği davranışlar yine Şekil 3.21'de akış diyagramı olarak ifade edilmektedir.



Şekil 3.21. İstemci bileşeni, sorgu paketi gönderim akış diyagramı

Şekil 3.21'de yer alan şemada belirtildiği gibi, tema bilgisi içeren sorgulama paketinin gönderiminin ardından belirli bir süre sorguya cevabın gelmesi beklenmektedir. Bu süre zarfında IoT koordinatörden herhangi bir bildirim alınmazsa, işlemler iki kez daha (toplamda üç kez) yeniden uygulanmaktadır. Maximum tekrar sayısının ardından hâlen herhangi bir sorgu cevabı gelmediği takdirde, istemci sorgunun cevaplanmadığına dair bilgilendirilmekte ve süreç sona ermektedir. Koordinatör tarafından bir sorgu cevabı ulaştığında, istemci cevap içerisinde sunulan Servis IP bilgisini (hizmetin sağlandığı bileşenin erişim bilgisi) almakta ve istek gönderim aşamasına geçilmektedir.

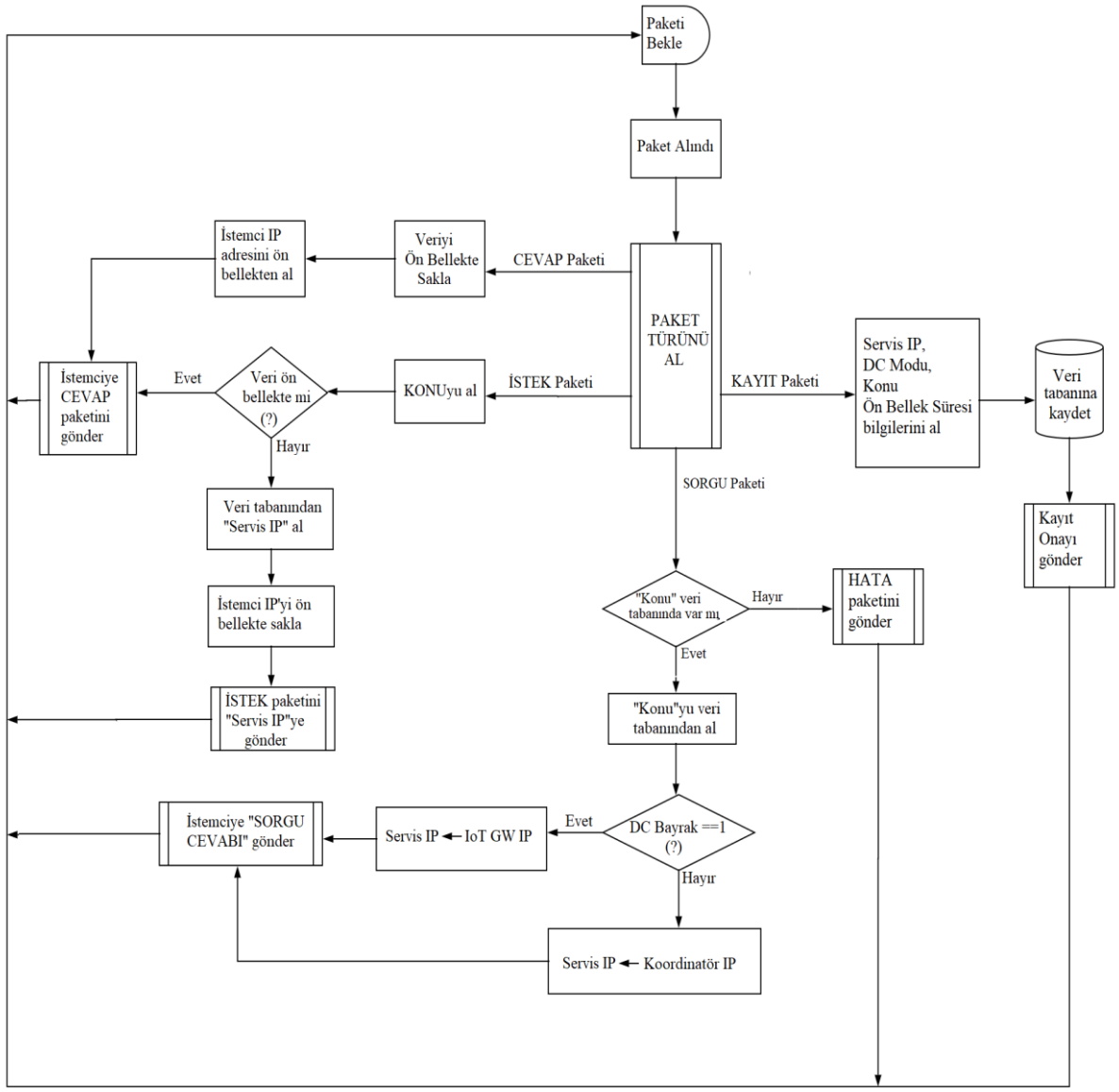


Şekil 3.22. İstemci bileşeni veri iletim akış diyagramı

Şekil 3.22'de istemcinin İstek durumundaki davranışları akış diyagramı olarak ifade edilmektedir. Bu aşama, hibrit IoT uygulama protokolünün aşamalarından olan "veri transferi" aşamasına denk gelmektedir. Bu aşamada, erişim bilgileri alınmış IoT bileşenine gönderilmek üzere, tema ile ilişkilendirilmiş istek paketleri gönderilir. Daha önce sorgu durumunda olduğu gibi, üç kez bu istek paketine cevap gelmediği durumda, işlem sonlandırılmaktadır.

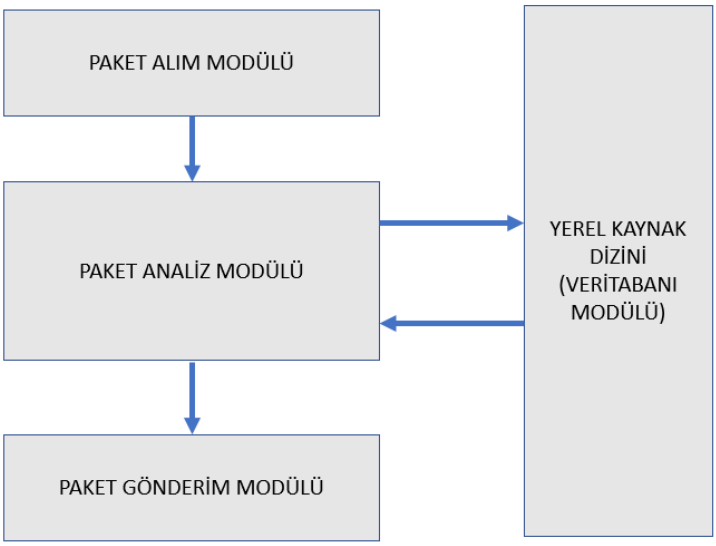
3.6.2 IoT koordinatör bileşeni

Geliştirilen uygulama protokolünün mimarisindeki en önemli bileşen, IoT Koordinatör bileşenindedir. IoT koordinatörün görevi, servis keşif sürecinde, istemcilerin ihtiyaç duyacağı bilgilerin barındırılması, IoT Ağ Geçidi üzerinden sunulan tema ile ilişkilendirilmiş kaynakların yerel kaynak dizininde kayıt ettirilmesi ve iletişimdeki koordinasyonun sağlanmasıdır. Ayrıca, IoT ekosisteminde yer alan hizmetlerin erişilebilirliğinin denetlenmesi ve doğrudan ya da sunucu aracılığı ile gerçekleştirilen veri transfer yönteminin belirlenmesidir.



Şekil 3.23. Koordinatör bileşeni akış diyagramı

Nesnelerin İnternetinde çözüm aranan sorunlardan biri de, servis keşfi sürecinde kullanılacak olan kaynak dizin ihtiyacıdır [74]. Geliştirilen mimaride IoT Koordinatör bileşeni üzerinde yer alan, yerel dizin kaynağı, bu sorunu yerel olarak çözmek amacıyla kullanılabilmektedir. IoT koordinatör bileşenin görevleri ve çalışma prensibi Şekil 3.23'te akış diyagramı ile ifade edilmektedir [5].



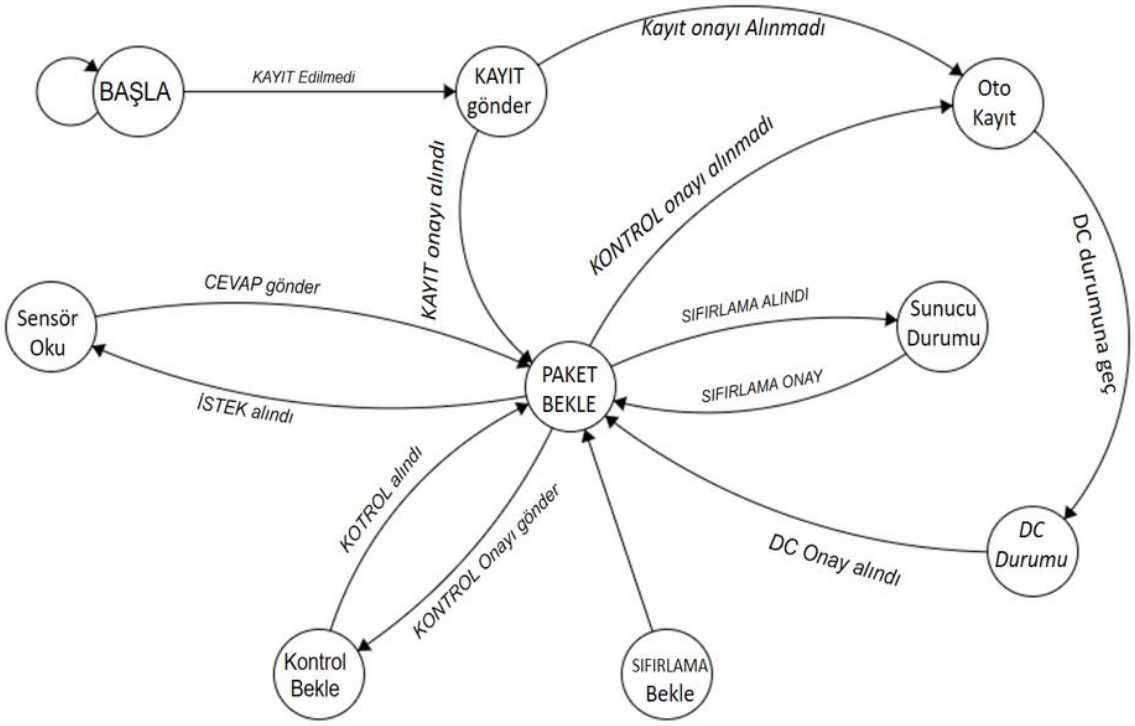
Şekil 3.24. IoT Koordinatörü akış diyagramında modüler yapı

IoT Koordinatör bileşeninin görevlerini tanımlamak amacıyla, birbiri ile ilişkili dört fonksiyonel modül tanımlanmaktadır (Şekil 3.24). Paket Alımı Modülü, gelen Hibrit IoT paketlerinin alımından sorumludur. Paket alındıktan sonra, bu paketin hangi tür paket olduğunun analizi ve ona göre yapılması gereken işlemleri yürütmek amacıyla Paket Analiz Modülü bulunmaktadır. Bu modül, IoTGW tarafından sunulan hizmetlerin veri tabanında tutulduğu yerel kaynak dizinini kontrol etmektedir. Veri tabanında tema ile ilişkilendirilmiş hizmetler ve bunlara erişim bilgilerinin kayıt ettirilmesi ve veri tabanından ilgili servise ilişkin alınması görevini yapmaktadır. Son olarak, veri tabanından elde edilen servis bilgilerinin iletilmesini sağlamak amacıyla paket gönderici modülüne bu bilgileri göndermektedir.

3.6.3 IoT Ağ geçidi bileşeni

IoT ekosisteminde yaygın kullanım alanı bulan sensörler gibi donanımlar, genellikle sınırlı bilgi işlemsel kapasitesi olan donanımlardır[17]. Bu nedenle bu tür donanımlardan üretilen verilerin anlamlandırılması amacıyla, bu cihazlar ile doğrudan ya da dolaylı olarak etkileşim kurabilen, dijital ya da analog sinyal gönderip alabilen ve aldığı verileri ağ ortamına sunabilen IoT Ağ Geçidi olarak isimlendirilen yetenekli donanımlara gereksinim duyulmaktadır [5]. Bu bileşen IoT ekosisteminde yer alan önemli donanımlardan biridir [87]. IoT Ağ Geçidi, fiziksel katmanda yer alan düşük kapasiteli donanımlar ile ağ ya da uygulama katmanı arasında köprü görevi görmektedir.

Bu mimarideki IoT Ağ Geçidi bileşeninin görevi, nesnelerden gelen verilerin alınıp, internet ya da ağ ortamında kullanılabilecek forma dönüştürülmesidir. IoTGW, istemcilerin veri taleplerine doğrudan cevap verebildiği gibi, sunucu (IoT Koordinatör) aracılığı ile de cevap verebilmektedir. IoTGW aracılığıyla sunulan ve tema ile ilişkilendirilen her bir hizmet için erişim yöntemi, Servis Kayıt Aşamasında gönderilen paketlerle mümkün olmaktadır. Bu işlemin koordinatör tarafından onaylanmasının ardından, ağ geçidi bileşeni istemcilerden gelen veri transferine cevap verebilmektedir. Şekil 3.25'te IoT Ağ Geçidi için sonlu durum akış diyagramı gösterilmiştir.

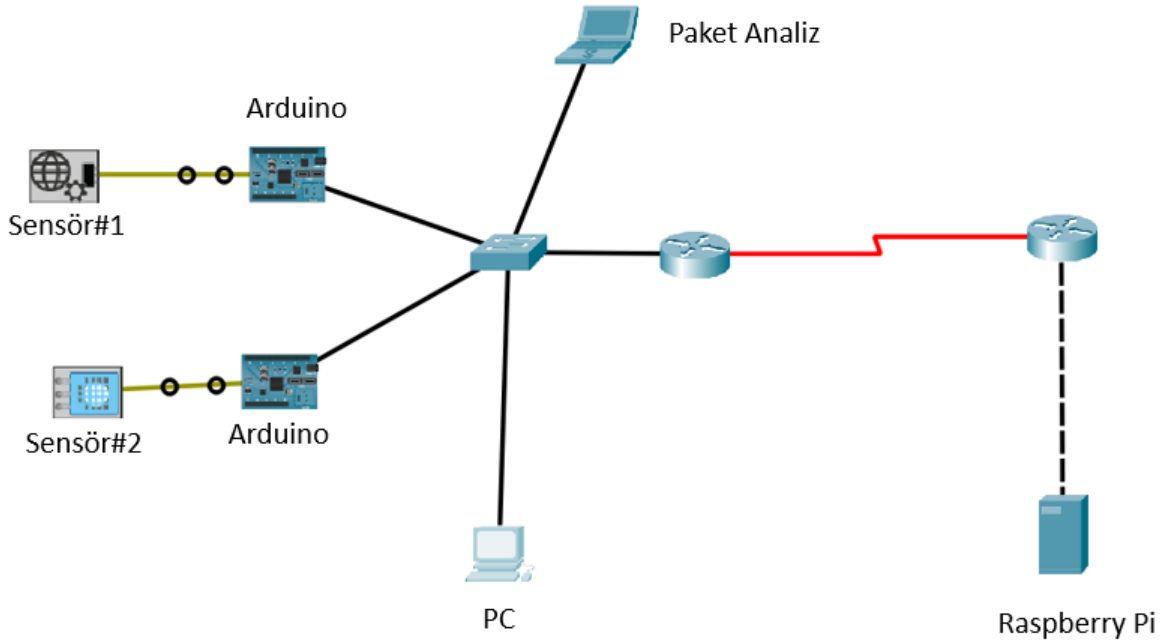


Şekil 3.25. IoT Ağ Geçidi bileşeni, sonlu durum akış diyagramı

IoT Ağ Geçidi bileşeni, IoT Koordinatör ile iletişimde olup olmadığını doğrulamak için Kontrol paketlerini kullanmaktadır. Koordinatörün devre dışı kalması ya da herhangi bir nedenle erişilemez olması durumunda, istemci tarafından gönderilen İstek ve Sorgu paketlerine cevap verebilmek ve bağlanabilirliği sağlamak amacıyla IoT Ağ Geçidi kendini doğrudan erişim moduna almaktadır [5]. IoTGW bileşeni, koordinatörden gelen Sıfırlama Paketi ile bu doğrudan çalışma modunu kaybetmektedir.

3.7. Performans Değerlendirmeleri

Bu çalışma kapsamında geliştirilen protokolün performansının ve işlevselliğinin değerlendirilebilmesi için, diğer ağ trafiklerinden yalıtılmış bir ortamda Şekil 3.26'da gösterilen kapalı bir ağ topolojisi hazırlanmıştır.



Şekil 3.26. Deneysel ortam topolojisi

Geliştirilen protokolün gerçek donanımlarla performansının değerlendirilmesi için tasarlanan bu topolojide, IoT uygulamalarında yaygın kullanılan MQTT protokolü ile Hibrit Uygulama Protokolü, işlevsellik, bant genişliği kullanımı, gecikme gibi çeşitli yönlerden karşılaştırılmaktadır. Denetleyici olarak, Arduino UNO donanımları, sunucu olarak Raspberry PI-3 donanımı kullanılmaktadır. Veri talebi PC üzerinden gönderilen paketlerle sağlanmaktadır. Ayrıca, üretilen tüm trafiğin yakalanıp analiz edilmesi için WireShark yazılımı yüklenmiş bir dizüstü bilgisayar kullanılmaktadır. MQTT protokolünde, üzerinde açık kaynaklı Mosquitto [88] yüklü Raspberry-PI donanımı aracı sunucu (broker) olarak kullanılmaktadır. Arduino cihazlar, MQTT yayıncı olarak, PC ise MQTT abone olarak çalışmaktadır.

Geliştirilen protokolde, Raspberry-Pi cihazı IoT koordinatör olarak, PC istemci bileşeni olarak, Arduino denetleyiciler ise IoT Ağ Geçidi olarak görev almaktadır. Her iki senaryoda da, PC ile

Arduino cihazlar IEEE 802.3 Ethernet bağlantısı sağlayan standart Ethernet anahtar (switch) ile birbirine bağlanmaktadır. Ayrıca, sekme sayısının etkisini analiz etmek ve çeşitli bant genişliklerindeki davranışlarını incelemek amacıyla iki adet yönlendirici bağlanmaktadır. Yönlendiriciler arasındaki bant genişlikleri değiştirilerek etkiler analiz edilmektedir.

Şekil 3.26'da gösterilen fiziksel topolojide, bant genişliği kullanımı, gecikme ve taşınan yük miktarına göre davranışları karşılaştırılmıştır.

3.7.1 Farklı bant genişliklerinde yaşanan gecikmeler

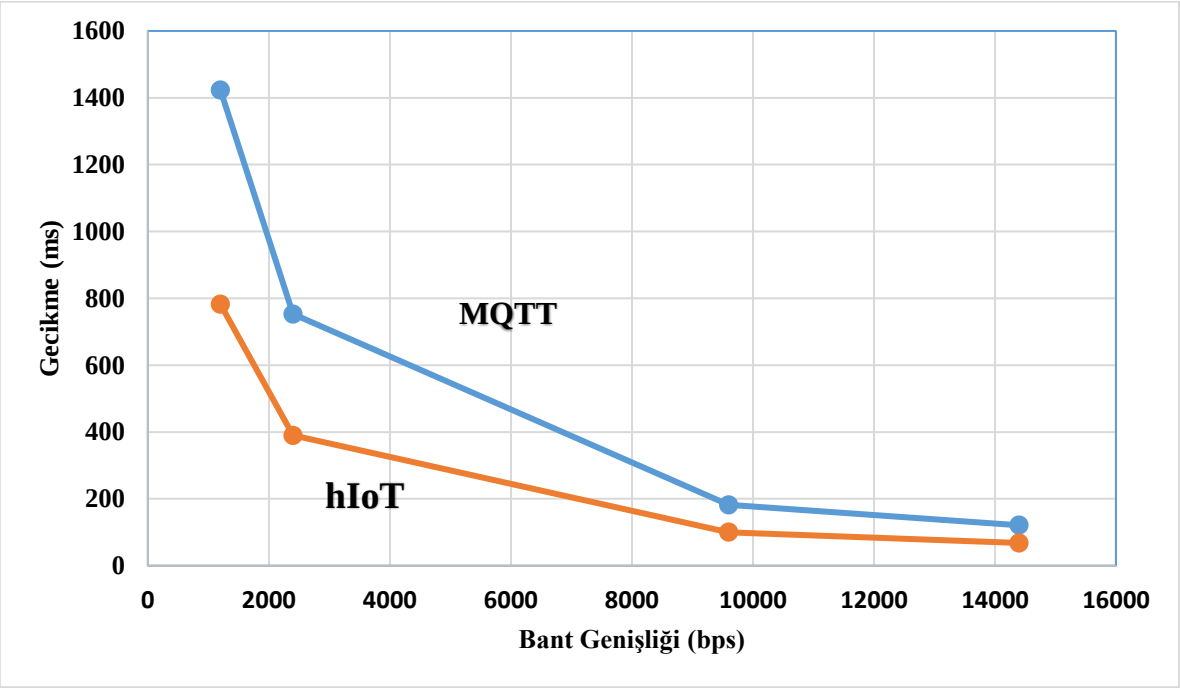
Şekil 3.26'da yer alan topolojide MQTT ve hIoT protokollerinin farklı bant genişliklerinde ne kadar gecikmeye neden oldukları değerlendirilmiştir. MQTT protokolünün yapısında üç farklı QoS düzeyi bulunduğundan, bu deneysel çalışmada en hızlı düzey olan QoS0 düzeyi kullanılmaktadır. İlk değerlendirme, her iki protokolde benzer işlevleri bulunan MQTT protokolünün "Connect" aşaması ile Hibrit Uygulama Protokolünün "Kayıt" aşamaları arasındaki farklı bant genişliklere dayalı gecikme üzerinden yapılmaktadır. Gönderilen paketlerin bir nüshası WireShark paket analiz programı ile yakalanmıştır. Gerçek ortamda yapılan her işlem 30 kez tekrarlanmakta ve böylece ortalama gecikme değerleri elde edilmektedir. Elde edilen değerler Çizelge 3.2'de gösterilmektedir.

Çizelge 3.2. Bant genişliğine bağlı gecikme değerleri

Bant Genişliği Kullanımı (bps)	**Uçtan Uca Gecikme Süreleri (ms)**	
	MQTT Protokolü (Bağlantı Aşaması)	**Hibrit IoT Protokolü (Kayıt Aşaması)**
1200	1423	782
2400	753	390
9600	182	100
14400	121	68
64000	26	41
128000	16	11

TCP tabanlı çalışan bir protokol olan MQTT'de Connect aşamasından önce, 3 yönlü el sıkışma adımlarının yapılması gerekmektedir. Her TCP iletişiminde gerçekleşen bu adımların ardından,

yayıncı ile aracı sunucu arasında Bağlantı (Connect) ve Bağlantı Onay (Connect-Ack) aşamalarına geçilmiştir. Bu tabloda, TCP üç yönlü el sıkışma süresince yaşanan ek gecikmeler dikkate alınmamış, doğrudan MQTT iletişiminin gecikmeleri dikkate alınmıştır.



Şekil 3.27. Bant genişliğine göre gecikme değerleri grafiği

Elde edilen sonuçlar Şekil 3.27'de verilen grafikte gösterilmektedir. Buna göre, kullanılan bant genişliğinin düşük olduğu durumlarda, Hibrit Uygulama Protokolü daha küçük gecikmeye göstermektedir. Bu durum, kısıtlı kaynaklara sahip donanımlar için geliştirilen protokolün kullanılabileceğini göstermektedir. Bant genişliğinin yüksek olduğu durumlarda, geliştirilen protokol ile MQTT protokolünün gecikme değerleri arasındaki farkı kapanmaktadır.

3.7.2 Taşınan yük miktarlarında yaşanan gecikmeler

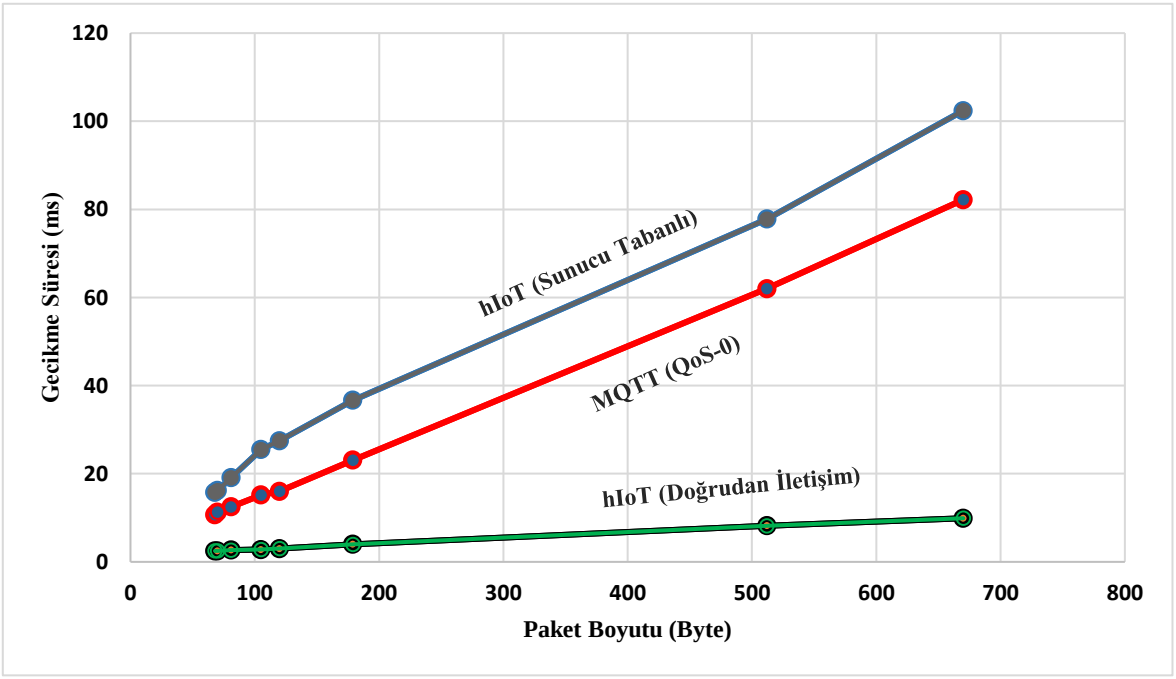
Nesnelerin İnternetinde kullanılan uygulamalarda farklı boyutlarda veriler taşınabilmektedir. Taşınan verinin boyutuna göre, bazı uygulama protokollerinin diğerlerine göre daha yüksek başarım gösterdiği görülmektedir [20]. Geliştirilen protokolün farklı yük miktarının taşınmasında davranışlarının değerlendirilmesi amacıyla, Şekil 3.26'da belirtilen topolojide, yük miktarlarına göre gecikme süreleri değerlendirilmektedir. hIoT protokolünün mimarisinde, hem doğrudan erişim hem de sunucu aracılığı ile erişim yöntemleri bulunmaktadır. Bu nedenle,

istemci tarafından gönderilen farklı boyutlardaki veriler için her iki yönteme ait süreler ayrı ayrı değerlendirilmiştir. İstemci ile IoT Ağ Geçidi arasında gerçekleşen bu iletişim, geliştirilen protokoldeki Veri Transferi aşamasında gerçekleşmektedir. MQTT yapısında ise verinin yayıncıdan çıkıp aboneye ulaşması sürecinde yaşanan gecikmeler dikkate alınmaktadır. Karşılaştırmaya yönelik elde edilen ortalama değerler Çizelge 3.3'te gösterilmektedir.

Çizelge 3.3. Taşınan yük miktarlarına bağlı gecikme değerleri

Taşınan Yük (Bayt)	**Gecikme Süreleri (ms)**		
	MQTT (Yayın - Abone)	**hIoT Doğrudan Erişim**	**hIoT Sunucu Tabanlı Erişim**
68	10,7	2,5	15,8
70	11,3	2,5	16,2
81	12,5	2,7	19,1
105	15,2	2,8	25,5
120	16	3	27,5
179	23,1	4	36,7
512	62	8,2	77,8
670	82,2	9,9	102,4

Deneysel çalışmada elde edilen ortalama gecikme sürelerinin taşınan veri boyutuna göre karşılaştırmaları Şekil 3.28'deki grafikte gösterilmektedir.



Şekil 3.28. Paket boyutuna göre gecikme değerleri grafiği

Grafikte, hIoT protokolünün her iki yöntemi MQTT protokolü ile karşılaştırılmaktadır. Geliştirilen protokolün doğrudan erişim yönteminde, arada herhangi bir ek cihaz bulunmadığından gecikme süreleri taşınan her veri miktarı için daha az olduğu görülmektedir. Ancak, sunucu tabanlı gerçekleşen veri iletişiminde, IoT Koordinatör bileşeninin bulunması nedeniyle ek gecikmeler yaşanmakta ve MQTT protokolüne göre daha yüksek gecikmeler göstermektedir. IoT ekosisteminde, veriye erişim süresinin kritik olduğu durumlarda doğrudan erişim yönteminin tercih edilmesinin daha uygun olacağı görülmektedir. Belirli oranda gecikmenin kabul edilebilir olduğu uygulamalarda ve IoT Ağ Geçidine doğrudan erişimin istenmediği ya da erişim denetiminin uygulanması istenen durumlarda, sunucu tabanlı iletişimin tercih edilmesi önerilmektedir.

3.7.3 Servis keşfi değerlendirmesi

Bu değerlendirmede, hibrit uygulama protokolü tarafından sağlanan keşif mekanizmasının bant genişliğine bağlı olarak gecikme süreleri değerlendirilmiştir. MQTT yapısında herhangi bir keşif mekanizması olmadığından, bu protokolle karşılaştırma yapılmamıştır. Bu deneysel çalışmada, istemci tarafından IoT koordinatör bileşenine “Sorgu” paketi gönderilmiş ve koordinatör tarafından da “Sorgu Cevabı” paketi alınmıştır. IoT Koordinatör bileşeninde yerel kaynak dizini, SQLite ile sağlanmaktadır. Değerlendirme yapılırken gönderilen paket ile alınan paket arasında geçen ortalama süre dikkate alınmaktadır. Çeşitli bant genişlikleri için elde edilen ortalama süreler Çizelge 3.4’te gösterilmektedir.

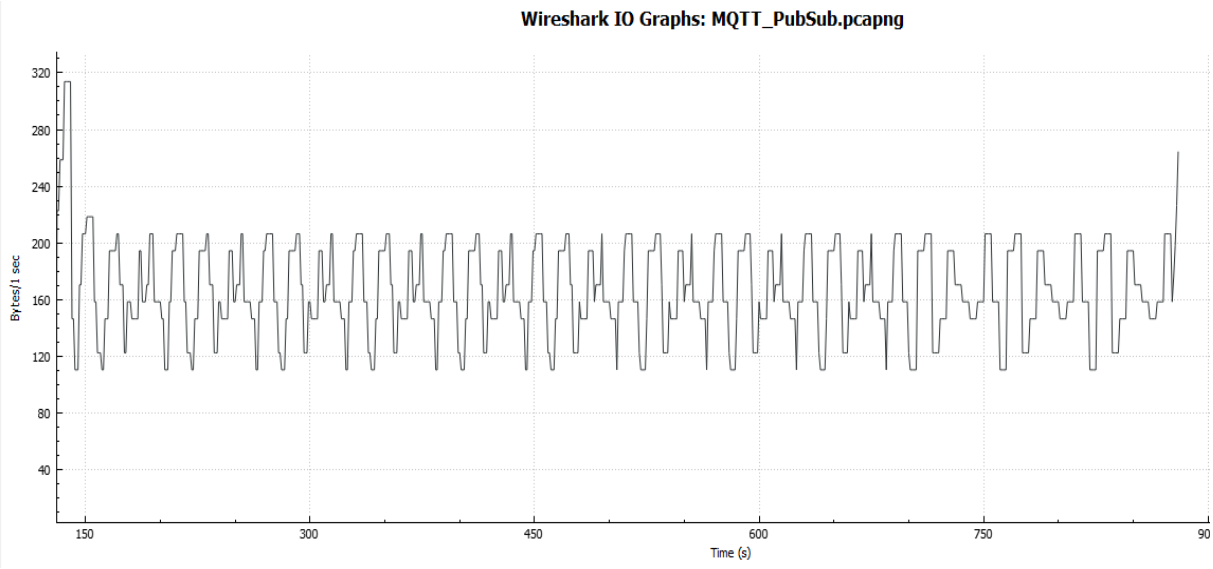
Çizelge 3.4. Bant genişliğine bağlı ortalama keşif süreleri

Bant Genişliği (bps)	**Ortalama Keşif Süresi (ms)**
1200	890
2400	384
9600	99
14400	67
64000	18
125000	15

Tablodan görüleceği üzere ortalama keşif süresi, bant genişliği ile ters orantılıdır. Yüksek bant genişliklerinde bu süre çok kısa 15 ms düzeylerinde iken, bant genişliği düştüğünde süre uzamaktadır.

3.7.4 Bant genişliği kullanımı değerlendirmesi

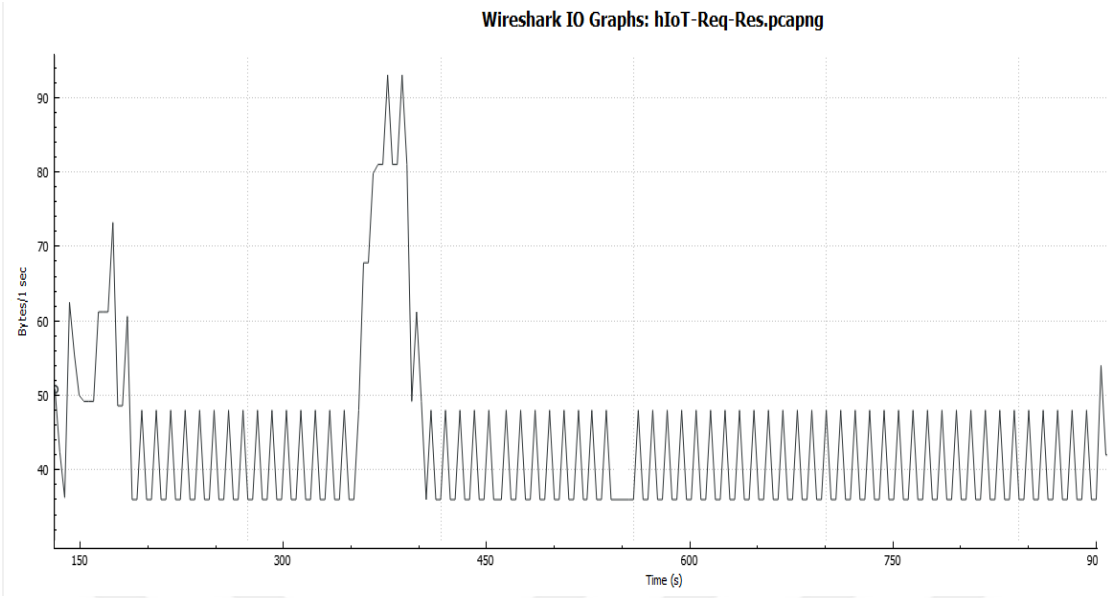
Bu değerlendirme aşamasında, istemciden periyodik olarak sabit uzunlukta (68 byte) paketler gönderilmiş ve protokollerin yeniden çalışması 900 saniye boyunca modellenmiştir. Gönderilen ve alınan tüm paketler WireShark programı ile analiz edilmiştir. Şekil 3.29'da MQTT protokolünün 900 saniye içerisindeki trafiklerinin WireShark grafiği analizi gösterilmektedir.



Şekil 3.29. MQTT trafiği için WireShark bant genişliği grafiği

MQTT protokolünün Bağlantı (Connect) aşamasındaki ek işlevler nedeniyle modellemenin ilk saniyelerinde yüksek bant genişliği tüketimi yaptığı, daha sonraki aşamalarda ise ortalama 160 Byte/sn bant genişliği tükettiği görülmektedir.

Aynı yöntemle Hibrit Uygulama Protokolünün bant genişliği tüketimi Şekil 3.30'da gösterilmektedir.

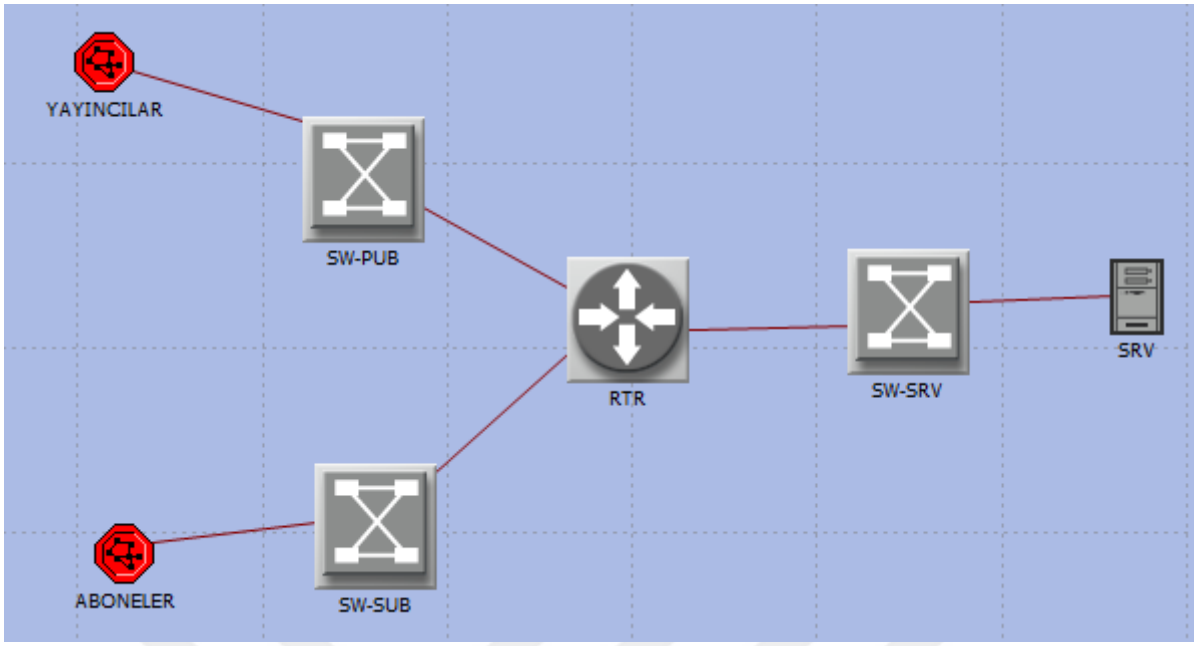


Şekil 3.30. hIoT Protokolü Wireshark bant genişliği tüketim grafiği

Grafikten görüleceği üzere, geliştirilen uygulama protokolün ortalama 45 Byte/saniye bant genişliği tüketmektedir. Modellemenin ilk sürelerinde, Kayıt paketleri nedeniyle nispeten daha yüksek bant genişliği harcadığı gözlemlenmiştir.

3.8. Benzetim Modellemesi

Bir ağın güvenilirliğini değerlendirmek için analitik metot, test metodu ve benzetim metodu kullanılmaktadır. Bu çalışmada, OPNET benzetim yazılımı kullanılarak geliştirilen uygulama protokolünün değerlendirilmesi ve MQTT protokolü ile başarım karşılaştırılması amacıyla Şekil 3.31'de gösterilen fiziksel topoloji tasarlanmıştır.



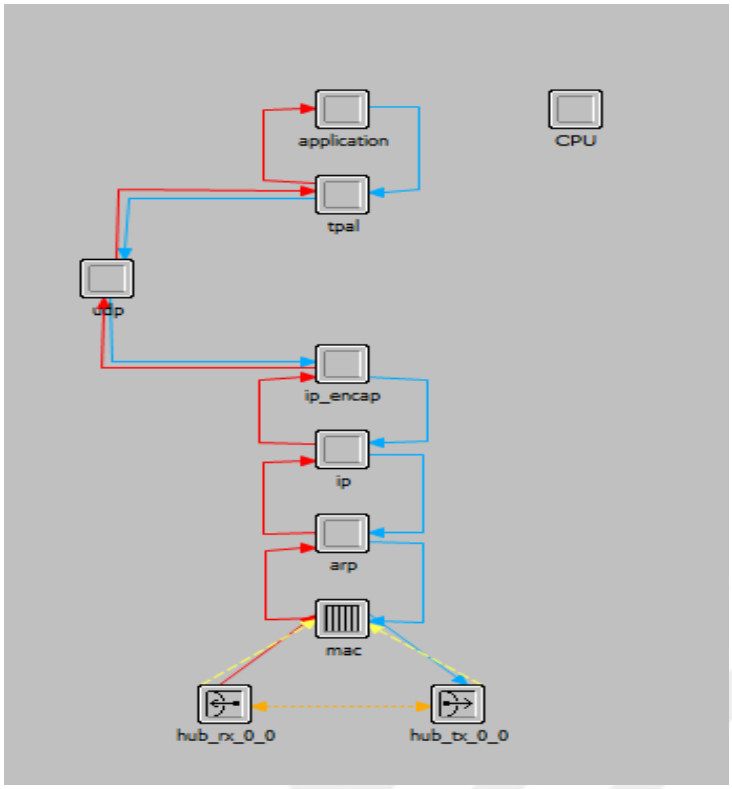
Şekil 3.31. OPNET benzetim topolojisi

MQTT protokolünün çalışma mekanizmasına uygun olarak geliştirilen bu topolojide kampüs ağlarında farklı konumlarda bulunan sensörler ve bu sensörlerden bilgi alması beklenen kullanıcıların bulunduğu trafik modellemesi yapılmıştır.

Küçük bir kampüs ağını yansıtmak amacıyla benzetim yazılımında kullanılan 3 farklı anahtar bir yönlendirici aracılığıyla bağlanmaktadır. Kullanılan kablolu ağ yapısında, düşük bant genişliklerini yansıtmak amacıyla sensör ağlarında 10Mbps Ethernetv2 genişliği kullanılmaktadır. Topolojide *SW_SUB* anahtarına bağlı olan cihazlar MQTT aboneleri, *SW_PUB* anahtarına bağlı olan cihazlar ise MQTT yayıncıları temsil etmektedir. Anahtarlara bağlı farklı sayıda sensörler kullanılarak, düğüm sayısına göre, bant genişliği kullanımına göre ve uçtan uca gecikmeye göre performans karşılaştırmaları yapılmaktadır.

3.8.1 Düğüm modeli

Deneysel olarak tasarlanan bu yapıda IoT ağ geçitlerinin görevi önerilen hIoT protokolü trafiklerinin yakalanmasıdır. Bu amaçla OPNET benzetim uygulamasında IoT Ağ geçidi *Node Model* olarak Şekil 3.32'deki gibi özelleştirilmiştir.



Şekil 3.32. Geliştirilen protokoldeki uyarlanan düğüm modeli

Ethernet mimarisinde uyarlanan bu düğüm modellemesinde sadece UDP paketlerinin alınması amaçlanmış, TCP protokolü IoT ağ geçidi cihazlarında devre dışı bırakılmıştır. Performans karşılaştırması amacıyla kullanılan MQTT protokolüne ait bilgiler, Wireshark yazılımı ile yakalanan paketlerin değerlerine göre belirlenmiştir. Buna göre MQTT QoS-0 düzeyinde, *Bağlantı*, *Yayın*, *Abonelik* ve *Onay* mesajlarının başlık uzunluklarına ve taşınan yük boyutlarına göre trafik modellemesi yapılmış ve Çizelge 3.5'te bu değerler belirtilmektedir.

Çizelge 3.5. MQTT aşamalarında taşınan yük miktarları

MQTT Aşamaları	**Yük Miktarı**
Connect	39 Byte
Connect Ack	4 Byte
Subscribe	18 Byte
Subscribe Ack	5 Byte
Publish	25 - 75 Byte
Ping Request	2 Byte
Ping Response	2 Byte

MQTT protokolünün aşamaları Şekil 3.33'te belirtilen fazlar halinde tanımlanmıştır.

	Phase Name	Start Phase After	Source	Destination	Source->Dest Traffic	Dest->Source Traffic
Connect	Connect	Application Starts	P1	SRV	(...)	(...)
Publish	Publish	Connect	P1	SRV	(...)	No Response
Publish2	Publish2	Publish	SRV	S1	(...)	No Response
ConnectS	ConnectS	Application Starts	S1	SRV	(...)	(...)
Subscribe	Subscribe	ConnectS	S1	SRV	(...)	(...)

Şekil 3.33. MQTT aşamalarının OPNET'de tanımlanması

MQTT protokolünün çalışma mantığına göre trafik fazlara ayrılmış ve her fazdaki değerler *Task Configuration* ile benzetime yansıtılmıştır.

(Manual Configuration) Table

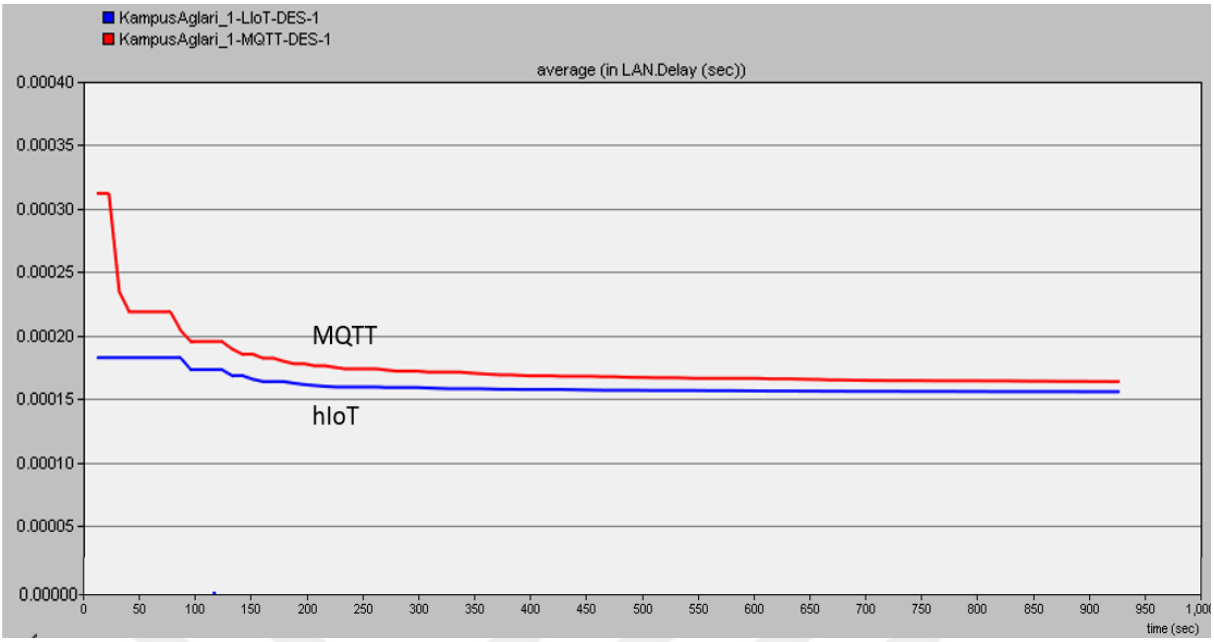
	Phase Name	Start Phase After	Source	Destination	Source->Dest Traffic	Dest->Source Traffic
Register	Register	Application Starts	P1	SRV	(...)	(...)
Search	Search	Register	S1	SRV	(...)	(...)
Query	Query	Register	S1	P1	(...)	(...)

Şekil 3.34. hIoT protokolünün aşamalarının OPNET'de tanımlanması

Tasarlanan protokolünün çalışma mantığında yer alan, Kayıt (Register), Sorgu (Search) ve Veri Transferi (Query) aşamaları *Task Configuration* ile benzetime yansıtılmaktadır (Şekil 3.34).

3.8.2 Gecikme

100 düğümün bulunduğu ortamda gerçekleştirilen Ethernet LAN gecikme testi sonuçları Şekil 3.35'de gösterilmiştir. MQTT protokolünün QoS0 düzeyi ile hIoT protokolünün doğrudan doğruya yöntemleri karşılaştırılmaktadır.

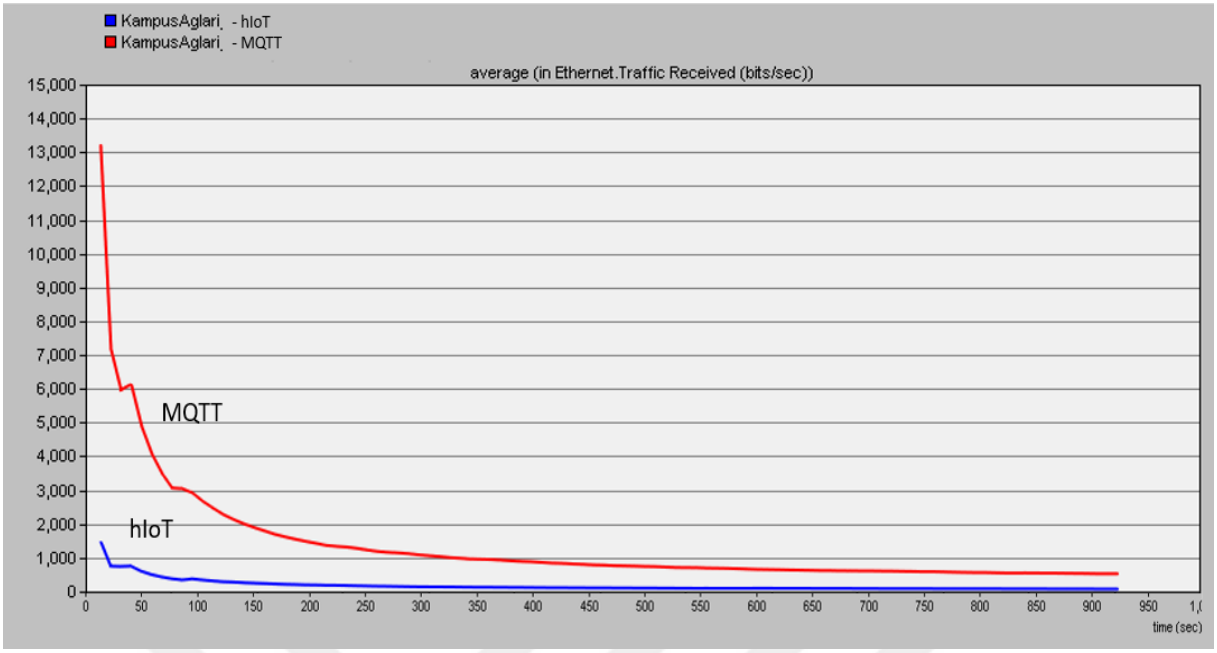


Şekil 3.35. MQTT ve hIoT protokolleri Ethernet gecikme değerleri karşılaştırması

Ethernet'te yaşanan gecikmeler karşılaştırıldığında başlangıçta MQTT ile hIoT arasındaki fark daha yüksek iken sonraki fazlarda bu gecikme arasındaki fark azalmıştır. MQTT'nin Connect aşamasındaki ek yük miktarı ve sunucudan gelen cevabın da yine göreli olarak yüksek yük miktarına sahip olması nedeniyle başlangıçta gecikmeler arasında anlamlı farklılıklar bulunmaktadır. Ancak sonraki aşamalarda (MQTT Publish, Subscribe ve Ping) bu ek yük değerleri ortalama 2 Byte'a kadar düştüğünden gecikmeler arasındaki fark kapanmıştır.

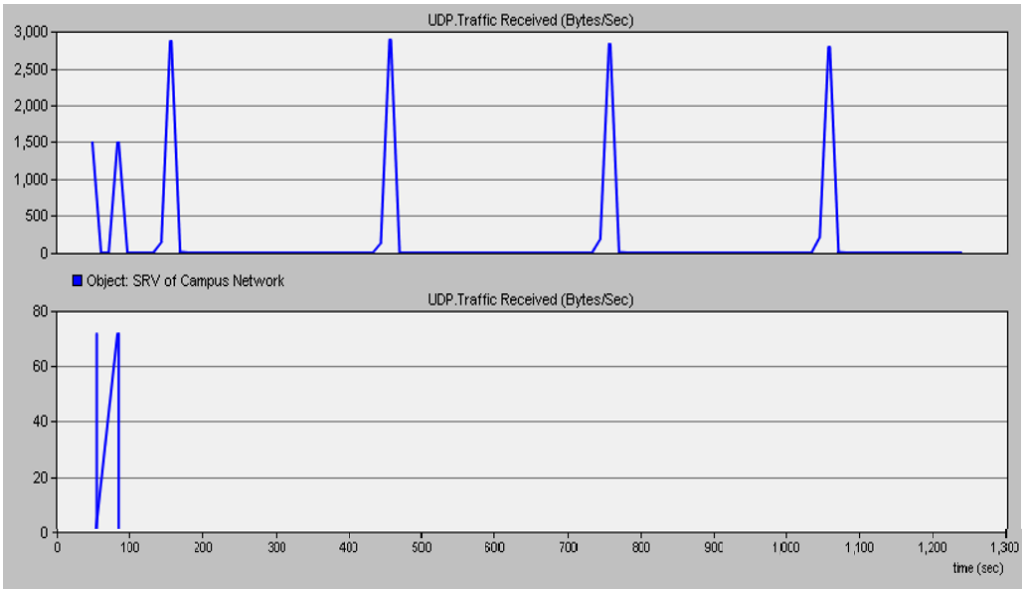
3.8.3 Sunucu trafiği karşılaştırma

MQTT sunucu ile önerilen topolojide sunucularda gerçekleşen trafiklere ilişkin başarım sonuçları Şekil 3.36'da gösterilmektedir. MQTT yapısında iletişimin merkezinde sunucu olduğundan, gecikmesi sabit bir değerde devam etmektedir. Ancak önerilen protokolde, sunucu sadece kayıt ve sorgulama aşamalarında rol aldığından bu aşamalarda daha fazla trafik üretmekte ve daha yüksek gecikmeler göstermektedir.



Şekil 3.36. Sunucu trafiği grafiği

hIoT protokolünün ek yükü daha az olduğundan saniyede alınan paket sayısı zamanla düşmektedir. Sunucunun aldığı paket sayıları ile ağdaki paket sayıları karşılaştırıldığında sunucunun görevinin tamamlanmasının ardından ağda trafik almadığı görünmektedir.

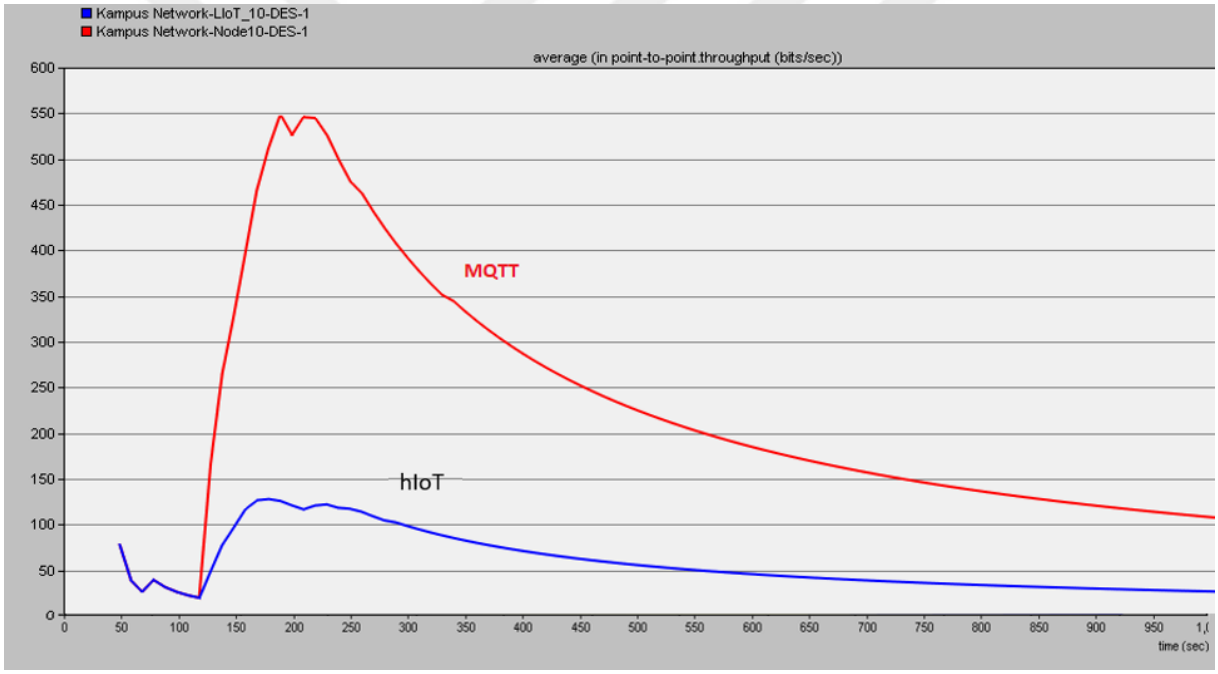


Şekil 3.37. Sensör düğümleri ile sunucu UDP trafiklerinin karşılaştırılması

Sunucu ile sensörlerin aldıkları UDP mesajlarının karşılaştırıldığı Şekil 3.37'de sensörlerden veri alımının devam ettiği ancak sunucunun aktif bir rolünün bulunmadığı görülmektedir. Bu sayede sensörden veri iletişimi yaparken MQTT gibi sunucu tabanlı protokollerde olduğu gibi sunucuya olan bağımlılık ve tek bir hata noktasının bulunması problemi yaşanmamaktadır.

3.8.4 Bant genişliği kullanımı ve işlevsellik karşılaştırmaları

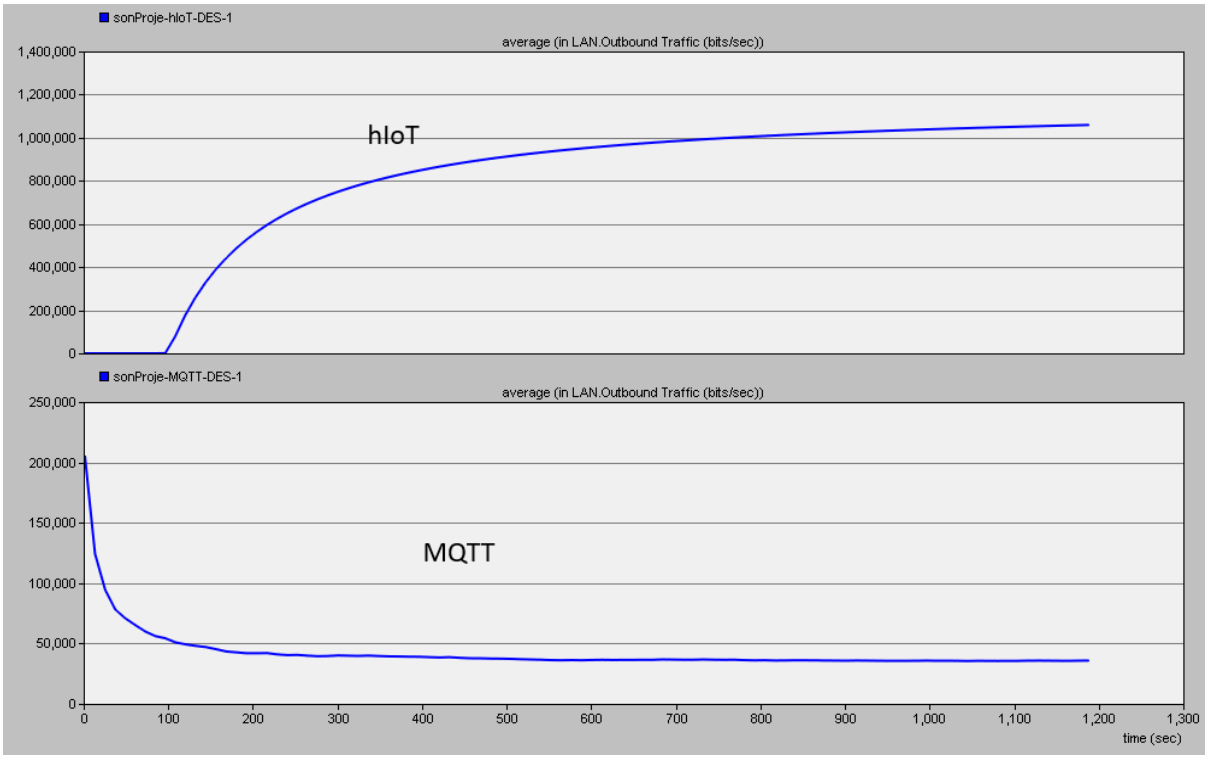
Benzetim yazılımında yapılan diğer bir başarım karşılaştırması da sensörler, kullanıcılar ve sunucunun modellendiği kampüs ağında toplam bant genişliği kullanımıdır. 10 sensörün bulunduğu sensör ağındaki toplam throughput değerlerinin karşılaştırma sonuçları Şekil 3.38'de gösterilmektedir.



Şekil 3.38. Sensör ağında bant genişliği kullanımı

Sensörlerin bulunduğu ethernet ağında MQTT ve hIoT protokolleri için bant genişliği kullanımı zamanla düşmektedir. Buna göre hem sunucu ağında hem de sensör ağında toplam bant genişliği kullanımında, önerilen protokolün genel ortalamada daha düşük bant genişliği harcadığı görülmektedir. Daha düşük bant genişliği kullanımı sadece ağ performansına değil, aynı zamanda düşük güç tüketimine ve sensör ömürlerinin oransal olarak daha uzun olmasına etki etmektedir. Bu nedenle geliştirilen protokol, enerji verimliliği ve maliyet yönünden de avantajlı olmaktadır.

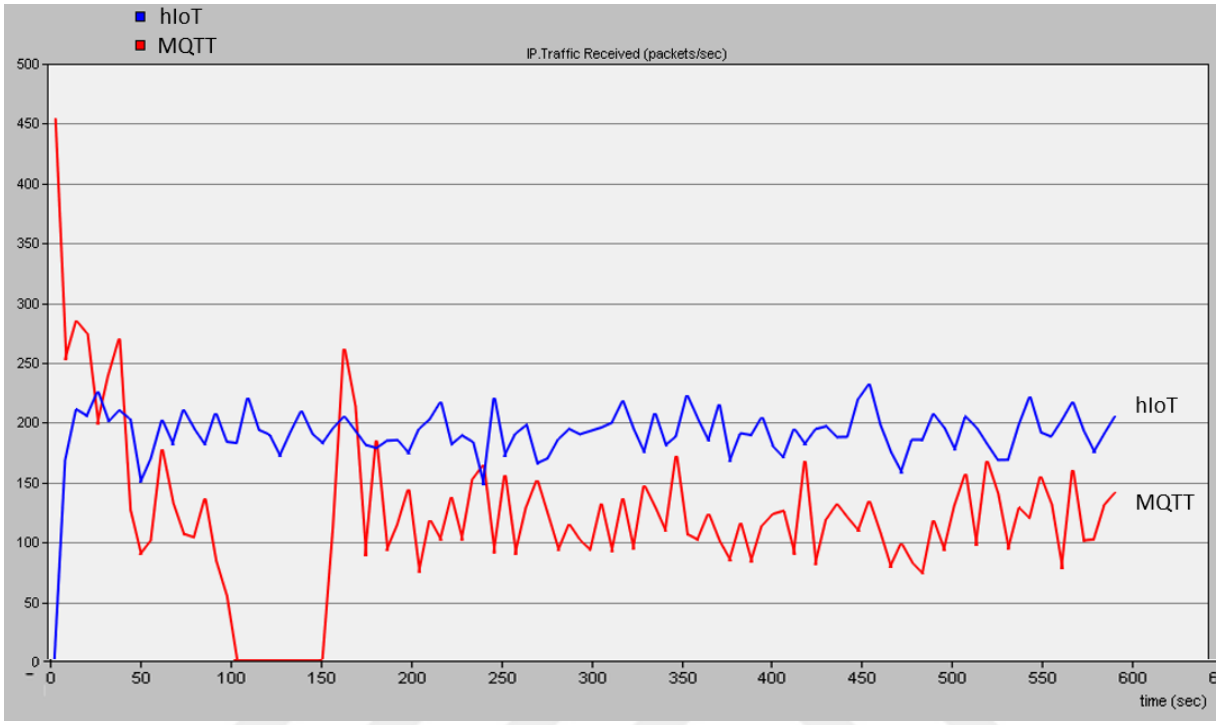
Yapılan performans değerlendirmelerinden biri de, cihaz sayısına göre toplam bant genişliği tüketimi ile ilgilidir. Toplam 100 istemcinin (MQTT'de abone) ve 100 IoT Ağ Geçidinin (MQTT'de yayıncı) bulunduğu bir ortamda, tüm istemcilerin tüm ağ geçitleri ile iletişiminde, 25-75 bayt arasında normal dağılım sergileyen veriler gönderilmiş ve hIoT protokolü ile MQTT protokolleri karşılaştırılmıştır.



Şekil 3.39. Cihaz sayısı arttıkça uçtan uca gerçekleşen iletişim trafiğinin değişimi

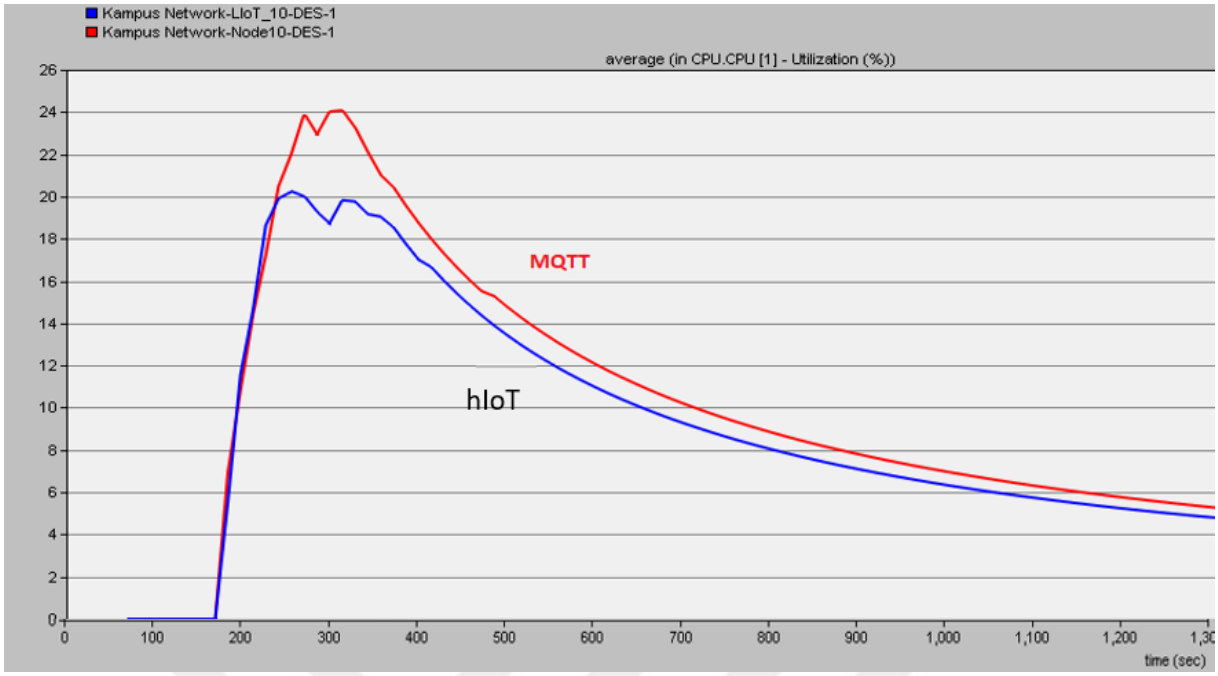
Cihaz sayısı arttıkça, Şekil 3.39'da göründüğü gibi uçtan uca iletişim sayısı artarak devam etmektedir. Bu nedenle, cihaz sayısı arttıkça, MQTT gibi Pub/Sub iletişimler bant genişliği tüketimi yönünden daha iyi performans göstermektedir. Ancak, sunucu tabanlı iletişimde SPOF oluşumu, fonksiyonelliği düşürmektedir. Geliştirilen protokolün fonksiyonelliğinin değerlendirilmesi amacıyla, 100 düğümün bulunduğu ortamda, hIoT protokolünün sunucu tabanlı iletişimi ile MQTT'nin iletişimleri karşılaştırılmıştır. Senaryoda her bir istemci sadece bir IoTGW cihazı ile iletişime geçmiştir. Sunucu tabanlı yaklaşımda MQTT bant genişliği kullanımı yönünden daha yüksek başarım sergilemektedir. Bu durum gerçek ortamda yapılan değerlendirmeler ile de paralellik göstermektedir. Ancak MQTT protokolünün kullanımında, olası sunucu arızası durumunda yayıncılar ile aboneler arasındaki veri akışı kesilecektir. Bu

durum benzetim ortamına aktarılmış ve 100. saniyede sunucunun 50 saniye boyunca arıza yaptığı kurgulanmıştır.



Şekil 3.40. Sunucu arızasında trafik akışları

Hibrit Uygulama Protokolünün fonksiyonelliğinin değerlendirildiği ortamda, sunucu arıza verdiğinde MQTT abonelerin bulunduğu ağ herhangi bir paket alamadığı Şekil 3.40'ta gösterilmektedir. Ancak daha yüksek bant genişliği tüketmesinde rağmen, hIoT protokolünün kullanıldığı ağda herhangi bir iletişim problemi yaşanmamaktadır.



Şekil 3.41. Sensör düğümlerinde CPU kullanımı

İşlemci Kullanımına yönelik yapılan 10 düğümün yer aldığı değerlendirmede (Şekil 3.41), sensör düğümlerinin bulunduğu ağlardaki cihazların CPU tüketimleri incelenmiştir. Hibrit Uygulama Protokolü daha az CPU kullanımı sergilese de MQTT ile arasında büyük bir farklılık bulunmamaktadır.

4. SONUÇ VE ÖNERİLER

Nesnelerin İnternet'inin sunmayı vadettiği avantajlardan faydalanabilmek için, karmaşık yapılardan oluşan IoT ekosistemlerinde kullanılacak uygulama katmanı protokolünün belirlenmesi son derece önemli bir konudur. IoT cihazlarının kullanıldığı uygulamalar temel kavramlar itibariyle birbirine benzese de farklı karakteristik özelliklere sahip olabilirler. IoT ekosisteminde uygulamaların performansı taşınan yük miktarı ve RTT ile doğrudan ilişkilidir. Birçok akademik çalışmada, bant genişliği, gecikme, paket kayıp olasılığı gibi parametrelere bağlı olarak kullanılan uygulama protokollerinin değişken performanslar sergilediği görülmektedir. Herhangi bir ortam için uygun olan bir protokol, ortamda değişiklikler gerçekleştiğinde düşük performans sergileyebilmektedir. Ayrıca, performans ölçütü olarak kullanılan birçok parametre ağda anlık olarak değişebilmektedir.

Bu tez çalışması kapsamında geliştirilen protokol, var olan yaklaşımların karması yeni bir yaklaşım getirmektedir. Protokol hem sunucu tabanlı erişim hem de doğrudan erişim sunmaktadır. Sensörlerden gelen verilerin servis edilmesinde, IoT Ağ Geçidi cihazlarının erişim yöntemini belirleyebilme özelliği önemli bir avantajdır. Böylece aynı ortamda bulunan kimi hizmetlere doğrudan erişim sunarken, kimi hizmetlere ise merkezi sunucu üzerinden erişim verilebilmektedir. Geliştirilen protokol, ağın anlık durumuna göre erişim yöntemini değiştirebilme potansiyeli de sunmaktadır.

Bant genişliği tüketimi IoT ekosisteminde kritik konulardandır. Nesnelerin İnternetinde yaygın olarak kullanılan MQTT gibi protokoller Yayın/Abone mantığında çalışmaktadır. Sunulan hizmetler yayıncı tarafından aboneye sürekli olarak gönderilmektedir. Abone, o an için veriye ihtiyaç duymasa bile yayınlanan veriyi almaktadır. Aynı yapı CoAP protokolünde "değişiklik olduğunda bildirim yapma" şeklinde karşımıza çıkmaktadır. Bu durum gereksiz bant genişliği kullanımına neden olmaktadır. Geliştirilen protokolde ise "isteğe bağlı erişim" yöntemi uygulanmaktadır. Bu sayede veriye gerçekten ihtiyaç duyulduğu anda erişilmekte ve böylece daha az bant genişliği kullanılarak, gereksiz bant genişliği tüketiminin önüne geçilmektedir.

Hibrit uygulama protokolünün (hIoT) sunduğu avantajlardan biri de, içerisinde servis keşif mekanizması barındırmasıdır. Bu sayede ek bir keşif protokolüne ihtiyaç duymadan servis keşfi yapılabilmektedir. Keşif sürecinde sunulan hizmeti tarif eden "*tema*" yapısı, MQTT

protokolündeki "*topic*" ve CoAP protokolündeki "*URI*" yapısı ile uyumlu olacak şekilde tasarlanmıştır. hIoT'nin bu özelliği, bu protokollerle birlikte kullanılması gerektiği uygulamalarda önemli bir avantaj sunmaktadır. Böylece kolaylıkla bir ara katman protokolü olarak, paketler arasında dönüştürme amacıyla da kullanılabilecektir.

Tez çalışması kapsamında gerek gerçek ortamda gerçekleştirilen testlerde gerekse benzetim ortamındaki değerlendirmelerde geliştirilen protokol MQTT protokolü ile karşılaştırılmıştır. Gecikme yönünden incelendiğinde, hIoT'nin uçtan uca doğrudan erişim yönteminde MQTT'ye göre daha düşük gecikme değerine sahip olduğu görülmektedir. Protokollerin performansı ile doğrudan ilişkili olan bu sürenin düşüklüğü yine önemli bir avantajdır. Ancak, sunucu tabanlı gerçekleştirilen iletişimlerde MQTT'ye oranla daha yüksek gecikme değerine sahip olduğu görülmüştür. Bu nedenle, hız gerektiren ve anlık erişimin önemli olduğu hizmetler için, doğrudan erişim yönteminin kullanılması daha uygun olacaktır.

Tüketilen bant genişliği yönünden incelendiğinde, hibrit uygulama protokolünde UDP'nin avantajları kullanılmıştır. Bu sayede daha az bant genişliği tüketilmektedir. Kısıtlı bant genişliğine sahip ortamlarda kullanılması, protokolün önemli bir avantajı olarak görülebilir.

Benzetim ortamında gerçekleştirilen testlerde yüz adet IoT Ağ Geçidine, yüz kullanıcının rastgele zamanlarda yapmak istediği doğrudan erişimlerde bant genişliği tüketiminin arttığı görülmektedir. Bu durum, hIoT'nin bir dezavantajı olarak görülebilir. Bu nedenle, tüm iletişimin "doğrudan erişim" uygulanması performans kayıplarına neden olmaktadır. Bu dezavantajlı durumun çözümü için, ağ geçitlerinin sunduğu hizmetlerin türüne göre erişim yöntemlerini paylaştırmak, bir yöntem olarak kullanılabilir.

Geliştirilen protokolün performans değerlendirmelerinden biri de, taşınan yük miktarına bağlı gecikme süreleri ve yük miktarına bağlı tüketilen bant genişliğidir. Değerlendirmelerde hIoT'nin düşük ek yüklerde daha iyi performans gösterdiği görülmektedir. Özellikle sensörler gibi düşük boyutlarda veri üretmesi beklenen donanımlardan gelen verileri transfer etmek için tasarlandığından, veri taşıma alanı hIoT'de 1024 byte ile sınırlandırılmış ve paket yapısı bu doğrultuda geliştirilmiştir. Daha yüksek boyutlu veri transferi için bu protokol uygun değildir.

IoT ekosisteminde kullanılan cihazlar genellikle sınırlı kapasitelere sahip donanımlar olduğundan, geliştirilen protokol de bu doğrultuda hafif bir yapıda tasarlanmıştır. Kısıtlı

kaynaklara sahip cihazlarda, IoT için son derece önemli olan kimlik doğrulama ve şifreleme gibi güvenlik unsurları bu tez çalışması kapsamında dikkate alınmamıştır. Yeni gelişen teknolojiler doğrultusunda bu cihazların kapasiteleri artacağından, bu gibi güvenlik parametrelerinin uygulanması protokolün daha verimli olmasını sağlayacaktır.

Hibrit Uygulama Protokolünde hem sunucu tabanlı iletişim hem de doğrudan iletişim yeteneği bulunmaktadır. Bu iki yöntem arasında geçiş yapabilme potansiyeli önemli bir avantajdır ancak, çalışma kapsamında iki yöntem arasında dinamik geçişi sağlayan bir protokol geliştirilmemiştir. Bu görev, hIoT protokolünü kullanacak olan üst katman protokole ya da uygulamaya bırakılmıştır. Ağdaki çeşitli ölçütler doğrultusunda yük devri gibi algoritmalar geliştirilerek, uygulama çalışırken erişim yöntemleri arasında dinamik anahtarlama yapılması sağlanabilir. Yine, ağdan sağlanan parametrelere göre trafik yükü önceden kestirilip, erişim yöntemleri arasında geçiş yapılması sağlanabilir.

KAYNAKLAR

1. Liu, X., Zhang, T., Hu, N., Zhang, P and Zhang, Y. (2020, January). The method of Internet of Things access and network communication based on MQTT. *Computer Communication*, 153 (January), 169–176, doi: 10.1016/j.comcom.2020.01.044.

2. Atzori, L., Iera, A. and Morabito, G. (2010, October). The Internet of Things: A Survey, *Computer Networks*, 54(15), 2787–2805, doi:10.1016/j.comnet.2010.05.010.

3. Höller, J., Tsiatsis, V., Mulligan, C., Karnouskos, S., Avesand, S and Boyle, D. (2014). M2M and IoT Technology Fundamentals. In *From Machine-to-Machine to the Internet of Things: Introduction to a New Age of Intelligence, Academic Press, Inc, USA*, 81-143, doi:10.1016/B978-0-12-407684-6.00005-X.

4. Vermesan, O., Fries, P., Guillemin, P., Sundmaeker, H., Eisenhauer,M., Moessner, K., Arndt, M., Spirito, M., Medagliani, P., Giaffreda, R., Gusmeroli, S., Ladid, L., Serrano, M., Hauswirth, M., Baldini G. (2014), Internet of Things Strategic Research and Innovation Agenda. In O.Vermesan, P.Fries (Eds), *Internet of Things – From Research and Innovation to Market Deployment*, 7-112, River Publishers.

5. Ozdogan, E. ve Erdem, O. A. (2020). Nesnelerin İnterneti İçin Hibrit Uygulama Katmanı Protokol Tasarımı. *Mühendislik Bilimleri ve Tasarım Dergisi*, 8(1), 285–304, doi: 10.21923/jesd.530295.

6. Gubbi, J., Buyya, R., Marusic, S. and Palaniswami, M. (2013). Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29(7), 1645–1660, doi:10.1016/j.future.2013.01.010.

7. İnternet: ITU-T, Focus Group on M2M service layer: APIs and protocols overview, International Telecommunication Union, URL: https://www.itu.int/opb/publications .aspx?parent=T-FG&selection=6&sector. Son Erişim Tarihi: 10.05.2020.

8. Florea, I., Rughinis, R., Ruse, L. and Dragomir, D. (2017, May). *Survey of Standardized Protocols for the Internet of Things*. Paper presented at 21st International Conference on Control Systems and Computer, CSCS 2017, 90–196, doi: 10.1109/CSCS.2017.33.

9. Talaminos-Barroso, A., Estudillo-Valderrama, M. A., Roa, L. M., Reina-Tosina, J. and Ortega-Ruiz, F. (2016). A Machine-to-Machine protocol benchmark for eHealth applications - Use case: Respiratory rehabilitation. *Computer Methods Programs in Biomedicine*, 129, 1–11, doi: 10.1016/j.cmpb.2016.03.004.

10. Roman, R., Zhou, J. and Lopez, J. (2013). On the features and challenges of security and privacy in distributed internet of things. *Computer Networks*, 57(10), 2266–2279, doi: 10.1016/j.comnet.2012.12.018.

11. Dizdarevic, J., Carpio, F., Jukan, A. and Masip, X. (2018). Survey of Communication Protocols for Internet-of-Things and Related Challenges of Fog and Cloud Computing Integration. *ACM Computing Survey*, 51, 1–27, doi: 10.1145/3292674.

12. Thota P., and Kim, Y. (2016, December). *Implementation and Comparison of M2M Protocols for Internet of Things.* Paper presented at 4th Intl Conf on Applied Computing and Information Technology/3rd Intl Conferance on Computational Science/Intelligence and Applied Informatics/1st Intl Conf on Big Data, Cloud Computing, Data Science Engineering (ACIT-CSII-BCD), 43–48, doi: 10.1109/ACIT-CSII-BCD.2016.021.

13. Bendele, B. and Akopian, D. (2017). A study of IoT MQTT control packet behavior and its effect on communication delays. *Electronic Imaging*, 2017(6), 120–129, doi: 10.2352 /ISSN.2470-1173.2017.6.MOBMU-311.

14. Al-Nabhan, N., Al-Aboody, N. and Alim Al Islam, A. B. M. (2019). A hybrid IoT-based approach for emergency evacuation. *Computer Networks*, 155, 87–97, doi:10.1016 /j.comnet.2019.03.015.

15. Fathia, O. and Said, R. (2017). Impact of Link Delay Variation on MQTT and CoAP based Communication Performances in Mobile Environment. *IJCSNS International Journal of Computer Science and Network Security*, 17(8), 187–193.

16. Bellavista, P. and Zanni, A. (2016, September). *Towards better scalability for IoT-cloud interactions via combined exploitation of MQTT and CoAP*. Paper presented at IEEE 2nd International Forum Research Technologies for Society and Industry Leveraging a Better Tomorrow, RTSI 2016, doi:10.1109/RTSI.2016.7740614.

17. Klauck, R. and Kirsche, M. (2012, January). *Bonjour Contiki: A Case Study of a DNS-Based Discovery Service for the Internet of Things.* Presented at first Ad-hoc, Mobile, and Wireless Networks, 316–329, doi:10.1007/978-3-642-31638-8_24.

18. Thangavel, D., Ma, X., Valera, A., Tan, H. X. and Tan,C. K. Y. (2014, April). *Performance evaluation of MQTT and CoAP via a common middleware.* Paper presented at IEEE ISSNIP 2014 - 2014 IEEE 9th International Conference on Intelligent Sensors, Sensor Networks and Information Processing, doi: 10.1109/ISSNIP.2014.6827678.

19. Lars, D. (2015, February). *Performance Evaluation of M2M Protocols Over Cellular Networks in a Lab Environment.* Paper presented at 18th International Conference on Intelligence in Next Generation Networks (ICIN), France, 70–75, doi: http://dx.doi.org/10.1109/ ICIN.2015.7073809.

20. Mun, D.H., Le Dinh, M. and Kwon, Y. W. (2016, June). *An Assessment of Internet of Things Protocols for Resource-Constrained Applications*. Paper presented at International Computer Software and Applications Conference, 555-560, doi: 10.1109/COMPSAC. 2016.51.

21. Fysarakis, K., Askoxylakis, I., Soultatos, O., Papaefstathiou, I., Manifavas, C. and Katos, V. (2016, December). *Which IoT Protocol ?*. Paper presented at 2016 IEEE Global Communications Conference, doi: 10.1109/GLOCOM.2016.7842383.

22. Moraes, T., Nogueira, B., Lira, V. and Tavares, E. (2019, October). *Performance Comparison of IoT Communication Protocols*. Paper presented at IEEE International Conference on Systems, Man and Cybernetics, 3249–3254, doi:10.1109/SMC.2019. 8914552.

23. Babovic, Z. B., Protic, J. and Milutinovic, V. (2016). Web Performance Evaluation for Internet of Things Applications. *IEEE Access*, 4, doi: 10.1109/ACCESS.2016.2615181.

24. Luzuriaga, J.E., Perez, M., Boronat, P., Cano, J.C., Calafate, C. and Manzoni, P. (2015, January). *A comparative evaluation of AMQP and MQTT protocols over unstable and mobile networks*. Paper presented at 12th Annual IEEE Consumer Communications and Networking Conference, CCNC 2015, 931–936, doi:10.1109/CCNC.2015.7158101.

25. Shin, I.-J., Song, B.-K. and Eom, D.-S. (2017). International Electronical Committee (IEC) 61850 Mapping with Constrained Application Protocol (CoAP) in Smart Grids Based European Telecommunications Standard Institute Machine-to-Machine (M2M) Environment. *Energies*, 10(3), 393, doi:10.3390/en10030393.

26. Collina, M., Bartolucci, M., Vanelli-coralli, A. and Corazza, G. E. (2014, September). *Internet of Things Application Layer Protocol Analysis over Error and Delay prone Links*. Paper presented at 7th Advanced Satellite Multimedia Systems Conference and the 13th Signal Processing for Space Communications Workshop (ASMS/SPSC), 398-404, doi: 10.1109/ASMS-SPSC.2014.6934573.

27. Kakakhel, S. R. U., Westerlund, T., Daneshtalab, M., Zou, Z., Plosila, J. and Tenhunen, H. (2019, June). *A qualitative comparison model for application layer IoT protocol*. Paper presented ar 4th International Conference on Fog and Mobile Edge Computing, FMEC 2019, 210–215, doi: 10.1109/FMEC.2019.8795324.

28. Naik, N. (2017, October). *Choice of Effective Messaging Protocols for IoT Systems : MQTT , CoAP , AMQP and HTTP*. Paper presented at IEEE International Systems Engineering Symposium (ISSE), 4, 1–8, doi:10.1109/SysEng.2017.8088251.

29. Chaudhary, A., Peddoju, S. K. and Kadarla, K. (2017, October). *Study of Internet-of-Things Messaging Protocols Used for Exchanging Data with External Sources*. Paper presented at 14th IEEE International Conference on Mobile Ad Hoc and Sensor Systems, MASS 2017, 666–671, doi: 10.1109/MASS.2017.85.

30. Tandale, U., Momin, B. and Seetharam, D. P. (2017, August). *An empirical study of application layer protocols for IoT*. Paper presented at 2017 International Conference on Energy, Communication, Data Analytics and Soft Computing ICECDS 2017, 2447–2451, doi: 10.1109/ICECDS.2017.8389890.

31. Choi, D., Jung, J., Kang, H. and Koh, S. (2017, February). *Cluster-based CoAP for Message Queueing in Internet-of-Things Network.* Paper presented at 19th International Conference on Advanced Communication Technology (ICACT), 584-588, doi:10.23919/ICACT. 2017.7890157.

32. Chen, Y. and Kunz, T. (2016, April). *Performance evaluation of IoT protocols under a constrained wireless access network*. Paper presented at International Conference on Selected Topics in Mobile and Wireless Networking (MoWNeT), doi:10.1109 /MoWNet.2016.7496622.

33. Huo, C. (2014). *A Centralized IoT Middleware System for DevicesWorking Across Application Domains Using Self-descriptive Capability Profile*. (Doctoral dissertation, University Of California, 2014).

34. Lee, C., Chang, Y., Chuang, C. and Lai, Y. H. (2016, October). *Interoperability enhancement for Internet of Things protocols based on software-defined network*. Paper presented at IEEE 5th Global Conference on Consumer Electronics, 1–2, doi:10.1109 /GCCE.2016.7800510.

35. Desai, P., Sheth, A. and Anantharam, P. (2015, June). *Semantic Gateway as a Service Architecture for IoT Interoperability*. Paper presented at 2015 IEEE International Conference on Mobile Services, 313–319, doi:10.1109/MobServ.2015.51.

36. Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M. and Ayyash, M. (2015). Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. *IEEE Communications Surveys & Tutorials*, 17(4),2347-2376, doi: 10.1109/ COMST.2015.2444095.

37. Mattern F. and Floerkemeier C. (2010). From the Internet of Computers to the Internet of Things. In: Sachs K., Petrov I., Guerrero P. (Eds) *From Active Data Management to Event-Based Systems and More. Lecture Notes in Computer Science*, Springer, Berlin, Heidelberg, pp.242-259. doi: 10.1007/978-3-642-17226-7_15.

38. İnternet: Cisco INC, Cisco Networking Academy- IoT Security Cisco INC., 2018. URL: https://static-course-assets.s3.amazonaws.com /IoTFIoTSec10/ en/index. html# 5.1.3.5. Son Erişim Tarihi: 15.05.2020.

39. Daş, R. (2018). Nesnelerin İnterneti : Gelişimi, Bileşenleri ve Uygulama Alanları. *Pamukkale Üniversitesi Mühendislik Bilimleri Dergisi*, 24(2), 327–335, doi:10.5505/ pajes.2017.89106.

40. Da Cruz, M. A. A., Rodrigues, J. J. P. C., Al-Muhtadi, J., Korotaev, V. V. and De Albuquerque, V. H. C. (2018). A Reference Model for Internet of Things Middleware. *IEEE Internet of Things Journal*, 5(2), 871–883, doi: 10.1109/JIOT.2018.2796561.

41. İnternet: Towards a Definition of the Internet of Things (IoT), IEEE Internet of Things, 2015. URL: https://iot.ieee.org/definition.html. Son Erişim Tarihi: 24.04.2020.

42. İnternet: The Internet of Things, IETF, 2019. URL: https://ietf.org/topics/iot/. Son Erişim Tarihi: 23.04.2020.

43. İnternet: International Telecommunication Union, Overview of the Internet of things, URL: https://www.itu.int/rec/dologin_pub.asp?lang=e&id= T-REC-Y.2060-201206-I!!PDF-E&type=items. Son Erişim Tarihi: 15.05.2020.

44. Perera, C. (2017). *Sensing as a Service for Internet of Things: A Roadmap*. Leanpub Publisher, 39-58.

45. C. Perera, Member, C. H. L., Jayawardena, S. and Chen, M. (2015). A Survey on Internet of Things From Industrial Market Perspective. *IEEE Access*, 2, doi: 10.1109/ACCESS .2015.2389854.

46. Nunes, L. H., Estrella, J. C., Delbem, A. C. B., Perera, C. and Reiff-Marganiec, S. (2016, November). *The Effects of Relative Importance of User Constraints in Cloud of Things Resource Discovery: A Case Study*. Paper presented at 9th International Conference on Utility and Cloud Computing, Shanghai, doi: 10.1145/2996890.3007867.

47. Sohraby, K., Minoli, D. and Znati, T. (2007). *Wireless Sensor Networks*. New Jersey: John Wiley & Sons Inc, 5- 31.

48. Sethi, P. and Sarangi, S. R. (2017). Internet of Things: Architectures, Protocols, and Applications. *Journal of Electrical and Computer Engineering*, 2017(1), 1-25, doi: 10.1155/2017/9324035.

49. Swathi, K., Sandeep, T. U. and Ramani, A. R. (2018). Performance Analysis of Microcontrollers Used In Iot Technology. *International Journal of Scientific Research in Science, Engineering and Technology IJSRSET*, 4(4). 1268–1273.

50. Mendoza, J. F., Ordóñez, H., Ordóñez, A. and Jurado, J. L. (2017). Architecture for embedded software in microcontrollers for Internet of Things (IoT) in fog water collection. *Procedia Computer Science*, 109:1092–1097, doi:10.1016/j.procs.2017 .05.395.

51. Schlautmann, A. (2011, March). *Embedded Networking Systems in the Smart Home & Office*. Paper presented at M2M Zone Conference at the International CTIA Wireless.

52. Glória, A., Cercas, F. and Souto, N. (2017). Design and implementation of an IoT gateway to create smart environments. *Procedia Computer Science*, 109, 568–575, doi: 10.1016/j.procs.2017.05.343.

53. Kashyap, M., Sharma, V. and Gupta, N. (2018). Taking MQTT and NodeMcu to IOT : Communication in Internet of Things. *Procedia Computer Science*, 132, 1611–1618, doi: 10.1016 /j.procs.2018.05.126.

54. Hentschel, K., Jacob, D., Singer, J. and Chalmers, M. (2016, August). *Supersensors: Raspberry Pi devices for smart campus infrastructure*. Paper presentet at IEEE 4th International Conference Future Internet Things Cloud, FiCloud 2016, 58–62, doi: 10.1109/ FiCloud. 2016.16.

55. Borgohain, T. and Sanyal, S. (2015). Survey of Security and Privacy Issues of Internet of Things. *International Journal of Advanced Networking Applications*, 6, 2372-2378.

56. Perera, C., Liu, C. H. and Jayawardena, S. (2015). The Emerging Internet of Things Marketplace from an Industrial Perspective: A Survey. *IEEE Transactions on Emerging Topics in Computing*, 3(4), 585–598, doi: 10.1109/TETC.2015.2390034.

57. Acer, U. G., Boran, A., Forlivesi, C., Liekens, W., Perez-Cruz, F. and Kawsar, F. (2015, October). *Sensing WiFi network for personal IoT analytics*. Paper presented at 5th International Conference on Internet of Things, 104–111, doi:10.1109/IOT.2015 .7356554.

58. Rajeesh Kumar, N. V., Praveen, B. S. B., Sarveswara Reddy, A. V. and Baron Sam, B. (2017, February). *Study on IOT with reference of M2M and WiFi*. Paper presented at International Conference on Information Communication and Embedded Systems, ICICES 2017, doi: 10.1109/ICICES.2017.8070754.

59. Söğüt, E. and Erdem, O. A. (2019, August). *Performance Comparison of the IEEE 802 .15.4 Standard (ZigBee) Topologies*. Paper presented at 8th International Conference on Advanced Technologies (ICAT'19), 4–9.

60. Huh, J., Je, S. and Seo, K. (2015). Design and Simulation of Foundation Technology for Zigbee-based Smart Grid Home Network System using OPNET Simulation. *Asia-pacific Journal of Multimedia Services Convergent with Art, Humanities, and Sociology*, 5(4), 81–89.

61. Ali, A. I., Partal, S. Z., Kepke, S. and Partal, H. P. (2019, June). *ZigBee and LoRa based Wireless Sensors for Smart Environment and IoT Applications*. Paper presented at IEEE 1st Global Power, Energy and Communication Conference GPECOM, 19–23, doi: 10.1109/GPECOM.2019.8778505.

62. Elkhodr, M., Shahrestani, S. and Cheung, H. (2016). Emerging Wireless Technologies in the Internet of Things : A Comparative Study. *International Journal of Wireless & Mobile Networks*, 8(5), 67–82, doi: 10.5121/ijwmn.2016.8505.

63. Tan, L. and Wang, N. (2010, August). *Future internet: The Internet of Things*. Paper presented at 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE), 5, V5-376-V5-380, doi: 10.1109/ICACTE.2010.5579543.

64. Khan, R., Khan, S. U., Zaheer, R. and Khan, S. (2012, December). *Future Internet: The Internet of Things Architecture, Possible Applications and Key Challenges*. Paper presented at 10th International Conference on Frontiers of Information Technology, 257–260, doi: 10.1109/FIT.2012.53.

65. Wu, M., Lu, T-J., Ling, F-Y., Sun, J. and Du, H-Y. (2010, August). *Research on the architecture of Internet of Things*. Paper presented at 3rd International Conference on Advanced Computer Theory and Engineering(ICACTE), 5, V5-484-V5-487, doi: 10.1109/ ICACTE.2010.5579493.

66. Antão, L., Pinto, R., Reis, J. and Gonçalves, G. (2018, April). *Requirements for Testing and Validating the Industrial Internet of Things*. Paper presented at International Conference on Software Testing, Verification and Validation Workshops (ICSTW), 110–115, doi: 10.1109/ICSTW.2018.00036.

67. İnternet: ITU-T. (2014). Focus Group on M2M service layer: APIs and protocols overview. URL: https://www.itu.int/opb/publications.aspx?parent=T-FG&selection =6&sector, Son Erişim Tarihi: 18.02.2020.

68. Karaagac, A., VanEeghem, M., Rossev, J., Moons, B., DePoorter E. and Hoebeke, J. (2018, October). *Extensions to LwM2M for Intermittent Connectivity and Improved Efficiency*. Paper presented at IEEE Conference on Standards for Communications and Networking (CSCN), Paris, 1-6, doi: 10.1109/CSCN.2018.8581821.

69. Mamo, F. T. and Sikora, A. (2015, September). *Implementation of standardized 6LoWPAN based application layer protocols*. Paper presented at IEEE 8th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), Warsaw, 817-822, doi:10.1109/IDAACS .2015.7341417.

70. Datta, S. K., Da Costa, R. P. F. and Bonnet, C. (2015, December). *Resource discovery in Internet of Things: Current trends and future standardization aspects*. Paper presented at 2015 IEEE 2nd World Forum on Internet of Things (WF-IoT), 542–547, doi:10.1109/WF-IoT.2015.7389112.

71. Ccori, P. C., De Biase, L. C. C., Zuffo, M. K. and da Silva, F. S. C. (2016, September). *Device discovery strategies for the IoT*. Paper presented at IEEE International Symposium on Consumer Electronics (ISCE), Sao Paulo, 97-98, doi: 10.1109 /ISCE.2016.7797388.

72. Zhu, F., Mutka, M. W. and Ni, L. M. (2005). Service discovery in pervasive computing environments. *IEEE Pervasive Computer*, 4(4), 81–90, doi: 10.1109/MPRV.2005.87.

73. Aziez, M., Benharzallah, S. and Bennoui, H. (2017, April). *Service discovery for the Internet of Things: Comparison study of the approaches*. Paper presented at 4th International Conference on Control, Decision and Information Technologies (CoDIT), Barcelona, 0599-0604, doi: 10.1109/CoDIT.2017.8102660.

74. Jara, A. J., Martinez-Julia, P. and Skarmeta, A. (2012, July). *Light-Weight Multicast DNS and DNS-SD (lmDNS-SD): IPv6-Based Resource and Service Discovery for the Web of Things.* Paper presented at Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, Palermo, 731-738, doi:10.1109 /IMIS.2012.200.

75. Mijovic, S., Shehu, E. and Buratti, C. (2016, September). *Comparing application layer protocols for the Internet of Things via experimentation.* Paper presented at IEEE 2nd Internationl Forum Research Technology Society and Industry Leveraging a Better Tomorrow, RTSI 2016, doi:10.1109/RTSI.2016.7740559.

76. Yassein, M.B., Shatnawi, M.Q. and Al-zoubi, D. (2016, September). *Application layer protocols for the Internet of Things: A survey*. Paper presented at International Conference on Engineering & MIS (ICEMIS), Agadir, 1-4, doi:10.1109/ICEMIS .2016.7745303.

77. Tantitharanukul, N., Osathanunkul, K., Hantrakul, K., Pramokchon P. and Khoenkaw, P. (2017, March). *MQTT-Topics Management System for sharing of Open Data*. Paper presented at International Conference on Digital Arts, Media and Technology (ICDAMT), Chiang Mai, 62-65, doi:10.1109/ICDAMT.2017.7904935.

78. Lee, S., Kim, H., Hong, D. and Ju, H. (2013, January). *Correlation analysis of MQTT loss and delay according to QoS level*. Paper presented at International Conference on Information Networking 2013 (ICOIN), Bangkok, 714-717, doi:10.1109/ICOIN .2013.6496715.

79. İnternet: A. Banks and R. Gupta, MQTT Version 3.1.1 Edited by Andrew Banks and Rahul Gupta, OASIS, 2014. URL: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html. Son Erişim Tarihi: 24.04.2020.

80. Ishaq, I., Carels, D., Teklemariam, G.K., Hoebeke, J., Abeele, F.V., Poorter, E.D., Moerman, I., Demeester, P. (2013). IETF Standardization in the Field of the Internet of Things (IoT): A Survey. *Journal of Sensor and Actuator Networks*, 2(2), 235-287, doi:10.3390/jsan2020235.

81. Herrero, R. (2019). Analytical model of IoT CoAP traffic. *Digital Communication Networks*, 5(2), 63–68, doi: 10.1016/j.dcan.2018.07.001.

82. Happ, D., Karowski, N., Menzel, T., Handziski, V., and Wolisz, A. (2017). Meeting IoT platform requirements with open pub/sub solutions. *Annals of Telecommunications*, 72, 41-52, doi:10.1007/s12243-016-0537-4.

83. Kang, B., Kim, D. and Choo, H. (2017). Internet of Everything: A Large-Scale Autonomic IoT Gateway. *IEEE Transactions on Multi-Scale Computing Systems*, 3(3), 206–214, doi: 10.1109/TMSCS.2017.2705683.

84. Nunes, L.H., Estrella, J.C., Nakamura, L.H., Libardi, R.M., Ferreira, C.H., Jorge, L., Perera, C. and Reiff-Marganiec, S. (2016). A Distributed Sensor Data Search Platform for Internet of Things Environments. *International Journal of Services Computing*, arxiv:abs/1606.07932.

85. Kim, S.M., Choi, H.S. and Rhee, W.S. (2015, August). *IoT home gateway for auto-configuration and management of MQTT devices*. Paper presented at IEEE Conference on Wireless Sensors (ICWiSe), 12–17, doi:10.1109/ICWISE.2015.7380346.

86. İnternet: Z. Shelby, K. Hartke, and C. Bormann, The Constrained Application Protocol (CoAP), URL: https://tools.ietf.org/html/rfc7252. Son Erişim Tarihi: 22.04.2020.

87. Grygoruk, A. and Legierski, J. (2016). IoT gateway – implementation proposal based on Arduino board. *Proceedings of the Federated Conference on Computer Science*, 8, 1011–1014, doi:10.15439/2016F283.

88. Johnsen, F.T., Landmark, L., Hauge, M., Larsen, E. and Kure, O. (2018, October). *Publish/Subscribe Versus a Content-Based Approach for Information Dissemination.* Paper presented at IEEE Military Communications Conference (MILCOM), 1–9, doi: 10.1109/MILCOM.2018.8599786.

EKLER

EK-1 IoT Koordinatör Örnek Kodu

```
################################################################################
###                                                                          ###
###                      Erdal OZDOGAN (erdalozd@gmail.com)                  ###
###         Doktora Tezi Çalışması - Gazi Üniversitesi Bilişim Enstitüsü     ###
###                  IoT için Uygulama Katmanı Protokol Tasarımı             ###
###                   Paket Manipülasyonu : Python Scapy                     ###
###                   NoSQL Veritabanı    : MongoDB                          ###
###                   Örnek IoT Koordinatör Kodları                          ###
###    Not: Scapy kısıtları nedeniyle, paket alan uzunlukları değiştirilmiştir    ###
###                                Mayıs 2020                                ###
################################################################################


import logging
logger = logging.getLogger("scapy")
logger.setLevel(logging.INFO)
logger.addHandler(logging.StreamHandler())

from scapy.all import *
import random
import pymongo

remoteIP="192.168.1.1"
localIP="192.168.1.103"
dstPort=1079
srcPort=1079

paketTR={"0":"Control","1":"Reset","2":"Register","3":"Error","4":"Query","
5":"Reply","6":"Request","7":"Response"}
hataKodlar={"0":"Unknown","1":"DB_Save","2":"NoTopic","3":"TopicUnreach","4
":"NoAuthor"}


class hIoT(Packet):
    name="Hybrid IoT Protocol"
    #0: Control, 1: Reset, 2: Register, 3:Error, 4:Query, 5:Reply,
6:Request, 7:Response
    fields_desc=[BitEnumField("Type",0,8, {0:"Control",
                                           1:"Reset",
                                           2:"Register",
                                           3:"Error",
                                           4:"Query",
                                           5:"Reply",
                                           6:"Request",
                                           7:"Response"}),
#default type=Control
                 FlagsField("Flags",2,8,["ACK",
                                         "RST",
                                         "DC",
                                         "SRV",
                                         "EX"]),
                 X3BytesField("PacketID",3),
# default ID=0x3 olsun (hex)
                 StrField("Data","Erd/al") ,     # default topic = erd/al
                 StrField("exData","")]          # exData varsayılan boş
    def __str__(self):
        return next(iter(self)).build()
```

EK-1 (Devam) IoT Koordinatör Örnek Kodu

```
def sendPkt(tip,bayrak,pID,Veri,xVeri,dIP):
    bind_layers(UDP,hIoT,sport=1079)
    bind_layers(hIoT,Ether)
    p=IP(dst=dIP)/UDP(sport=1079,
dport=1079)/hIoT(Type=tip,Flags=bayrak,PacketID=pID,Data=Veri,exData=xVeri)
    print("Gönderilen Paket Türü :",paketTR[str(tip)]," ; Flaglar
:",p.Flags)
    print(p.show())
    i=1
    while i<=1:
        i+=1
        sendp(Ether()/p)


def randomPacketIDGenerator():
    return random.randint(0,511) #kolaylık olsun diye alan kısaltıldı. 2^24
desteklenmektedir


def saveToDB(hIoT):
    print("Topic Kaydedildi: ",str(hIoT.Data))
    print("Time : ?",hIoT.exData)
    myAdd={"Topic":
str(hIoT.Data),"IP":"192.168.1.111","DC":str(hIoT.Flags.DC),"CacheTime":hIo
T.exData}
    connectDB(myAdd)
    print("Kaydedilen Topic:",hIoT.Data)

def connectDB(myDict):
    myclient=pymongo.MongoClient("mongodb://localhost:27017/")
    mydb=myclient["hIoTDB"]
    mycol=mydb["hIoTDB"]
    x=mycol.insert_one(myDict)

def deleteFromDB(hIoT):
    myclient=pymongo.MongoClient("mongodb://localhost:27017/")
    mydb=myclient["hIoTDB"]
    mycol=mydb["hIoTDB"]
    myQuery={"Topic":hIoT.Data}
    mycol.delete_one(myQuery)

def requestFromDB(topic):
    myclient = pymongo.MongoClient("mongodb://localhost:27017/")
    mydb = myclient["hIoTDB"]
    mycol = mydb["hIoTDB"]
    myquery = { "Topic": topic }
    mydoc = mycol.find_one(myquery)
    try:
        print("Service IP Bulundu : ",mydoc["IP"])
        #return(mydoc["IP"])
    except ValueError:
        print("Kayıt Yok")
    finally:
        print("......")
```

EK-1 (Devam) IoT Koordinatör Örnek Kodu

```
def queryFromDB(topic):
    myclient = pymongo.MongoClient("mongodb://localhost:27017/")
    mydb = myclient["hIoTDB"]
    mycol = mydb["hIoTDB"]
    myquery = { "Topic": str(topic) }

    mydoc = mycol.find_one(myquery)
    queryValues={"IP":"1.1.1.1","DC":"0"}
    print(" DB'de sorgulanan Topic:",topic)
    for x in mydoc:
        #print("Data Base Kayıt : ",x)
        queryValues["IP"]=mydoc["IP"]
        queryValues["DC"]=mydoc["DC"]
        print("Servis IP:",mydoc["IP"])
        return queryValues
    else:
        print("Topic Yok")

def createhIoTPacket(tip, bayraklar, pID, Veri, exVeri):
    newhIoT=hIoT()
    print("Oluşturulan Paket 1: ",newhIoT.show())
    newhIoT.Type=tip
    newhIoT.Flags=bayraklar
    newhIoT.PacketID=pID
    newhIoT.Data=Veri
    newhIoT.exData=exVeri
    print("Oluşturulan Paket 2: ",newhIoT.show())
    return newhIoT

def packetAnalyzer(hIoT):
    tur=hIoT.Type
    bayrak=hIoT.Flags
    print("Analize Giren PaketID : ",hIoT.PacketID, " tür=",tur,"bayrak=",
bayrak,"Data=", hIoT.Data)

    if tur==1:
        print("Reset") #işleme gerek yok

    elif tur==2 and hIoT.Flags.RST: # De-Register
            print("DB Sil")
            #db Kayıt Sil
            deleteFromDB(hIoT)
            #registerAck oluştur
            #Kapsülle ve Gönder
            sendPkt(2,9,hIoT.PacketID,hIoT.Data,"",remoteIP)

    elif tur==2 and hIoT.Flags.RST==False: # Register
            print("DB Kayıt")
            #db Kayıt Yap
            saveToDB(hIoT)
            #registerACK oluştur
            #Kapsülle ve gönder
            sendPkt(2,5,hIoT.PacketID,hIoT.Data,"",remoteIP)

    elif tur==4: # Query
        print("Query Başladı, Sorgulanan topic: ",str(hIoT.Data))
        # DB den sorgula
```

EK-1 (Devam) IoT Koordinatör Örnek Kodu

```
        gelen=queryFromDB(hIoT.Data)
        serviceIP=(gelen["IP"])
        dcMode=gelen["DC"]
        if dcMode=="1":
            print("DOĞRUDAN ERİŞİM DESTEKLİYOR")
            sendPkt(5,28,hIoT.PacketID,hIoT.Data,serviceIP,remoteIP)
        else:
            print("SUNUCU TABANLI ERİŞİM DESTEKLİYOR",dcMode)
            sendPkt(5,24,hIoT.PacketID,hIoT.Data,localIP,remoteIP)

    elif tur==3: # Error (seçimlik Logla)
        print("Error Süreci Başladı ")
        sendPkt(3,8,hIoT.PacketID,hIoT.Data,hataKodlar["0"],remoteIP)
        #print(tmp.show())


    elif tur==5: # Reply işleme gerek yok
        print("Reply")

    elif tur==6: # Request
        print("Request Başladı....")
        #data Extract yap
        print("İstenen Topic:",hIoT.Data)
        proxySrcIP=remoteIP
        topic=hIoT.Data
        #Gondericiyi Cache'te sakla
        #Servis IP yi bul
        serviceIP=requestFromDB(hIoT.Data)
        #IPye Request At
        sendPkt(6,8,hIoT.PacketID,topic,"",serviceIP)

    elif tur==7: # Response
        print("Response")
        proxyData=hIoT.Data
        sendPkt(7,9,hIoT.PacketID,proxyData,"",proxySrcIP)

    else:
        print("Control")
        sendPkt(0,1,hIoT.PacketID,hIoT.Data,"",remoteIP)

def udpOku():
    print(" UDP Paket Okuma Başladı")

def main():
    myText="Erdal"
    print(myText[2:(len(myText))])


if __name__ == "__main__":
    interact(mydict=globals(), mybanner="Test add-on v3.14")
    #main()
```

ÖZGEÇMİŞ

Kişisel Bilgiler

Soyadı, adı : ÖZDOĞAN, Erdal

Uyruğu : T.C.

Doğum tarihi ve yeri : 14.06.1977, Tatvan

Medeni hali : Evli

Telefon : 0 506 298 72 90

e-mail : erdalozdogan@hotmail.com



Eğitim

Derece	Eğitim Birimi	Mezuniyet Tarihi
Doktora	Gazi Üniversitesi / Bilişim Enstitüsü	Devam ediyor
Yüksek lisans	Gazi Üniversitesi / Bilişim Enstitüsü	2011
Lisans	Anadolu Üniversitesi / Yönetim Bilişim Sistemleri	2019
Lisans	Ankara Üniversitesi / Astronomi ve Uzay Bilimleri	2000
Lise	Tatvan Atatürk Lisesi	1994

İş Deneyimi

Yıl	Yer	Görev
2000-Hâlen	Millî Eğitim Bakanlığı	Öğretmen

Yabancı Dil

İngilizce

Yayınlar

- Özalp, A.N., Özdoğan, E., & Özeren, A. (2011). Milli Eğitim Bakanlığı Cisco Netacad Eğitimlerinin İncelenmesi. *Akademik Bilişim – İnönü Üniversitesi*

- Özdoğan, E. ve Erdem, O.A. (2020). Nesnelerin İnterneti İçin Hibrit Uygulama Katmanı Protokol Tasarımı. *Mühendislik Bilim. ve Tasarım Derg.*, 8(1), 285–304, Mar. 2020, doi: 10.21923/jesd.530295

- Özdoğan, E. ve Erdem, O.A. (2019). *Hybrid Application Layer Protocol Design for IoT Environments.* 6th International Conference on Computer Science, Engineering and Information Technology.,2019, 267-286 doi :https://doi.org/10.5121/csit.2019.91322

Hobiler

Fotoğrafçılık, Robotik



GAZİLİ OLMAK AYRICALIKTIR.